UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CIÊNCIAS DA COMPUTAÇÃO

Bruno Izaias Bonotto Souza

Bloco Acelerador em Hardware para Predição Intra Quadros do Padrão HEVC

Florianópolis
2019

Bruno Izaias Bonotto Souza

Bloco Acelerador em Hardware para Predição Intra Quadros do Padrão HEVC

Trabalho de Conclusão de Curso do Curso de Graduação em Ciências da Computação do Centro Tecnológico da Universidade Federal de Santa Catarina como requisito para obtenção do título de Bacharel em Ciências da Computação.
Orientador: Ismael Seidel, Me.
Coorientador: Prof. José L. A. Güntzel, Dr.

Florianópolis
2019

Bonotto, Bruno
Bloco Acelerador em Hardware para Predição Intra Quadros do Padrão HEVC / Bruno Bonotto; orientador, Ismael Seidel; coorientador, José Luís A. Güntzel, 2019.
503 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Departamento de Informática e Estatística em Ciências da Computação, Florianópolis, 2019.

Inclui referências

1. Ciências da Computação. 2. Codificação de vídeo digital. 3. HEVC. 4. Predição intra quadros. 5. Arquitetura de hardware. I. Seidel, Ismael. II. Luís A. Güntzel, José. III. Universidade Federal de Santa Catarina. Departamento de Informática e Estatística em Ciências da Computação. IV. Título.

Bruno Izaias Bonotto Souza

Bloco Acelerador em Hardware para Predição Intra Quadros do Padrão HEVC

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de Bacharel em Ciências da Computação e aprovado em sua forma final pelo curso de Graduação em Ciências da Computação.

Florianópolis, 9 de Dezembro de 2019.

Prof. José Francisco Fletes, Dr.
Coordenador do Curso

Banca Examinadora:

Ismael Seidel, Me.
Orientador
Universidade Federal de Santa Catarina

Prof. José L. A. Güntzel, Dr.
Coorientador
Universidade Federal de Santa Catarina

Prof^{a}. Cristina Meinhardt, Dr^{a}.
Avaliador
Universidade Federal de Santa Catarina

Daniel Ferrão, Me.
Avaliador
Chipus Microeletronics

Aos que me apoiaram nessa jornada.

AGRADECIMENTOS

A minha mãe, Renata Bonotto, por acreditar e apoiar meus sonhos. Por depositar em mim tamanha confiança e, principalmente, pela formação humana e ética.

As minhas irmãs, Brenda e Heloiza, pela compreensão durante os quatro anos em que não lhes dei a devida atenção.

Aos meus melhores amigos, Aline e Henrique, pela amizade fiel e verdadeira. Pelos diversos conselhos e por acreditarem em meu potencial.

A minha família e demais amigos, pelo apoio e carinho. Por fazerem eu me sentir único e ter vários lugares para chamar de lar.

Ao meu amigo, João Vicente Souto, por ser uma pessoa de tamanha dedicação. Por ter me inspirando e motivado a levar os estudos tão a sério.

Aos amigos e colegas do ECL e NIME, em especial ao Luiz Henrique Cancellier, por todas as conversas e conselhos durante a graduação. Por me proporcionar tamanho crescimento e amadurecimento enquanto cientista da computação e também como pessoa.

Ao meu orientador, Ismael Seidel, por todo o tempo dedicado a mim e ao meu TCC. Pelos conselhos durante a graduação e por ser um exemplo de pesquisador e pessoa.

Ao professor José L. A. Güntzel, por me conceder a chance de trabalhar em seu laboratório. Consequentemente, por permitir a minha permanência na universidade.

“O medo de arriscar é o que te impede de evoluir.
(Itachi Uchiha) ”

RESUMO

O aumento na utilização de aplicações com *streaming* de vídeo, tais como *Youtube* e *Netflix*, demanda a criação de novas técnicas de compressão de vídeo para viabilizar a transmissão via rede. Além disso, a compressão de vídeo é indispensável quando se trata do armazenamento desse tipo de dado, já que arquivos de vídeos em resolução Ultra-HD e maiores, sem compressão, chegam a ocupar até dezenas de *gigabytes* por minuto. Por exemplo, um vídeo em resolução 4K com 30 quadros por segundo e 3 *bytes* para representar cada pixel possui, aproximadamente, 44,8 *gigabytes* de dados por minuto. Para aumentar a taxa de compressão, codificadores de padrões do estado-da-arte apresentam complexidade 10 vezes maiores que aqueles de padrões anteriores. Ainda, com o objetivo de viabilizar a compressão em dispositivos móveis operados à bateria, as etapas mais complexas do codificador devem ser executadas em *hardware* dedicado que, por sua vez, deve ser projetado para ser energeticamente eficiente. Este trabalho apresenta o projeto, descrição e avaliação de um bloco acelerador em *hardware* otimizado para realizar o processo de predição intra quadros para o padrão de codificação *High Efficiency Video Coding* (HEVC). Além disso, foram utilizadas técnicas de projeto e síntese de baixa potência visando reduções no consumo de energia. Finalmente, o acelerador proposto possui uma frequência de operação de 800 MHz, sendo capaz de processar vídeos em resolução 4K com 60 quadros por segundo (qps) à um consumo médio de potência de 28,17 mW.

Palavras-chave: Codificação de vídeo digital, HEVC, Predição intra quadros, Arquitetura de hardware.

ABSTRACT

The increased use of streaming video applications, such as Youtube and Netflix, calls for the creation of new video compression techniques to enable network transmission. In addition, video compression is a must when it comes to storing such data, as uncompressed Ultra-HD resolution and larger video files can take up to tens of gigabytes per minute. For example, a 4K resolution video with 30 frames per second and 3 bytes to represent each pixel has, approximately, 44.8 gigabytes of data per minute. To increase the compression ratio, state-of-the-art standards encoders are 10 times more complex than those of previous standards. Also, in order to enable compression on battery-operated mobile devices, the most complex encoder steps must be performed on dedicated hardware which, in turn, must be designed to be energy efficient. This work presents the design, description and evaluation of a hardware accelerator block optimized to perform the intraframe prediction process for the High Efficiency Video Coding (HEVC) coding standard. In addition, low power design and synthesis techniques were used to reduce energy consumption. Finally, the proposed accelerator has an operating frequency of 800 MHz and is capable of processing 4K video at 60 frames per second (fps) at an average power consumption of 28.17 mW.

Keywords: Digital video coding, HEVC, Intra frame prediction, Hardware architecture.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

As últimas décadas têm sido marcadas pela crescente demanda por dispositivos móveis (e.g. *smartphones* e *notebooks*). Conti et al. (2018) constatou que em 2015 o número de usuários de *smartphones* e *tablets* correspondiam a cerca de 25% e 14% da população mundial, respectivamente. Estima-se que estes valores aumentem para 37% e 19% até 2020, totalizando um aumento aproximado de 1,2 milhão no número de usuários. Em suma, a popularização desses dispositivos deve-se à maior facilidade de acesso à internet através de tecnologias *wireless* e a grande variedade de aplicações disponíveis para esses dispositivos (Conti et al., 2018). Dentre as aplicações mais populares encontram-se aquelas que permitem a captura, armazenamento e transmissão via rede (*streaming*) de arquivos de vídeo. Muitas das aplicações que utilizam *streaming* de vídeo (e.g. *Youtube* e *Netflix*) também são amplamente utilizadas por dispositivos não portáteis (e.g. *desktops* e *smart TVs*). O uso em massa dessas aplicações ajuda a entender o fato de que, em 2016, aproximadamente 73% de todo o tráfego na internet estava relacionado a vídeos. Além disso, estima-se que em 2021 esse percentual esteja em torno de 83% (CISCO, 2017).

Além da ampla demanda por aplicações multimídia, existe uma crescente demanda por resoluções maiores (Ling, 2012). Como há uma maior quantidade de *pixels* nessas resoluções, faz-se necessário um volume de dados maior para representar vídeos em tais resoluções. Essa característica pode ser claramente observada ao comparar as resoluções *Super Video Graphics Array* (SVGA) (800×600 *pixels*), ainda suportada por grande parte dos televisores e monitores, e a resolução *Full* HD (1920×1080 *pixels*): um vídeo sem compressão de 30 minutos, amostragem de 30 qps, 3 *bytes* para representar cada *pixel* e resolução SVGA teria, aproximadamente, 77,8 *gigabytes* de dados; aumentando a resolução para *Full* HD e mantendo-se as demais configurações, o mesmo vídeo necessitaria quase 4,32 vezes mais dados.

Tal volume dificulta a manipulação de vídeos digitais sem compressão. Além disso, para aplicações de vídeo conferência, que permitem transmissões de vídeos ao vivo (e.g *Skype* e *Whatsapp*), seria necessário uma banda de 1,5 *gigabits* por segundo para transmitir vídeos em resolução *Full* HD.

Portando, a demanda por resoluções cada vez maiores faz com que aumente, também, a demanda por novas ferramentas de compressão. Assim, codificadores de vídeo podem atingir as taxas de compressão desejadas. Por outro lado, a introdução de novas ferramentas também aumenta a complexidade computacional dos codificadores. O atual padrão de codificação de vídeo digital HEVC, por exemplo, embora permita uma compressão até 50% maior que seu antecessor (*Advanced Video Coding* (AVC)) mantendo a mesma qualidade visual, teve também um aumento estimado de 2 a 10 vezes em sua complexidade (SULLIVAN et al., 2012). Ainda, a demanda por codificação em tempo real dificulta que esta possa ser realizada em *software* executando sobre um processador de propósito geral. Por conta disso, em geral, as etapas mais complexas da codificação são executadas em blocos aceleradores em *hardware*. Estes, por sua vez, precisam ser energeticamente eficientes, uma vez que o codificador pode ser embarcado

em dispositivos móveis operados à bateria (Herglotz; Springer; Kaup, 2014).

As etapas executadas na codificação visam reduzir a quantidade de dados necessária para representar vídeos e viabilizar sua manipulação. Assim, compressores de vídeos buscam explorar redundâncias existentes neste tipo de arquivo, tais como:

- **Espacial (intra quadro):** existente entre regiões semelhantes dentro de um mesmo quadro;
- **Temporal (inter quadros):** existente entre regiões semelhantes entre quadros próximos no tempo;
- **Entrópica:** relacionada com a representação binária da informação.

Dentre as etapas de compressão de um codificador, as predições intra (foco deste trabalho) e inter quadros são responsáveis por reduzir redundâncias espaciais e temporais, respectivamente. A predição intra quadros do padrão HEVC é subdividida em três principais etapas: pré-processamento das amostras de referência, predição de amostras e pós-processamento das amostras preditas. Ao longo do desenvolvimento do padrão HEVC, todas as etapas foram projetadas para permitir altas taxas de processamento minimizando os recursos computacionais necessários, tanto para codificar quanto decodificar um vídeo digital (Sze et al., 2014). Cada uma destas etapas será melhor detalhada no Capítulo 2.

1.1 OBJETIVOS

O principal objetivo deste trabalho consiste no projeto, descrição e avaliação de um bloco acelerador em *hardware* otimizado para realização do processo de predição intra quadros do padrão HEVC. O bloco acelerador foi implementado fazendo uso de técnicas de projeto e síntese de baixa potência visando atender às restrições de consumo energético, uma vez que pode ser embarcado em dispositivos móveis operados à bateria. Por se tratar da implementação de uma etapa do processo de codificação do padrão HEVC, o *hardware* foi desenvolvido para estar em conformidade com a especificação deste.

1.1.1 Objetivos Específicos

1. Projeto da arquitetura dos blocos de *hardware* responsáveis pela geração dos candidatos da predição intra quadros;
2. Aplicação de técnicas de projeto visando a redução do consumo de potência da arquitetura proposta;
3. Integração e validação da arquitetura utilizando dados reais extraídos do *software* de referência do padrão HEVC;

4. Análise crítica das estimativas de área, consumo de potência e quantidade de células utilizadas;

5. Comparação dos resultados obtidos com trabalhos correlatos.

1.2 ORGANIZAÇÃO DESTE TRABALHO

No Capítulo 2 será explicado o fluxo de codificação híbrida da qual o padrão HEVC se baseia. No capítulo 3 serão apresentados os trabalhos correlatos, trazendo as principais informações quanto aos objetivos e resultados de cada um. No Capítulo 4 serão apresentados os módulos que compõem o bloco acelerador em *hardware* proposto neste trabalho. No Capítulo 5 serão apresentados detalhes do método para avaliação da arquitetura proposta para predição intra quadros do padrão HEVC. Por fim, nos Capítulos 6 e 7 serão apresentados os resultados e conclusões deste trabalho, respectivamente.

2 CONCEITOS SOBRE CODIFICAÇÃO DE VÍDEO

Um padrão de codificação de vídeo, tal como o HEVC, define como um vídeo é representado em uma sequência de *bits* compacta (*bitstream* de saída), i.e., define muito mais a decodificação do que a codificação. No entanto, para que um codificador realize a compressão de um vídeo de modo que ele possa ser decodificado corretamente, o padrão necessita especificar algumas etapas da codificação. Assim, o codificador de vídeo deve gerar o *bitstream* conforme definido por um padrão e, dessa maneira, o vídeo poderá ser decodificado por qualquer outra ferramenta que segue o mesmo padrão. Com isso, os codificadores tem autonomia para selecionar quais ferramentas serão utilizadas para a compressão do vídeo, desde que, a sequência de *bits* resultante esteja de acordo com a especificação do padrão.

2.1 PARTICIONAMENTO DE BLOCOS

Durante a compressão, os quadros originais são particionados em blocos de tamanhos menores e, portanto, as etapas de compressão são realizadas sobre blocos e não sobre quadros. Trabalhar com blocos menores é uma característica incorporada por diversos padrões de codificação de vídeo desenvolvidos ao longo do tempo (*e.g.* AVC e *Google* VP9). O particionamento em blocos permite uma melhor exploração das redundâncias presentes no vídeo, bem como uma melhor relação entre taxa de compressão e qualidade visual (Sze et al., 2014).

O padrão HEVC possui uma estrutura de particionamento de blocos bastante flexível se comparado com seu antecessor (AVC). Nesse contexto, o HEVC introduz quatro conceitos que não existiam no AVC, sendo eles: *Coding Tree Unit* (CTU), *Coding Unit* (CU), *Prediction Unit* (PU) e *Transform Unit* (TU). A seguir, cada um dos quatro conceitos serão melhor descritos.

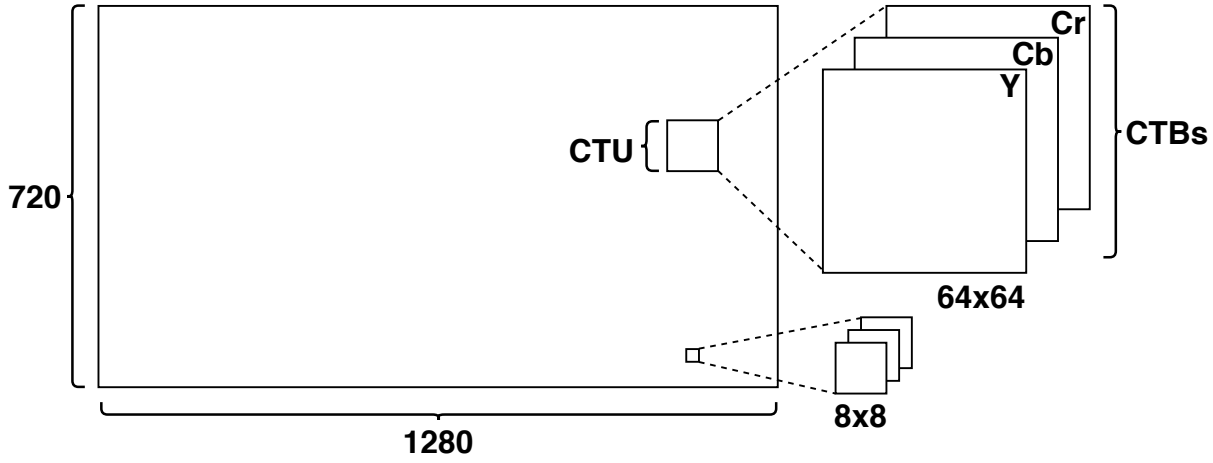
2.1.1 *Coding Tree Unit* (CTU)

Um quadro é particionado em diversos blocos quadrados chamados de CTUs. Uma CTU tem tamanho N×N, onde N $\in$ {8, 16, 32, 64}, porém, como pode sofrer diversos particionamentos, blocos de tamanho 4×4 podem ser gerados. Uma CTU é constituída por uma *Coding Tree Block* (CTB) (bloco com valores (amostras) de um canal de cor dos *pixels*) para o canal *luma* e duas CTBs para os canais *chroma* (Kim et al., 2012). Dado que a visão humana é mais sensível à luz do que à cor, os canais *chroma* geralmente são sub-amostrados, i.e., reduz-se a dimensão desses canais *chroma* sem muita perda de informação Lainema et al. (2012). A Figura 1 ilustra a divisão inicial de um quadro em *slices* e das *slices* em CTUs.

2.1.2 *Coding Unit* (CU)

Uma CTU pode ser particionada várias vezes em blocos quadrados, desde que o particionamento respeite um tamanho mínimo que, por sua vez, é parametrizável no HEVC (ge-

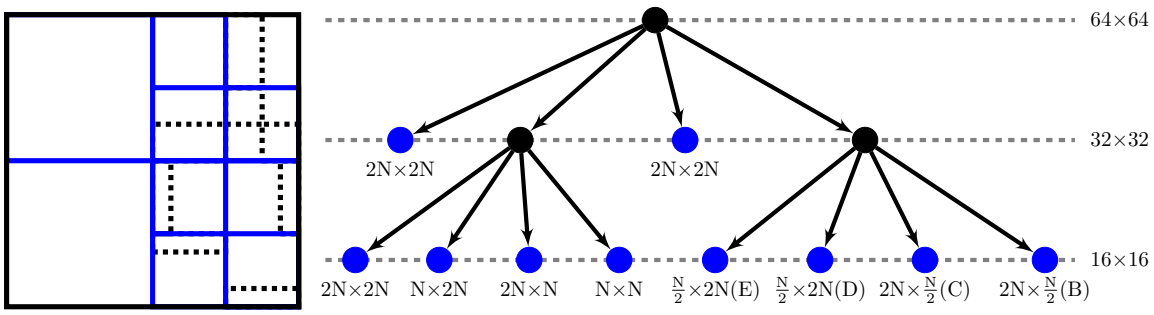
Figura 1 – Ilustração do particionamento inicial de um quadro em resolução 1280×720 pixels. "Y" refere-se ao canal de cor *luma* e "Cb" e "Cr" aos canais de cor *chroma*. Criada pelo autor.



Fonte: o autor.

ralmente 8×8). Para controlar as decisões de particionamento tomadas pelo HEVC, é utilizada uma árvore auxiliar chamada *quadtree*. A Figura 2 exemplifica uma possível forma de particionamento de uma CTU de tamanho 64×64.

Figura 2 – Ilustrações de uma CTU particionada à esquerda e da *quadtree* representando o particionamento dessa CTU à direita.



Fonte: adaptada de Seidel et al. (2019).

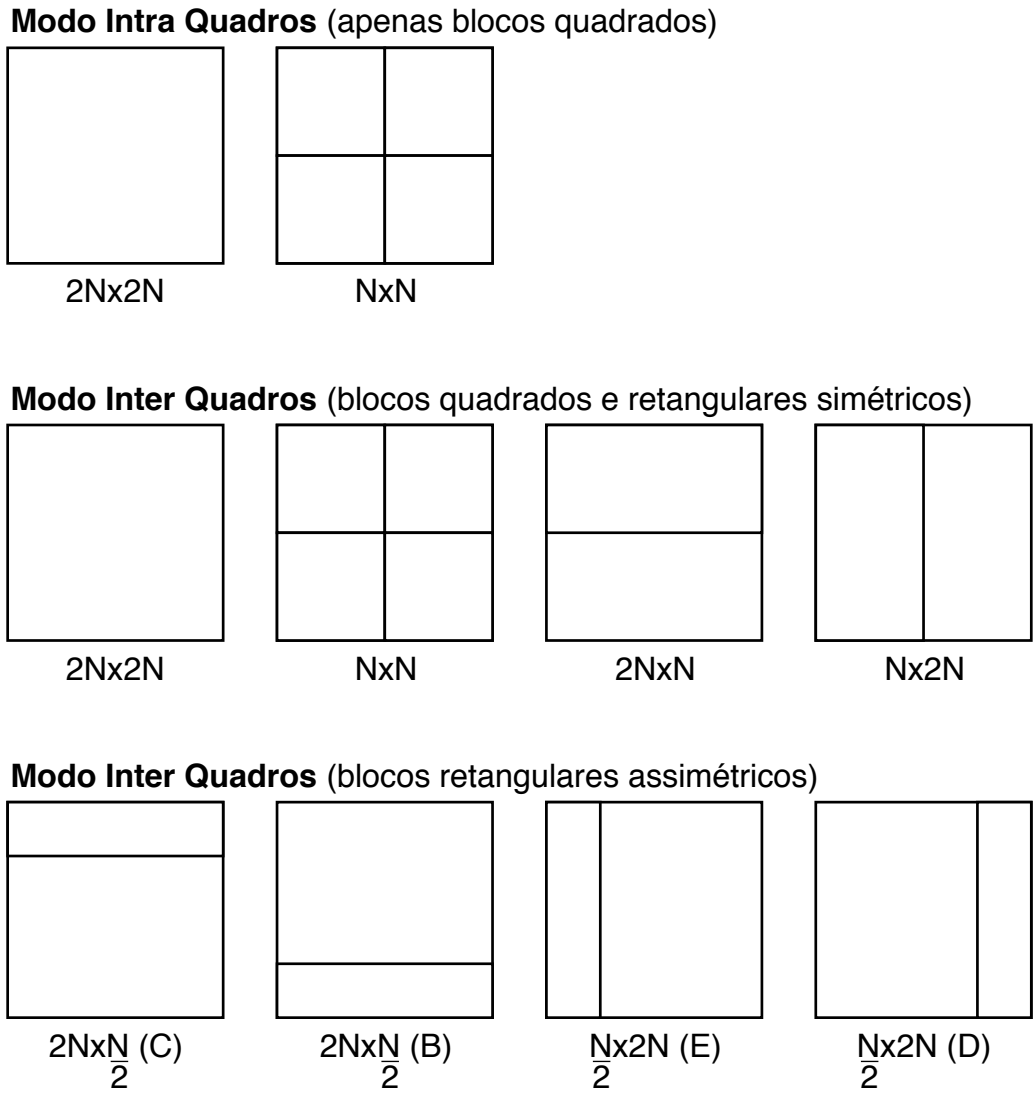
Na *quadtree* da Figura 2, o tamanho dos nós é mostrado à direita das linhas tracejadas. Consequentemente, todos os nós em um mesmo nível possuem o mesmo tamanho. As notações imediatamente abaixo das folhas não necessariamente correspondem ao tamanho do bloco, são notações utilizadas em outra etapa de particionamento (mais detalhes serão apresentados à seguir).

Todos os nós da *quadtree* são CUs resultantes do particionamento da CTU. De maneira similar a CTU, uma CU é composta por uma *Coding Block* (CB) *luma* e duas CBs *chroma* (Kim et al., 2012).

2.1.3 *Prediction Unit* (PU)

Diferente das CUs, o único particionamento de PUs ocorre quando uma CU folha é particionada em uma ou mais PUs. As notações das folhas da *quadtree* apresentada na Figura 2 indica a decisão de particionamento das CUs em PUs. No HEVC, a notação de particionamento de PUs em blocos de tamanho 2N×2N significa que a PU possui o mesmo tamanho da CU. Logo, uma PU de tamanho N×N possui um quarto do tamanho de uma CU e assim por diante. O particionamento de PUs pode ocorrer de duas ou oito maneiras, dependendo do modo de predição selecionado, tal como ilustra a Figura 3.

Figura 3 – Ilustração das formas de particionamento de uma CU em PUs de acordo com o modo de predição predefinido. Nas partições assimétricas, as letras entre parênteses indicam a posição da menor partição do bloco, i.e., Esquerda, Direita, Cima e Baixo.



Fonte: adaptada de Kim et al. (2012).

Também, uma PU é composta por *Prediction Blocks* (PB) assim como as CTUs e CUs.

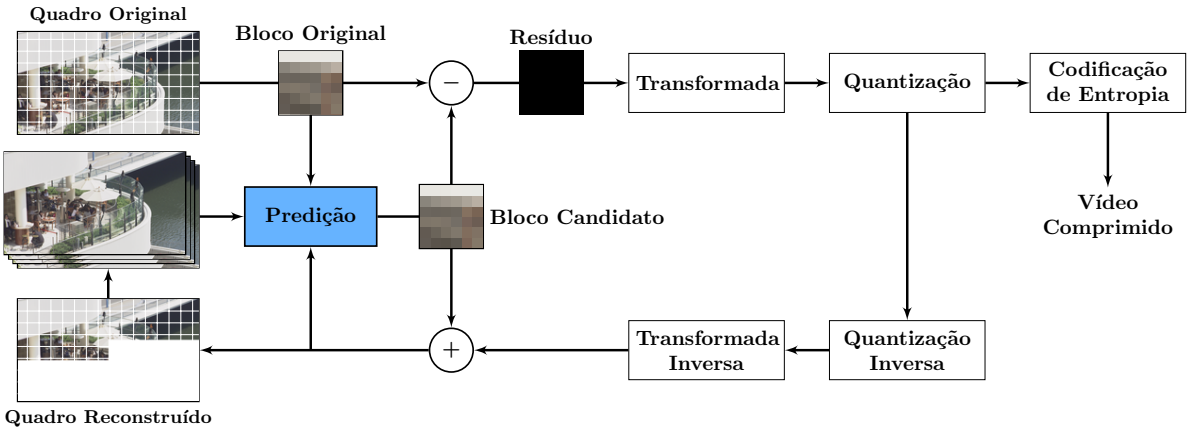
2.1.4 *Transform Unit* (TU)

TUs são compostas por *Transform Blocks* (TBs) e, assim como as PUs, são definidas à partir das CUs. Neste caso, TUs são blocos utilizadas para o cálculo das transformações e quantizações. As decisões de particionamento de CUs em TUs é controlada por uma outra estrutura, similar a *quadtree*, denominada *Residual QuadTree* (RQT).

2.2 FLUXO DE CODIFICAÇÃO HÍBRIDA

Um codificador de vídeo híbrido, tal como o HEVC, é um codificador que realiza compressão explorando redundâncias no vídeo que, por sua vez, é realizada através de etapas de predição. A Figura 4 ilustra uma estrutura simplificada de um codificador de vídeo híbrido. Nesta estrutura, como entrada estão os quadros originais (quadros do vídeo sem compressão) e como saída estão as sequências comprimidas de *bits*.

Figura 4 – Estrutura simplificada de um codificador baseado no modelo híbrido de codificação.



Fonte: adaptada de Seidel et al. (2019).

Como pode ser visto na Figura 4, cada bloco do quadro original, chamado de bloco original, passará por uma etapa de predição. A predição é responsável por construir blocos candidatos (utilizando algoritmos conhecidos pelo codificador) que serão utilizados para predizer o bloco original. Existem dois tipos de predições que podem ser utilizadas: inter quadros, que utiliza informações reconstruídas de quadros vizinhos já codificados (CORRêA, 2017), i.e., informações de quadros que passaram pela decodificação e, dessa forma, podem ser utilizado de amostra na codificação de outros quadros; ou intra quadros, quando são utilizadas informações reconstruídas do próprio quadro sendo codificado (CORRêA, 2017).

Após a etapa de predição, o codificador realizará o cálculo da diferença entre os blocos original e candidato à fim de obter o resíduo (valor de diferença entre os blocos). Posteriormente, o resíduo será transformado do domínio espacial para o de frequências. Nesta representação, é possível organizar os dados de forma que as baixas frequências (mais percep-

tíveis pelo sistema visual humano) sejam melhor separadas das altas frequências (CORRêA, 2017).

A etapa de quantização serve para reduzir a quantidade de informação necessária para representar o bloco. Teoricamente, todas a etapas de codificação vistas até agora são reversíveis sem perda de qualidade, o que não ocorre na quantização. Nesta etapa é utilizado o *Quantization Parameter* (QP), parâmetro este que determina a quantidade de informação de cada frequência que será mantida no bloco. Quanto maior o QP, maior a quantidade de informação descartada nessa etapa da compressão.

Antes de finalizar a codificação, a etapa de codificação de entropia é responsável por reduzir a redundância de representação dos dados, i.e., utiliza uma outra forma de codificação que minimiza a quantidade de *bits* para representar a mesma informação. A codificação entrópica não introduz perda de qualidade na imagem, pois é apenas uma maneira compacta de representar exatamente a mesma informação.

Por fim, após a quantização, o bloco codificado passa por etapas de quantização e transformada inversas e, somado ao bloco candidato inicial, o bloco codificado pode ser reconstruído e utilizado como referência para predição de outros blocos.

Como este trabalho é focado na predição intra quadros do padrão HEVC, a seção seguinte trás um maior detalhamento de como o padrão define esta etapa da codificação.

2.3 A PREDIÇÃO INTRA QUADROS DO PADRÃO HEVC

Após as etapas de particionamento, os blocos submetidos a predição intra quadros terão tamanho 4×4, 8×8, 16×16 ou 32×32. No padrão de codificação HEVC, a predição intra quadros é realizada em quatro principais etapas (ilustradas na Figura 5):

1. **Pré-processamento de amostras de referência:** onde as amostras de referência são preparadas para a realização da predição;

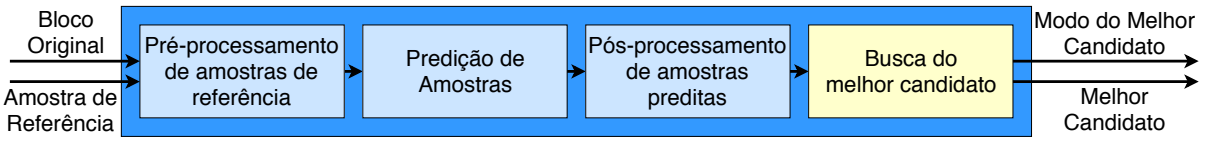
2. **Predição de amostras:** onde blocos candidatos serão gerados para predizer o bloco original;

3. **Pós-processamento de amostras preditas:** onde alguns dos blocos candidatos serão filtrados para gerar imagens mais suaves, i.e, sem descontinuidades geradas pela etapa de predição;

4. **Busca do melhor candidato:** onde será realizada a seleção do bloco candidato mais parecido com o bloco original.

A seguir, cada umas das etapas do processo de predição intra quadros será melhor detalhada.

Figura 5 – Etapas da predição intra quadros do HEVC.

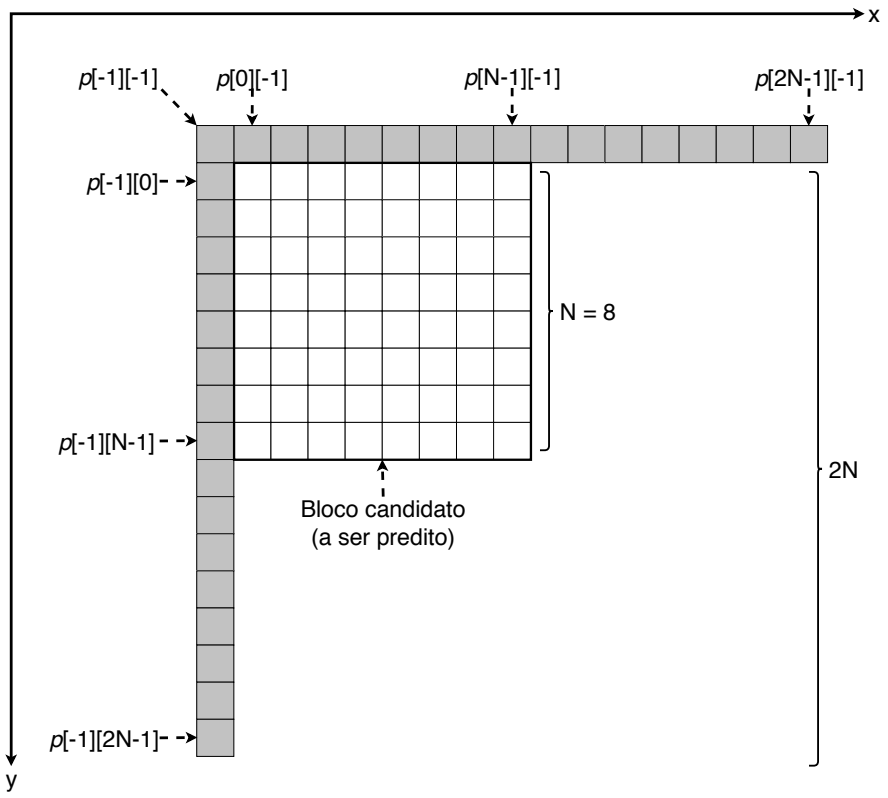


Fonte: o autor.

2.3.1 Pré-processamento de amostras de referência

A predição intra quadros de um bloco original de tamanho N×N é realizada utilizando-se amostras de referência de fora do bloco. A Figura 6 ilustra o número máximo de amostras de referência que podem ser utilizadas para a construção de um bloco candidato de tamanho N=8 (4N+1).

Figura 6 – Cada caixa em cinza é uma amostra de referência p[x][y] utilizada durante a predição para construir o bloco candidato que, neste caso, tem tamanho N=8.



Fonte: adaptada de Sze et al. (2014).

Note que, nem sempre todas as amostras de referência estarão disponíveis. Isso acontece, por exemplo, sempre que for realizada a predição do primeiro bloco de cada quadro original. Para esses casos particulares, o padrão define duas formas para preencher as amostras indisponíveis. Uma delas é utilizada quando nenhuma amostra estiver disponível. Neste caso, todas as amostras são preenchidas com um valor nominal médio definido de acordo com o

número de *bits* (n) de cada amostra, tal como mostra a Equação (1).

$$\text{valor nominal} = 2^{n-1} \tag{1}$$

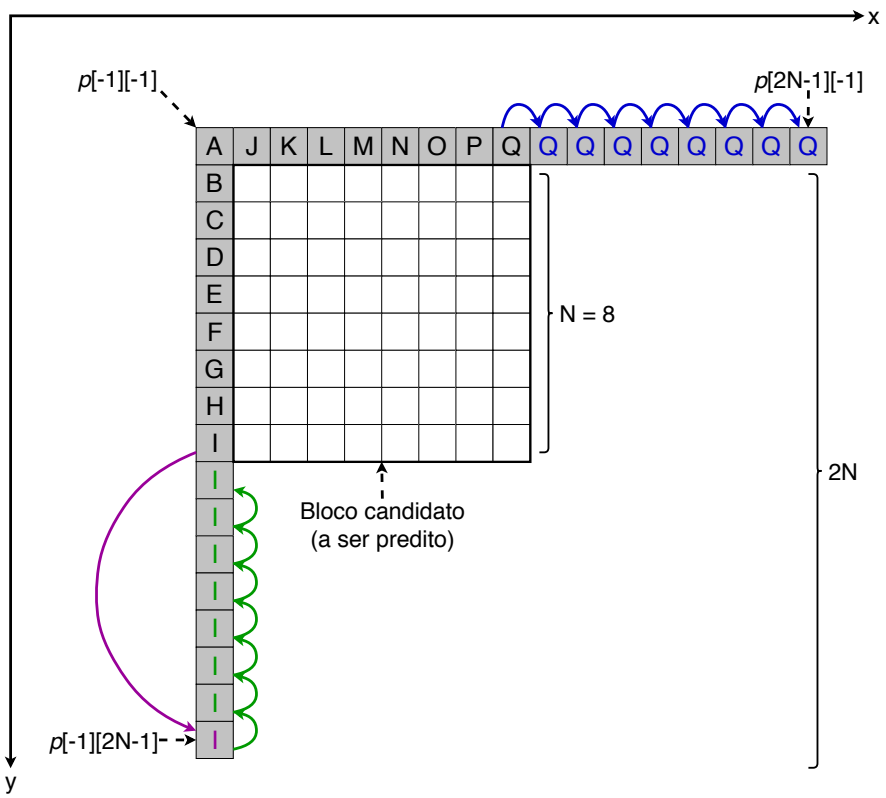
A segunda forma de preencher as amostras indisponíveis necessita que ao menos uma amostra possua valor disponível. Neste caso, como ilustra a Figura 7, as amostras indisponíveis são preenchidas utilizando o seguinte algoritmo de substituição:

1. Se a amostra $p[-1][2N-1]$ estiver indisponível, então ela será preenchida com o valor da primeira amostra disponível partindo da posição $p[-1][2N-2]$ até $p[-1][-1]$ e, posteriormente, até $p[2N-1][-1]$.

2. Com y de $2N-1$ à -1, se a amostra $p[-1][y]$ estiver indisponível, então será preenchida com o valor da amostra imediatamente abaixo, i.e., com o valor da amostra $p[-1][y+1]$.

3. Com x de 0 à $2N-1$, se a amostra $p[x][-1]$ estiver indisponível, então será preenchida com o valor da amostra imediatamente à esquerda, i.e., com o valor da amostra $p[x-1][-1]$.

Figura 7 – Os valores em preto representam amostras válidas inicialmente. Em rosa, verde e azul é possível visualizar o resultado da execução dos passos 1, 2 e 3, respectivamente, do algoritmo de substituição de amostras indisponíveis.



Fonte: adaptada de Sze et al. (2014).

Após tornar disponíveis todas as amostras de referência, o padrão define uma etapa opcional, onde filtros podem ser utilizados para suavizar a transição entre blocos vizinhos, i.e., evitar bordas indesejadas. Optado pela realização de filtros, sua aplicação leva em consideração o tamanho do bloco e o modo de predição (35 modos no total, sendo eles enumerados de 0 à 34 e melhor descritos na Subseção 2.3.2). A Tabela 1 resume as situações em que o padrão define filtros que podem ser aplicados às amostras de referência.

Tabela 1 – Tamanhos de bloco e modos de predição onde pode ser aplicado filtro sobre as amostras de referência. Adaptado de Corrêa (2017).

Tamanho do Bloco	**Modos Com Pré-filtragem**
4×4	Nenhum
8×8	2, 18 e 34
16×16	Todos exceto 1, 9, 10, 11, 25, 26 e 27
32×32	Todos exceto 1, 10 e 26

Na coluna esquerda da Tabela 1 encontram-se todos os possíveis tamanhos de bloco candidatos que podem ser gerados na predição intra quadros do HEVC. À direita, encontram-se os respectivos modos de predição, condição em que é realizada a aplicação de filtros nas amostras. Dessa forma, para os modos que não constam na coluna direita da Tabela 1 as amostras de referência não passam por essa etapa de filtro.

Dois tipos de filtros podem ser aplicados: interpolação linear ou filtro 3 *taps* ([x_0 + 2*x_1 + x_2] / 4). A escolha do tipo de filtro utilizado é dependente do tamanho do bloco e da continuidade dos valores das amostras de referência.

Caso o tamanho do bloco seja 32 × 32 e as Condições (C1) e (C2) forem satisfeitas (onde b o número de *bits* por amostra), então as amostras são consideradas descontinuas. Neste caso, será utilizado o filtro de interpolação linear que, por sua vez, é definido através das Equações (2) e (3), sendo: "<<" e ">>" operações aritméticas de deslocamento à esquerda e a direita, respectivamente, e $0 \leq x, y \leq 2N-2$.

$$\left|p[-1][-1]+p[2N-1][-1]-2p[N-1][-1]\right| < \left(1 << (b-5)\right) \quad \text{(C1)}$$

$$\left|p[-1][-1]+p[-1][2N-1]-2p[-1][N-1]\right| < \left(1 << (b-5)\right) \quad \text{(C2)}$$

$$p[-1][y] = \left((2N-1-y)*p[-1][-1]+(y+1)*p[-1][2N-1]+N\right) >> 6 \quad (2)$$

$$p[x][-1] = \left((2N-1-x)*p[-1][-1]+(x+1)*p[2N-1][-1]+N\right) >> 6 \quad (3)$$

Para os demais casos é utilizado o filtro 3 *taps* ([x_0 + 2*x_1 + x_2] >> 2), assim como mostram as Equações (4) - (6), sendo: $0 \leq x, y \leq 2N-2$.

$$p[-1][-1] = (p[-1][0]+2p[-1][-1]+p[0][-1]+2) >> 2 \quad (4)$$

$$p[-1][y] = (p[-1][y+1] + 2p[-1][y] + p[-1][y-1] + 2) >> 2 \quad (5)$$

$$p[x][-1] = (p[x+1][-1] + 2p[x][-1] + p[x-1][-1] + 2) >> 2 \quad (6)$$

2.3.2 Predição de amostras

O processo de predição de um bloco pode ser realizado através de 35 modos distintos, onde cada um permite gerar candidatos com conteúdos diferenciados, tipicamente presentes em imagens (Sze et al., 2014). O modo 0, denominado planar, é utilizado para predizer blocos com conteúdo contínuo mas com suaves transições. Já o modo 1, denominado DC, permite predizer blocos com conteúdo neutro, i.e., com nenhuma variação. Os demais modos, denominados angulares, foram elaborados para predizer blocos com diferentes características direcionais e, por conta disso, os modos 2 a 17 são também chamados de horizontais e os restantes de verticais. A Figura 8 ilustra a geração dos 35 blocos candidatos de acordo com os modos de predição intra para um bloco original de tamanho 32×32.

2.3.2.1 Predição Planar

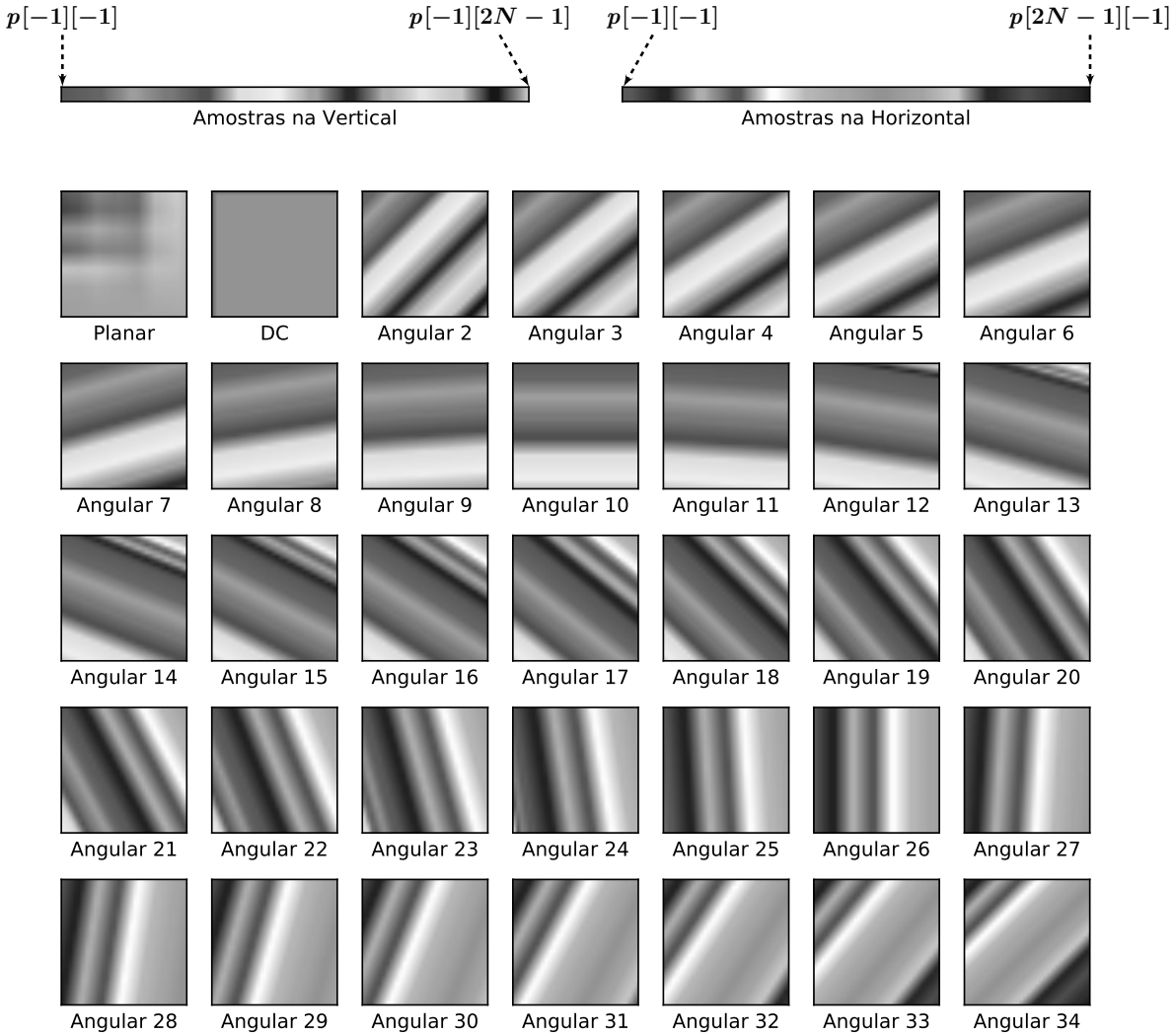
A predição através do modo planar foi projetada com a capacidade de construir uma imagem cujo conteúdo é uma superfície suave e sem descontinuidades. Para isso, são construídos dois blocos intermediários, um para capturar o conteúdo das amostras verticais (*ph*) e outro para capturar o conteúdo das amostras horizontais (*pv*). Assim como ilustra a Figura 9, os blocos *ph* e *pv* são construídos através de uma predição linear definidas nas Equações (7) e (8), respectivamente. O bloco final, por sua vez, é obtido através da média das predições lineares, bem como mostra a Equação (9), sendo: $0 \leq x, y \leq \mathrm{N} - 1$ para todos os casos.

$$ph[x][y] = (N-1-x) * p[-1][y] + (x+1) * p[N][-1] \quad (7)$$

$$pv[x][y] = (N-1-y) * p[x][-1] + (y+1) * p[-1][N] \quad (8)$$

$$p[x][y] = (ph[x][y] + pv[x][y] + N) >> (\log_2(N) + 1) \quad (9)$$

Figura 8 – Ilustração dos 35 possíveis blocos candidatos gerados a partir dos modos de predição intra para um bloco original de tamanho N=32. No topo estão as amostras de referência verticais e horizontais utilizadas para predizer os candidatos.



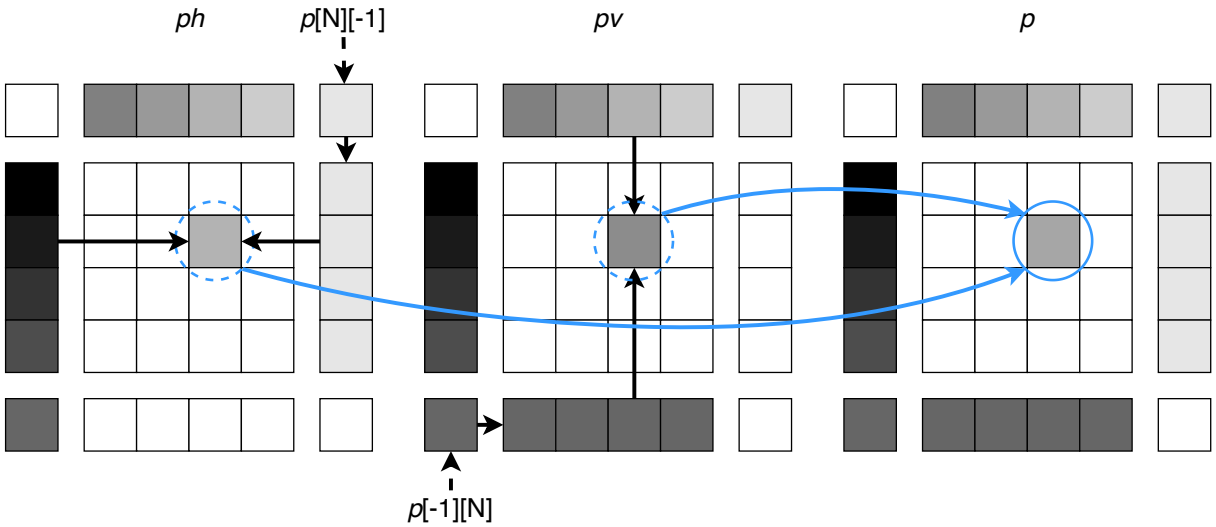
Fonte: o autor.

2.3.2.2 Predição DC

O modo DC é utilizado para gerar imagens que possuem conteúdos neutros e quase sem variação. Assim sendo, o bloco candidato é construído simplesmente fazendo a média dos valores das amostras de referência (chamada de *dc_val*), bem como mostra a Equação (10).

$$dc\_val = \left(N + \sum_{i=0}^{N-1} (p[i][-1] + p[-1][i])\right) >> \left(\log_2(N) + 1\right) \tag{10}$$

Figura 9 – Ilustração do processo de predição de um canal de cor de um bloco intra através do modo planar. As matrizes à esquerda e central ilustram as predições lineares para geração dos blocos horizontal e vertical. A matriz à direita mostra o resultado da média das duas outras matrizes.



Fonte: adaptada de Sze et al. (2014).

2.3.2.3 Predição Angular

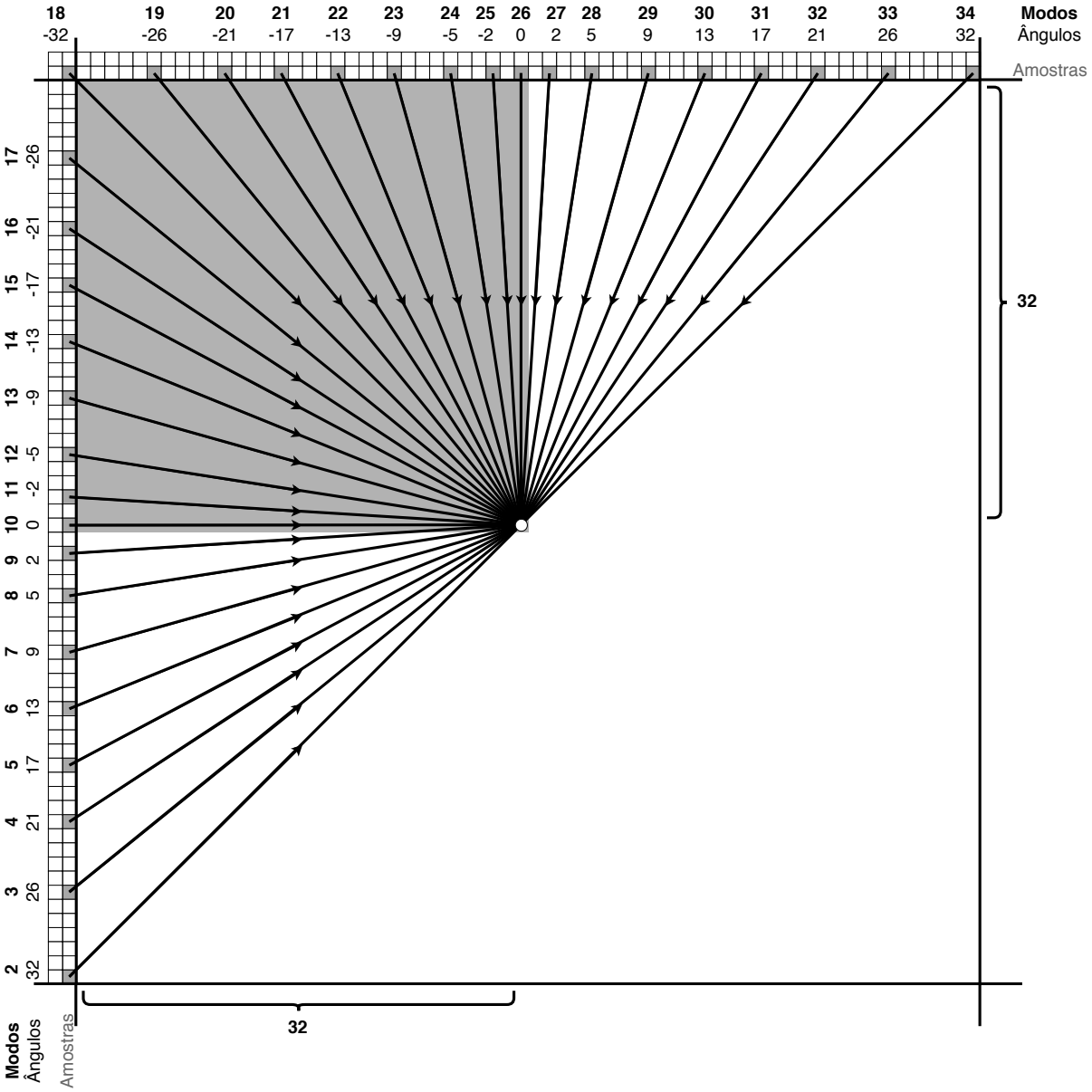
A predição angular permite gerar blocos com diferentes características angulares, bem como pode ser visto na Figura 8. O HEVC define 33 modos angulares para a predição intra quadros, onde cada modo possui um ângulo (A) que define a direção da predição. Assim como pode ser visto na Figura 10, a predição para cada um dos modos vai gerar um bloco candidato com diferentes imagens, de acordo com as amostras utilizadas na predição.

A predição angular define uma direção de predição através dos ângulos e preenche o bloco candidato projetando as amostras de referência nessa direção. Portanto, o ângulo define o deslocamento de uma amostra entre uma coluna e outra para os modos horizontais e, no caso dos verticais, o deslocamento entre uma linha e outra. A Figura 11 ilustra a direção de predição definida pelo ângulo do modo 18.

Dada a direção de predição definida pelo ângulo, os modos de 2 à 17 são chamados de horizontais e os demais de verticais. Ainda, por estarem alinhados aos eixos x e y, os modos 10 e 26 são chamados de diretamente horizontal e vertical, respectivamente.

Segundo Sze et al. (2014), imagens construídas a partir dos modos 10 e 26 ocorrem com mais frequência do que as construídas através dos demais ângulos. Aproveitando-se desse fenômeno, o padrão define mais modos com ângulos próximos aos dos modos 10 e 26. Logo, a medida que o ângulo se distancia dos modos 10 e 26, menor a quantidade de modos angulares para capturar esse comportamento. Essa decisão sobre a quantidade de modos e as características angulares dos mesmos possibilita melhor *trade-off* entre complexidade e eficiência de compressão (Lainema et al., 2012).

Figura 10 – Ilustração da direção de predição de um bloco de tamanho N=32 para todos os modos angulares (números em negrito) com os respectivos ângulos (valor imediatamente abaixo ao modo).

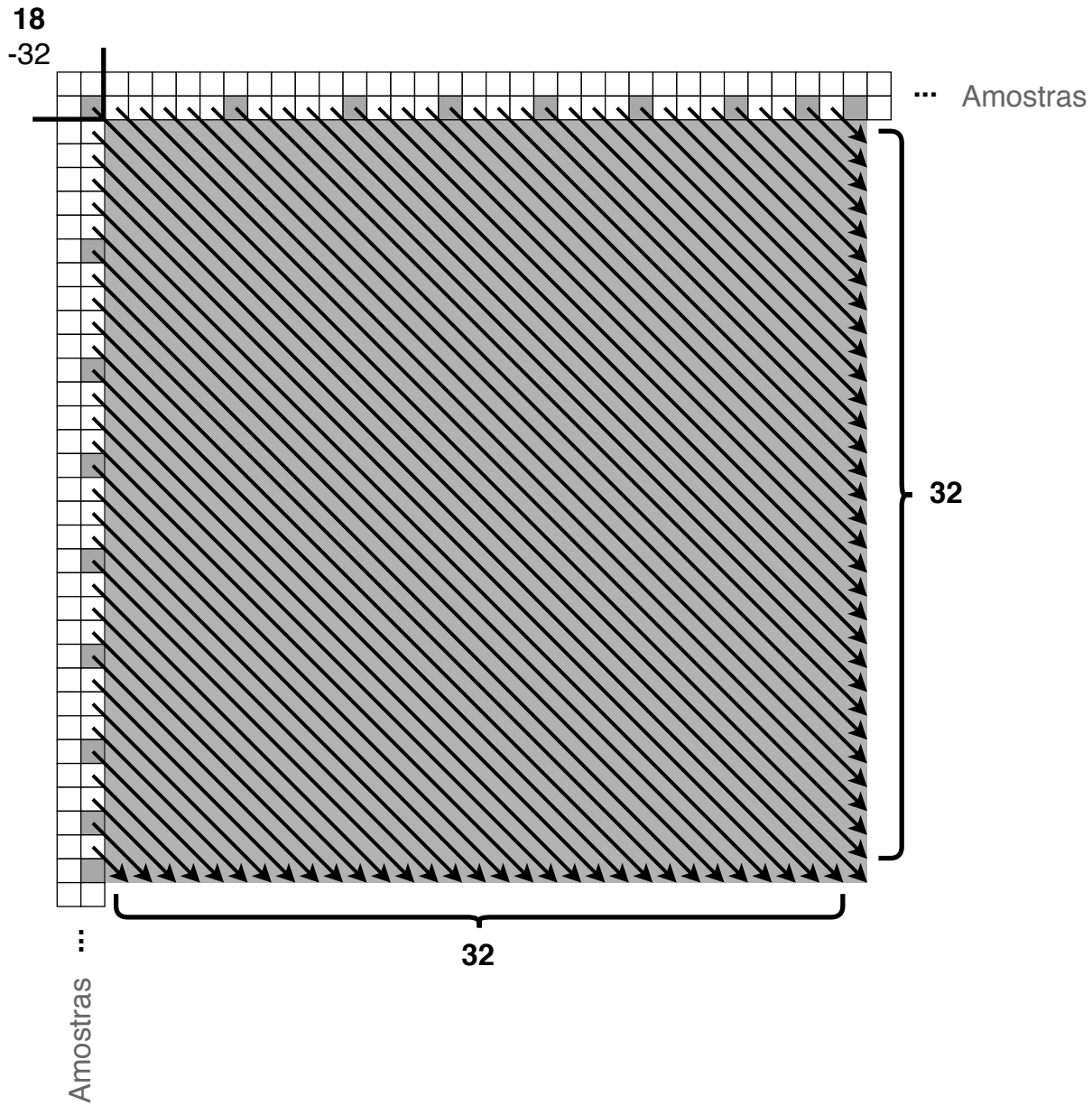


Fonte: adaptada de Sze et al. (2014).

A primeira diferença entre o processo de predição dos modos angulares para os demais está no fato de que as amostras não são utilizadas diretamente, mas sim de um vetor unidimensional construído à partir delas. A construção do vetor de amostras possui uma etapa quando o ângulo do modo é positivo e, caso contrário, possui duas etapas. Para modos horizontais com ângulo positivo, a construção do vetor de amostras é realizado simplesmente copiando as amostras de referência na ordem em que elas estão, assim como mostra a Equação (11), sendo: $y \geq 0$. A construção do vetor para os modos verticais com ângulos positivos é análogo.

$$ref[y] = p[-1][y-1] \tag{11}$$

Figura 11 – Ilustração da direção de predição de um bloco de tamanho N=32 para o modo angular 18.



Fonte: o autor.

Quando a predição for realizada com modos horizontais cujo ângulo seja negativo, a construção do vetor de amostras, além das posições positivas, também é construído em posições negativas. Assim como mostra a Equação (12) (onde $y < 0$), a construção do vetor em posições negativas utiliza o inverso do ângulo (B) do modo correspondente (a Tabela 2 mostra o inverso de todos os ângulos negativos). Para os modos verticais com ângulo negativo a construção do vetor em posições negativas é análoga.

$$ref[y] = p[\Big((y * B + 128) >> 8\Big) - 1][-1] \tag{12}$$

Tabela 2 – Ângulos negativos dos modos e os respectivos ângulos inversos.

A	-32	-26	-21	-17	-13	-9	-5	-2
B	-256	-315	-390	-482	-630	-910	-1638	-4096

Para realizar a construção do bloco candidato, primeiramente é necessário determinar o deslocamento das amostras no sentido indicado pelo ângulo do modo de predição. O padrão separa o deslocamento em uma parte inteira (i) e outra fracionária (f), ambas podem ser calculadas através das Equações (13) e (14) para os modos horizontais e (15) e (16) para os modos verticais, sendo: "&" a operação lógica "AND" *bit* a *bit*.

$$i = \Big((x+1)*A\Big) >> 5 \tag{13}$$

$$f = \Big((x+1)*A\Big)\&31 \tag{14}$$

$$i = \Big((y+1)*A\Big) >> 5 \tag{15}$$

$$f = \Big((y+1)*A\Big)\&31 \tag{16}$$

Finalmente, a predição é realizada projetando as amostras do vetor para o bloco candidato no sentido indicado pelo ângulo do modo. As Equações (17) e (18) (onde $0 \leq x, y < \mathrm{N}$) mostram como é realizada a construção do bloco candidato para os modos horizontais e verticais, respectivamente. Note que: i é utilizado para indexar o vetor de referências na posição inteira e f é utilizado para indicar a influência que a amostra seguinte possui sobre a predição do índice atual do bloco candidato.

$$p[x][y] = \Big((32-f)*ref[x+i+1]+f*ref[x+i+2]+16\Big) >> 5 \tag{17}$$

$$p[x][y] = \Big((32-f)*ref[y+i+1]+f*ref[y+i+2]+16\Big) >> 5 \tag{18}$$

2.3.3 Pós-processamento de amostras preditas

Mesmo realizando pré-filtragem das amostras no início da predição, algumas descontinuidades nas bordas dos blocos candidatos podem ser geradas dependendo do modo de predição. Esses casos são mais visíveis nas bordas superior e esquerda do bloco predito com o modo DC, uma vez que todo o bloco é preenchido com um valor neutro. Descontinuidades também

são comuns nas bordas esquerda e superior dos blocos preditos com os modos diretamente vertical e horizontal, respectivamente.

Visando suavizar essas descontinuidades, o padrão define filtros opcionais que podem ser aplicados aos blocos candidatos após a sua construção. Caso optado pelo uso dos filtros pós-predição, sua aplicação será decidida com base no modo de predição e no tamanho do bloco. Os filtros são aplicados apenas para o modo de predição DC (em todos os tamanhos de bloco, exceto 32×32) ou para os modos diretamente horizontal e vertical (em todos os tamanhos de bloco).

Para o modo DC, um filtro 3 *taps* será aplicado à posição $p[0][0]$ do bloco candidato, tal como mostra a Equação (19). Ainda, cada uma das posições das bordas do bloco serão atualizadas com a média ponderada entre o valor da amostra correspondente e o *dc_val*, tal como mostram as Equações (20) e (21), sendo: $0 < x, y <$ N.

$$p[0][0] = (p[-1][0] + 2 * dc\_val + p[0][-1] + 2) >> 2 \tag{19}$$

$$p[x][0] = (p[x][-1] + 3 * dc\_val + 2) >> 2 \tag{20}$$

$$p[0][y] = (p[-1][y] + 3 * dc\_val + 2) >> 2 \tag{21}$$

Para o modo diretamente horizontal, o valor das posições da borda superior do bloco serão modificados através da diferença de outras duas amostras, bem como mostra a Equação (22), sendo: $0 \leq x <$ N. De forma análoga, para o modo diretamente vertical, o valor das posições da borda esquerda do bloco serão modificados de acordo com a Equação (23), sendo: $0 \leq y <$ N..

$$p[x][0] = p[x][0] + \Big((p[x][-1] - p[-1][-1]) >> 1\Big) \tag{22}$$

$$p[0][y] = p[0][y] + \Big((p[-1][y] - p[-1][-1]) >> 1\Big) \tag{23}$$

2.3.4 Busca do melhor candidato

Como dito anteriormente, na etapa de predição intra quadros são gerados 35 blocos candidatos, cada um com características distintas. O melhor candidato, será aquele que for mais parecido com o bloco original, i.e., aquele resultar no menor resíduo quando comparado com o bloco original. Dentre todas as informações de controle tomadas pelo codificador, também são salvos no arquivo de vídeo comprimido os resíduos entre os blocos originais e candidatos e o

modo como estes foram gerados. Assim, o decodificador pode gerar o bloco candidato através do modo especificado e, somando o resíduo, reconstruir o bloco original. Portanto, quanto mais parecido o bloco candidato em relação ao original, i.e., quanto menor for o resíduo, menor será o *bitstream* de saída do codificador.

Uma das maneiras de avaliar qual o melhor bloco candidato, levando em consideração a taxa de *bits* e distorção da imagem, é utilizando o algoritmo *Rate Distortion Optimization* (RDO). O RDO consiste em realizar todas as etapas de codificação (incluindo a codificação de entropia) para um modo de predição e então avaliar o custo em taxa-distorção desse modo (Sze et al., 2014). Portanto, o melhor modo será aquele que resultar no menor custo. A Equação (24) resume o cálculo do custo (J) para um determinado modo de predição, levando em consideração a distorção (D), o multiplicador de Lagrange (λ) e a taxa de *bits* (R) do modo.

$$J = D + \lambda * R \tag{24}$$

Na Equação (24) é possível notar que, quanto maior o valor do multiplicador de Lagrange, maior será o impacto, no custo, em otimizações na taxa de *bits*. De forma semelhante, quanto menor o valor do multiplicador de Lagrange, maior será o impacto, no custo, em otimizações na distorção. Por conta disso, o multiplicador de Lagrange é responsável por priorizar uma variável em detrimento de outra (CORRêA, 2017). A definição do valor de λ depende da métrica utilizada para cálculo da distorção e da taxa de *bits* que pretende-se atingir. Trabalhos da literatura utilizam diversas abordagens e experimentações na tentativa de definir o melhor valor de λ para um dado cenário (Wiegand; Girod, 2001) ou valor adaptativo para diversos cenários (Yang; Wan; Izquierdo, 2007).

Dada a grande variedade de tamanhos de bloco e modos de predição intra quadros, a utilização do algoritmo RDO torna-se agravante no HEVC. Ainda, para as aplicações que demandam codificação em tempo real, esse tipo de avaliação é inviável. Por conta disso, até mesmo o HEVC busca medidas alternativas para reduzir a complexidade das avaliações através do algoritmo RDO. Uma delas, adotada no *software* de referência do HEVC, chamado HEVC *Model* (HM), consiste em selecionar apenas os modos mais prováveis de resultar no menor custo, os quais são chamados de *Most Probable Modes* (MPM). Os critérios de seleção dos MPM levam em consideração o modo do melhor candidato de PUs vizinhas. Dessa forma, apenas os MPM são submetidos à avaliação do algoritmo RDO (Sze et al., 2014).

Outra alternativa para reduzir a complexidade de avaliação dos modos, adotada por Corrêa et al. (2017) por exemplo, consiste em estimar o custo dos modos utilizando métricas de distorção (e.g. *Sum of Absolute Differences* (SAD) e *Sum of Absolute Transformed Differences* (SATD)). Porém, a confiabilidade dessa abordagem é baixa quando os custos estimados entre os modos são próximos. Por conta disso, alguns trabalhos, como o proposto por Kim, Shih e Kuo (2006), utilizam as métricas de distorção apenas nos casos onde o menor custo estimado é relativamente distante do custo dos demais modos. Ainda, caso não seja possível selecionar o melhor candidato apenas estimando o custo, então é recorrido à utilização de um algoritmo de

RDO simplificado.

3 REVISÃO BIBLIOGRÁFICA

Neste capítulo são apresentados trabalhos que propõem implementações de blocos aceleradores em *hardware* dedicados à predição intra quadros do HEVC e tiveram como base as últimas versões do padrão. A seleção dos trabalhos foi realizada à fim de manter compatibilidade com o *draft* final do HEVC, visto que, inicialmente nem todas as etapas vistas no Capítulo 2 haviam sido incorporadas ao padrão. Assim, os trabalhos apresentados neste capítulo foram desenvolvidos quando todas as etapas da predição intra vistas anteriormente já estavam presentes no HEVC. Ainda, todos os trabalhos a seguir realizam a predição intra para todos os tamanhos de blocos e modos previstos pelo padrão.

3.1 ZHOU ET AL. (2013)

O trabalho proposto por Zhou et al. (2013) consiste na implementação de uma arquitetura de *hardware* altamente paralelizada para realização da predição intra quadros do HEVC. O que permite a alta paralelização da predição neste trabalho consiste na manipulação das equações que definem as predições. Os autores chegam a conclusão de que todas as predições podem ser reduzidas à uma única equação e, neste caso, foi possível a construção de módulos operacionais genéricos para predição. Com isso, os autores apresentam uma arquitetura com 64 unidades paralelas de elementos de processamento, i.e., unidades de processamento capazes de realizar a predição para qualquer um dos 35 modos possíveis.

Embora a arquitetura proposta por Zhou et al. (2013) possua uma forma de decisão de modo interna utilizando a métrica de distorção SAD, não há resultados do impacto dessa forma de decisão em termos de eficiência de codificação. O trabalho não contém informações quanto à simulação da arquitetura proposta, poucos resultados de síntese são apresentados e não há informações sobre o método de síntese utilizado. Ainda, a arquitetura proposta não implementa as etapas de pré e pós-processamento, mesmo elas sendo previstas no *draft* do padrão utilizado no desenvolvimento do trabalho de Zhou et al. (2013).

3.2 CONG LIU ET AL. (2013)

O trabalho proposto por Liu et al. (2013) é uma abordagem visando reduzir a dependência de dados causada em função das amostras de referência serem obtidas de blocos já codificados. Para isso, a arquitetura explora o uso de paralelismo utilizando um *pipeline* de três estágios. No estágio A, como chamam os autores, são realizadas todas as etapas de predição, tal como definidas pelo HEVC. Também em A é realizada a seleção do melhor candidato utilizando a métrica de distorção SATD com a *Hadamard Transform* (HT). Já no estágio B, o modo do melhor candidato é utilizado para predizer novamente o bloco original (apenas o modo selecionado) e calcular o resíduo final. Por fim, no terceiro estágio é realizada a reconstrução do bloco predito, como apresentado na Seção 2.2. Dessa maneira, a arquitetura implementa seu

próprio fluxo de reconstrução dos blocos preditos e, portanto, é possível acelerar o processo de codificação.

Os autores afirmam que a forma de decisão de modo utilizada no trabalho resultou, em média, um impacto de 10% no *Bjøntegaard Delta Rate* (BD-Rate), i.e., a abordagem resultou na redução da eficiência de codificação. Porém, assim como o trabalho anterior, este não contém detalhes do método de síntese e simulação da arquitetura proposta.

3.3 ABRAMOWSKI; PASTUSZAK (2014)

Assim como no trabalho anterior, a arquitetura de *hardware* proposta por Abramowski e Pastuszak (2014) tem por objetivo reduzir a dependência de dados. Neste caso, o fluxo de predição é dividido de acordo com o tamanho do bloco a ser predito.

Através de experimentos com o HM, verificou-se que em pouco mais de 50% dos casos o *software* de referência decide por particionar um quadro em blocos de tamanho 4×4. Logo, existem mais blocos de tamanho 4×4 que dos demais tamanhos e, por conta disso, a arquitetura proposta possui um fluxo de codificação responsável apenas por predizer este tamanho de blocos. Além disso, existe outro fluxo responsável por realizar a predição para os demais tamanhos de bloco. Porém, o trabalho de Abramowski e Pastuszak (2014) descreve apenas os modos de predição e não apresenta detalhes da arquitetura, como o método de busca do melhor candidato, por exemplo.

A arquitetura proposta por Abramowski e Pastuszak (2014) foi sintetizada utilizando a ferramenta *Synopsys®Design Compiler®* (DC®) e simulada através do *software Active*-HDL com dados obtidos do HM versão 9.2. Assim como nos trabalhos anteriores o método de síntese foi parcialmente informado.

3.4 CORRÊA (2017)

O trabalho de Corrêa (2017) tem como principal foco o desenvolvimento arquitetural de diversas abordagens para predição intra quadros. A justificativa do autor para implementação de diferentes abordagens consiste na heterogeneidade de aplicações e dispositivos que podem fazer uso de um codificador de vídeo. Assim, o trabalho de Corrêa (2017) busca apresentar três soluções arquiteturais que, por sua vez, podem ser utilizadas de acordo com a demanda.

Todas as propostas de Corrêa (2017) implementam as etapas de pré- e pós-processamento. Porém, como apenas uma delas (chamada de solução 5.4 pelo autor) realiza a predição para todos os tamanhos de bloco e modos previstos pelo padrão, apenas ela será utilizada para comparações neste trabalho. No entanto, as demais soluções realizam a predição apenas para blocos de tamanho 4×4, onde uma faz a predição para todos os modos previstos pelo padrão e a outra apenas para os modos 0, 1, 10 e 26.

A solução 5.4 proposta por Corrêa (2017) faz uso de paralelismo para atingir alta taxa de processamento, necessário para resoluções UHD. Ainda, a etapa de busca é realizada através

da seleção de oito melhores candidatos avaliados com a métrica de distorção SAD. Portanto, a arquitetura tem como saída o modo dos oito melhores candidatos gerados. Assim como definido pelo HEVC, os melhores candidatos poderão ser submetidos ao processo *Full* RDO (execução de todo o processo de codificação para posteriormente decidir, de fato, qual o melhor candidato).

Segundo experimentos realizados pelo autor a forma de decisão de modo utilizada teve um impacto de 0,17% no BD-Rate. Foi realizada a síntese ASIC da descrição em nível *Register-Transfer Level* (RTL) da arquitetura utilizando a ferramenta *Mentor Graphics ModelSim* SE-64 6.5f. Como ponto negativo deste trabalho, a atividade de chaveamento utilizada na síntese foi gerada à partir da propagação de entradas aleatórias. Logo, a solução proposta por Corrêa (2017) não foi simulada com dados reais extraídos de vídeos.

3.5 CONSIDERAÇÕES FINAIS

Neste capítulo foram discutidas algumas propostas arquiteturais para a predição intra quadros do HEVC. A Tabela 3 resume características obtidas de soluções arquiteturais propostas nos trabalhos correlatos.

Tabela 3 – Resumo das características de soluções para predição intra quadros do HEVC.

Critérios	**Zhou (2013)**	**Cong Liu (2013)**	**Abram-owski (2014)**	**Corrêa (2017)**
pós-filtros	Não	Sim	Sim	Sim
Decisão de Modo	Único Melhor SAD	Único Melhor SATD	n/d	Oito Melhores SADs
BD-Rate	n/d	10%	n/d	0.17%
Simulação	n/d	n/d	Sim	Não
Tecnologia	TSMC 130 nm	TSMC 65 nm	TSMC 130 nm	*NanGate* 45 nm
Área (*gates*)	324 k	77 k	127 k	4952 k
Frequência	400 MHz	600 MHz	200 MHz	529 MHz
Potência	n/d	n/d	n/d	363,23 mW
Taxa de Proces.	HD 1080p 60 qps	HD 1080p 30 qps	HD 1080p 35 qps	UHD 8K 120 qps

É possível observar na Tabela 3 que apenas uma das propostas de predição fornece a informação de dissipação de potência. Essa informação, no entanto, é de extrema valia jus-

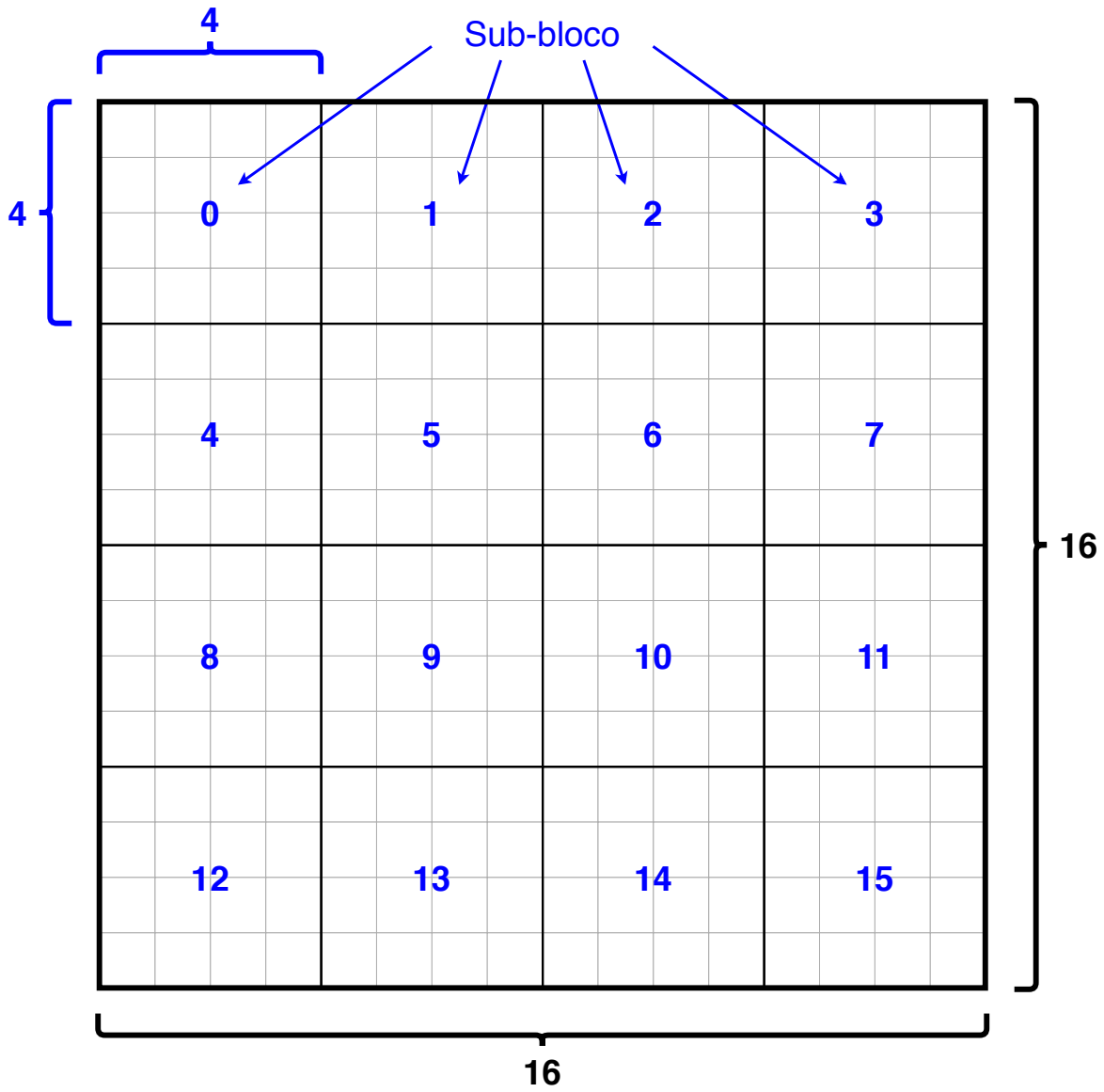
tamente pelo fato que as soluções podem ser embarcadas em dispositivos móveis operados à bateria. Nota-se que, a maioria dos trabalhos focam no processamento de vídeos em resolução *Full* HD (1080p). Também é possível notar que, embora a arquitetura proposta por Corrêa (2017) seja a única capaz de processar vídeo em resolução 8K, esta solução é no mínimo 15 vezes maior que as demais. Ainda, algumas informações importantes quanto às propostas não são disponibilizados em alguns trabalhos (e.g. impacto em BD-Rate e se foi realizada simulação com dados reais obtidos de um *benchmark* específico para testar codificadores de vídeo).

Vale lembrar que todos os trabalhos citados foram desenvolvidos tendo como base *drafts* compatíveis do HEVC. Também, todas as implementações realizam a predição intra quandros para qualquer tamanho de bloco e modos de predição previstos pelo padrão.

4 DESENVOLVIMENTO DA ARQUITETURA PROPOSTA

A arquitetura proposta neste trabalho é um tanto parecida com a proposta por Zhou et al. (2013). No trabalho de Zhou et al. (2013), a construção de um bloco candidato é realizada agregando ou dividindo blocos de tamanho 8×8 que, por sua vez, são processados em unidades genéricas chamadas de *Processing Element*. Neste trabalho, a arquitetura foi elaborada para construir candidatos através da composição de blocos de tamanho 4×4. Por convenção, os blocos de tamanho 4×4 serão chamados de sub-blocos, portanto, um candidato de tamanho 4×4 possui 1 sub-bloco, um de tamanho 8×8 possui 4 sub-blocos e assim por diante. A Figura 12 ilustra a lógica de construção de um candidato de tamanho 16×16.

Figura 12 – Lógica de construção de um candidato de tamanho 16×16.

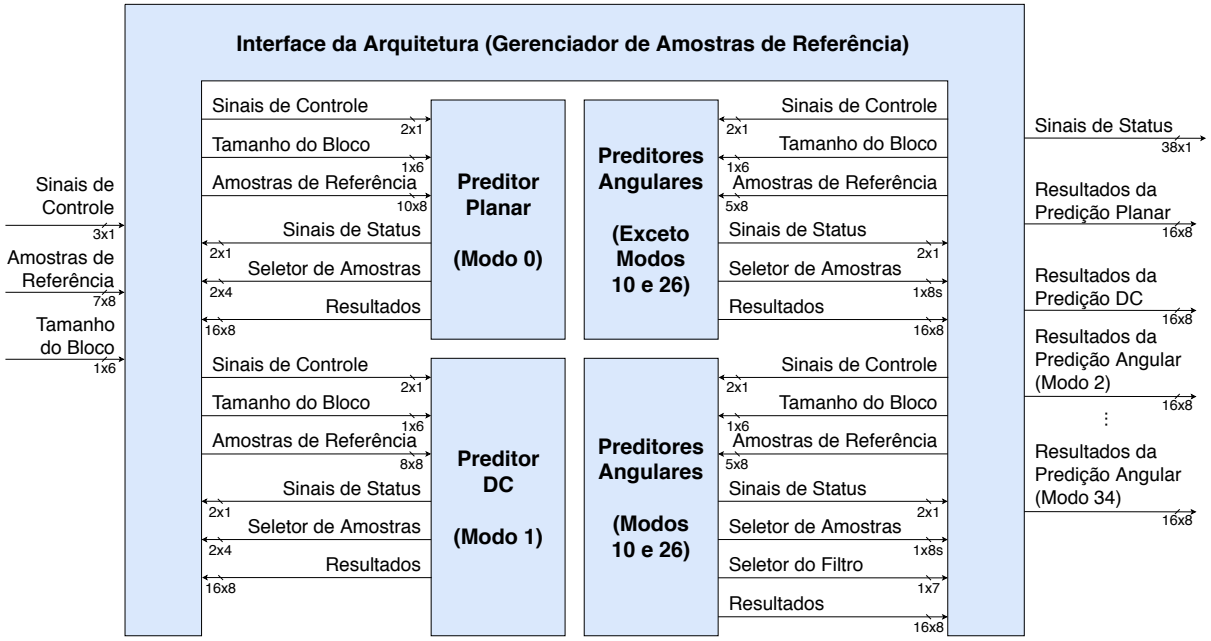


Fonte: o autor.

Neste trabalho, optou-se pela construção de preditores específicos para cada modo de predição e, tal como dito anteriormente, genéricos para diferentes tamanhos de blocos. A Figura

13 ilustra o diagrama de blocos da arquitetura proposta neste trabalho.

Figura 13 – Diagrama de blocos da arquitetura proposta neste trabalho. Nos fios, "s" significa que os dados são sinalizados.



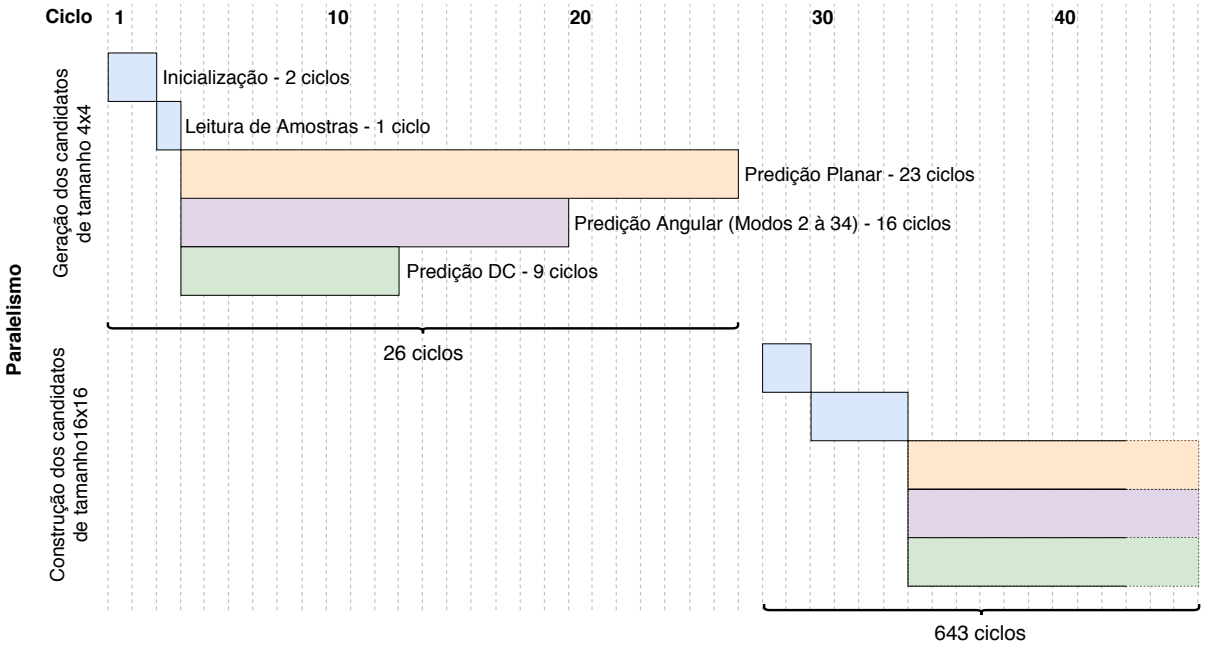
Fonte: o autor.

Como mostra a Figura 13, ao todo foram descritos cinco módulos em *hardware*: um responsável pela leitura e distribuição das amostras de referência, que também é a interface do acelerador; e quatro preditores responsáveis por gerar os 35 candidatos da predição intra quadros. Dois preditores são específicos para os modos planar e DC e os demais para os modos angulares. No caso dos preditores angulares, um é específico para a predição com os modos diretamente horizontal e vertical, enquanto que, o outro é específico para os demais modos angulares. A decisão de implementar preditores diferentes para os modos angulares está relacionada ao fato de que o ângulo dos modos 10 e 26 é zero. Dessa forma, os valores de deslocamento inteiro e fracionário também são sempre zero, o que simplifica a lógica de construção dos candidatos para esses modos. Também, como foi implementada a etapa de filtro pós-predição, os modos 10 e 26 são os únicos modos angulares dos quais o padrão prevê a aplicação desses filtros.

Os preditores angulares foram parametrizados para que, durante a compilação da arquitetura, o módulo seja especializado para um único modo de predição. Portanto, existem 2 instâncias do preditor específico para os modos 10 e 26 e 31 instâncias do preditor específico para os demais modos angulares, onde todos executam paralelamente. A Figura 14 apresenta um diagrama de tempo da arquitetura proposta.

No primeiro estágio do *pipeline* estão sendo gerados todos os 35 candidatos para um bloco original de tamanho 4×4, o que demanda um total de 26 ciclos de *clock*. Já no segundo estágio, estão sendo gerados os candidatos de um bloco original de tamanho 16×16, sendo necessários 643 ciclos de *clock*. A diferença no número de ciclos necessários na etapa de leitura

Figura 14 – Diagrama de tempo da arquitetura proposta neste trabalho.



Fonte: o autor.

de amostras, mostrada na Figura 14, decorre da maior quantidade de amostras de referência necessárias para a geração de candidatos de tamanho maior.

A lógica de construção dos candidatos reflete-se diretamente nas máquinas de estado dos preditores. Como pode ser visto na Figura 15, todas as máquinas de estados possui um *loop* para a construção de um sub-bloco (ilustrado em azul). Também, existe um *loop* maior para construção de todos os sub-blocos necessários para a compor o candidato de tamanho maior do que 4×4 (ilustrado em verde).

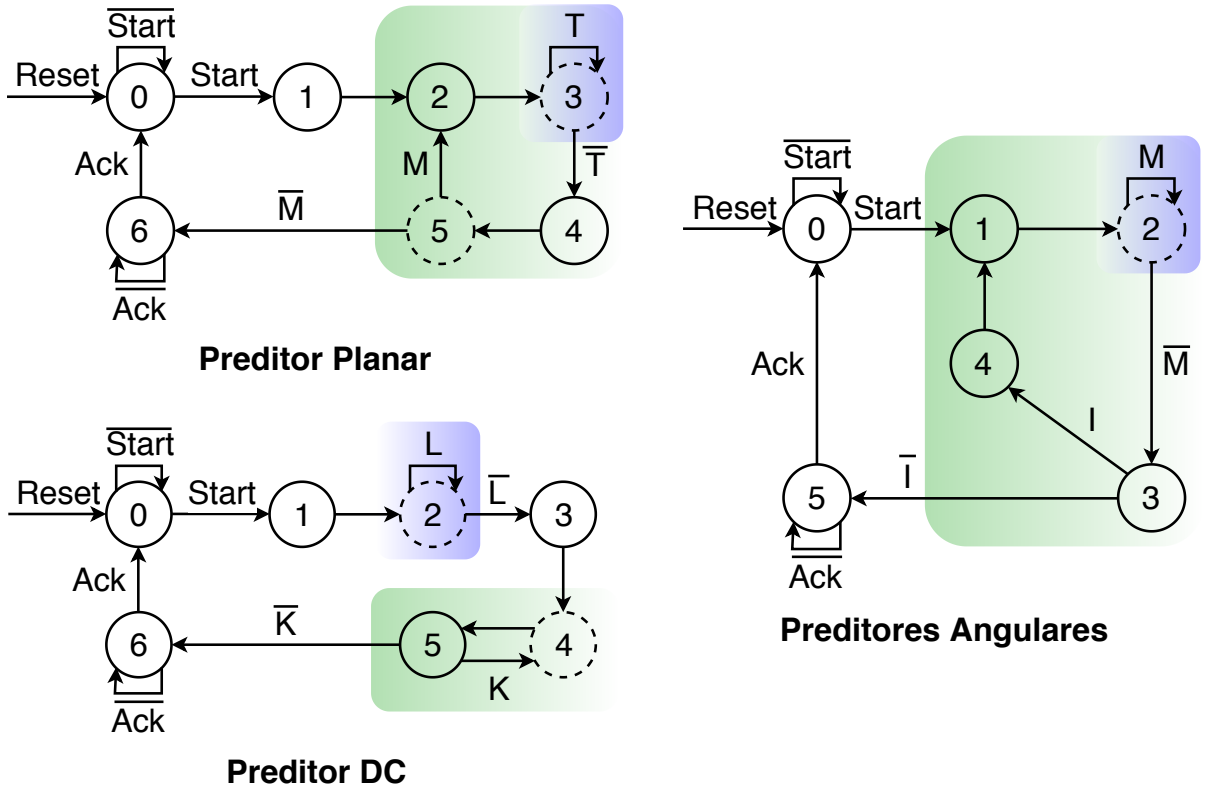
Nas seções seguintes, as máquinas de estados apresentadas na Figura 15 serão melhor detalhadas, bem como a iteração dos preditores com o módulo de gerência de amostras.

4.1 PREDITOR PLANAR

Não só no preditor planar, mas em todos os demais preditores, enquanto o sinal "*Start*" não estiver ativo o preditor permanece no estado inicial onde todos os registradores recebem o valor zero.

Como mostra a Figura 16, no estado 1 é realizada a leitura do tamanho do bloco (N), das amostras nas posições (-1,N) e (N,-1) e das amostras necessárias para a construção do primeiro sub-bloco do candidato planar (amostras de (-1,0) à (-1,3) e (0,-1) à (3,-1)). Vale citar que são sempre carregadas 8 amostras para a construção de um sub-bloco, porém, não necessariamente a posição dessas amostras correspondem às posições do vetor de amostras. No estado 1, também serão selecionadas as máscaras que, junto ao iterador "m", serão utilizadas no cálculo dos seletores de amostras em "x" e "y". A Figura 17 ilustra o cálculo dos seletores de amostras em x e y, bem como o sinal de controle "M" e o valor log(N)+1, utilizado na Equação

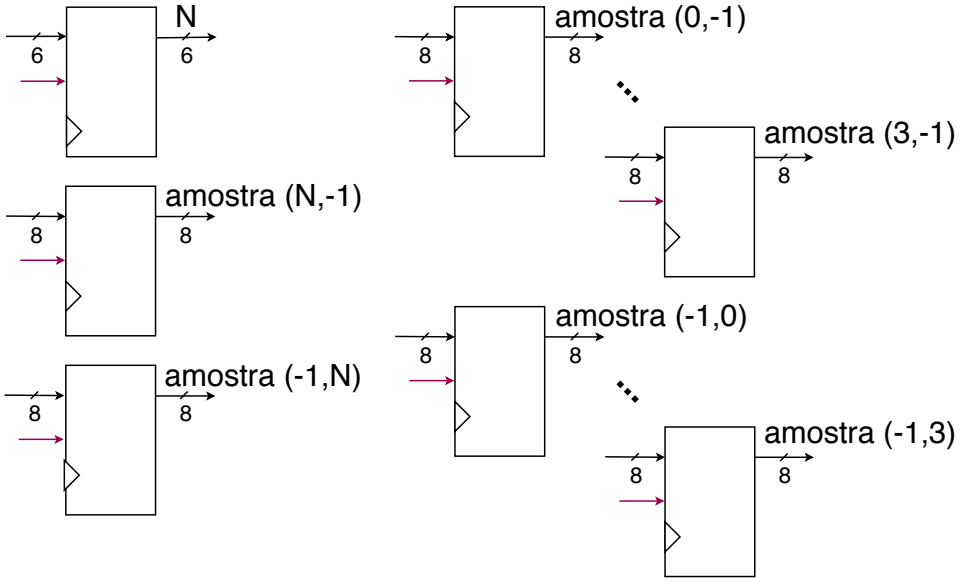
Figura 15 – Máquinas de estados simplificadas dos preditores.



Fonte: o autor.

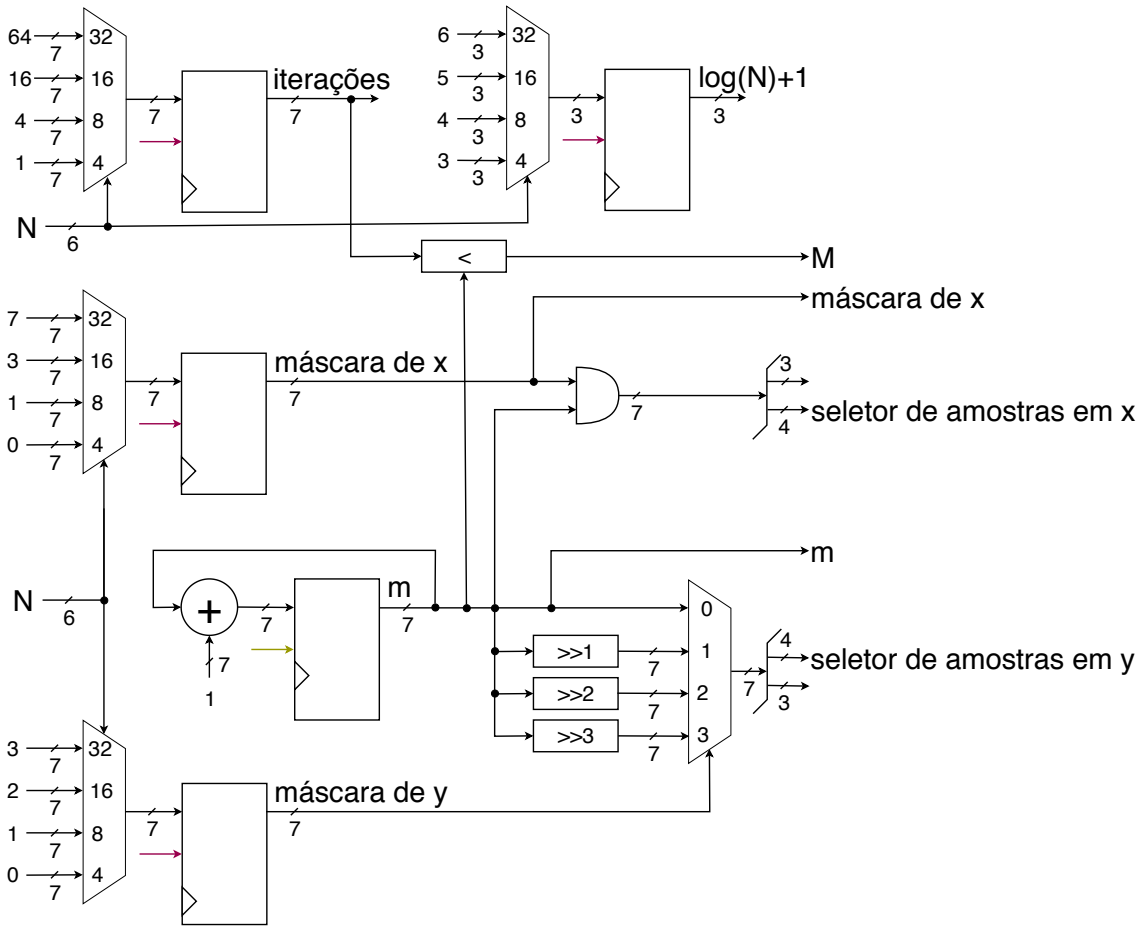
9.

Figura 16 – Parte do bloco operativo do preditor planar responsável pela leitura dos dados de entrada. Os sinais coloridos nos registradores correspondem aos sinais de "*enable*". Sinais de "*enable*" com mesma cor correspondem ao mesmo sinal. Sinais sem especificação de número de *bits* possui 1 único *bit*.



Fonte: o autor.

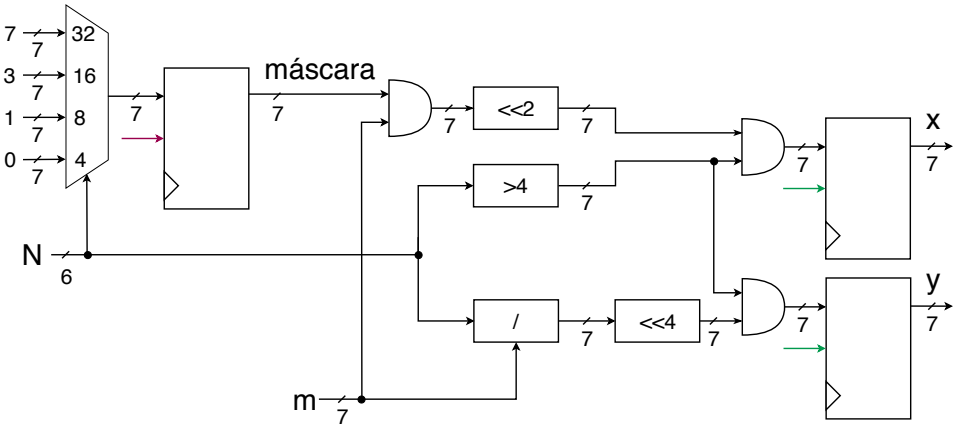
Figura 17 – Parte do bloco operativo do preditor planar responsável pela seleção de máscaras, cálculo de iteradores e do sinal de controle “M”.



Fonte: o autor.

No estado 2 é realizado o cálculo dos índices “x” e “y”, tal como ilustra a Figura 18. Tais índices são utilizados na construção dos blocos intermediários (ph e pv) e da matriz final (p), bem como especificam as equações (7) - (9).

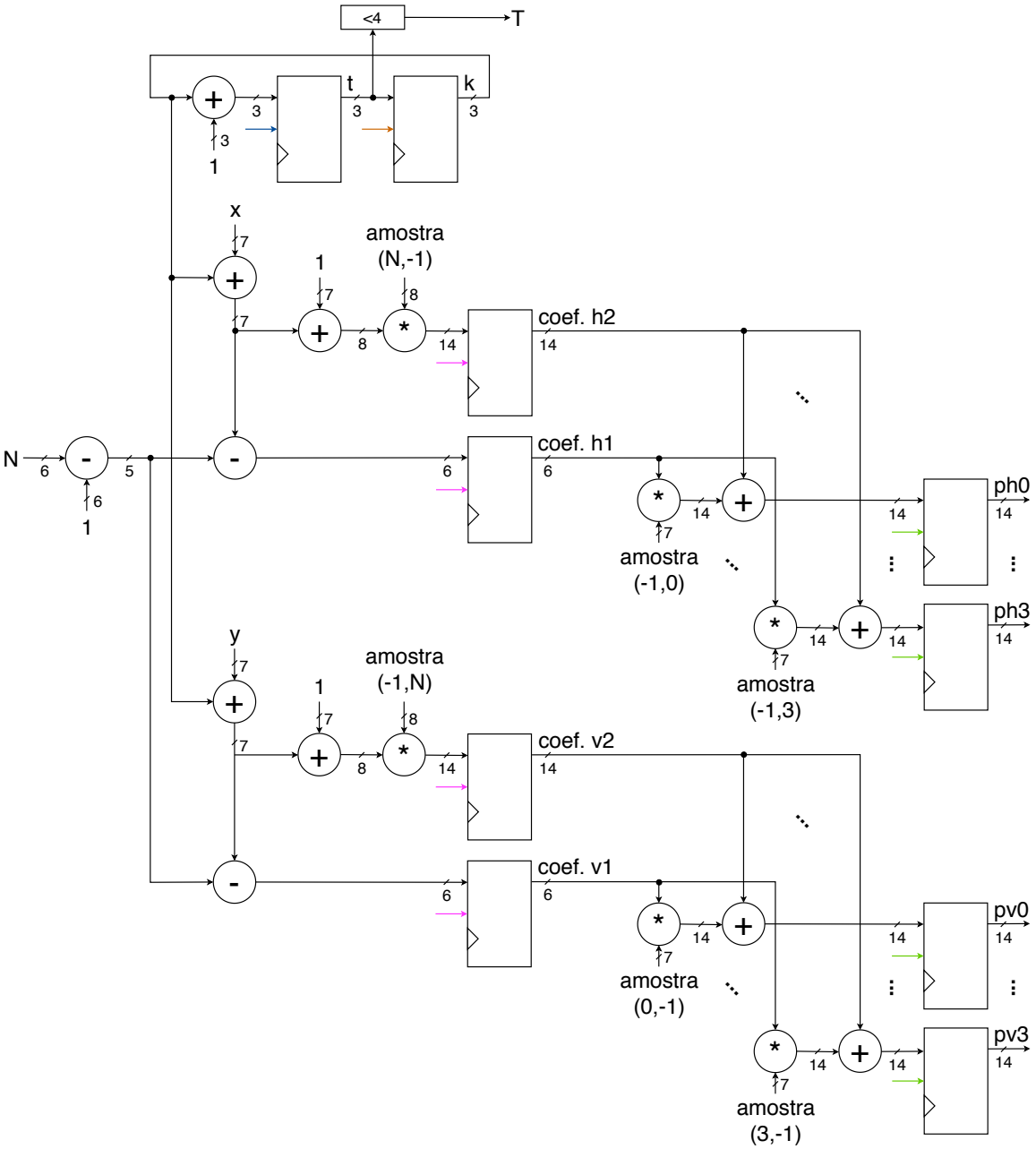
Figura 18 – Parte do bloco operativo do preditor planar responsável pelo cálculo dos índices “x” e “y” com base no valor da máscara selecionada, no tamanho do bloco e do índice “m”.



Fonte: o autor.

No estado 3, tal como ilustra a Figura 19, a condição “T” é calculado através do índice “t” que, por sua vez, varia de 0 à 4. A Figura 19 também ilustra o cálculo do índice “k”, utilizado para ajustar o valor dos índices “x” e “y” de acordo com o sub-bloco que está sendo calculado.

Figura 19 – Parte do bloco operativo do preditor planar responsável pelo cálculo do sinal de controle “T” e dos coeficientes utilizados nas equações (7) e (8). Também é mostrado o cálculo dos valores de uma linha do bloco *pv* e uma coluna do bloco *ph*.

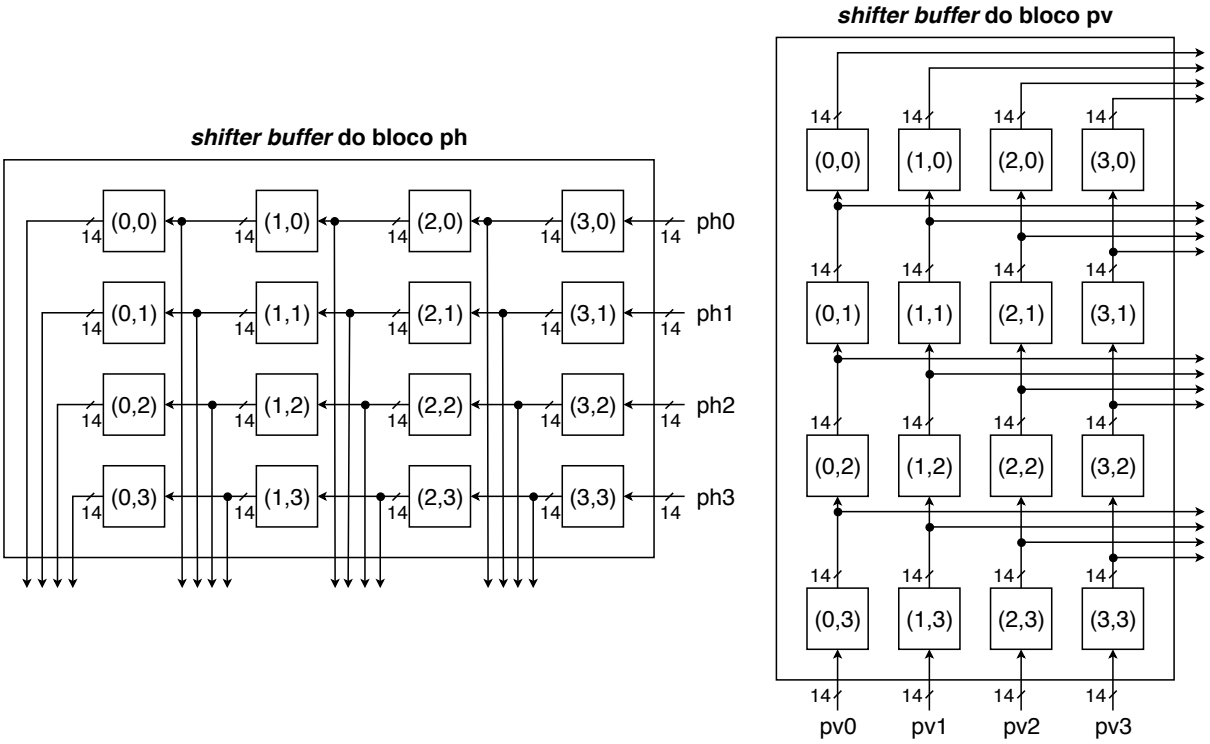


Fonte: o autor.

A construção dos blocos intermediários (*ph* e *pv*) é feita linha a linha para o bloco *pv* e coluna a coluna para o bloco *ph*. A Figura 19 ilustra o cálculo dos coeficientes que multiplicam o valor das amostras, tal como definem as Equações (7) e (8). Posteriormente, tais coeficientes serão utilizados para calcular os valores finais dos blocos *ph* e *pv*. Como a cada ciclo tem-se uma linha do bloco *pv* e uma coluna do bloco *ph*, é utilizado um *shift buffer* para armazenar os

valores das matrizes. A Figura 20 ilustra o funcionamento do *shift buffer* implementado neste trabalho.

Figura 20 – *Shift buffers* utilizados no preditor planar para armazenamento dos valores dos blocos intermediários *pv* e *ph*.



Fonte: o autor.

No estado 4 é onde será construído o sub-bloco final, tal como define a Equação (9). A Figura 21 ilustra como essa etapa é realizada pelo bloco operativo do preditor planar.

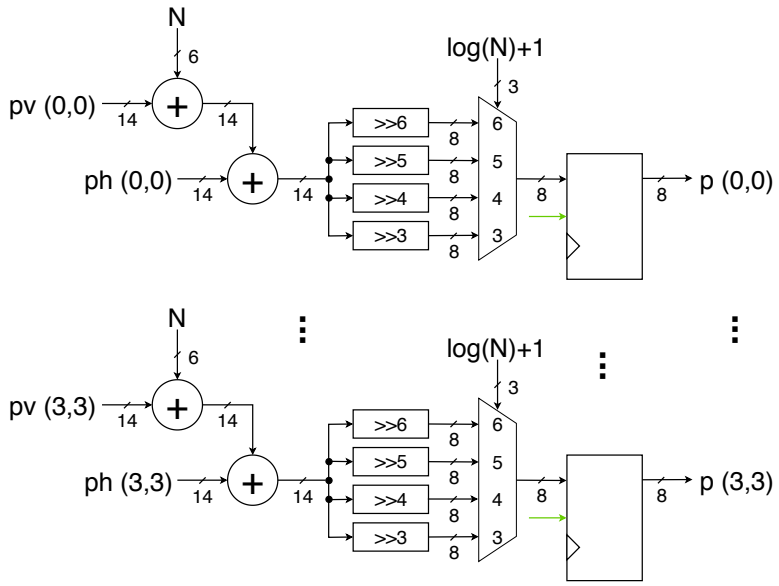
No estado 5 será verificado, através do sinal de controle "M", a necessidade da construção de outro sub-bloco. A construção do sinal "M" pode ser vista na Figura 17. Caso "M" seja verdadeiro, será reiniciado o ciclo de construção do sub-bloco, senão a máquina de estados segue para o estado final. No estado 5 também será acionado um sinal de "*status*" para notificar que o sub-bloco está construído. Dessa forma, é possível a captura do sub-bloco e comparação com os valores gerados pelo *software* de referência.

Assim como todos os demais preditores, quando todos os sub-blocos foram construídos o preditor vai para o estado final e permanece nele enquanto os sinais "Ack" ou "*Reset*" não for ativado.

4.1.1 Seleção de amostras

O módulo de gerência de amostras é fundamental para o correto funcionamento dos demais módulos da arquitetura. Seu principal papel consiste na leitura e armazenamento de todas as amostras de referência necessárias para a geração dos candidatos de acordo com o tamanho do bloco e, também, distribuir amostras de acordo com a demanda dos preditores. A

Figura 21 – Parte do bloco operativo do preditor planar responsável pelo cálculo dos valores finais do sub-bloco.



Fonte: o autor.

Figura 22 ilustra a parte do bloco operativo do módulo de gerência de amostras responsável pela leitura e armazenamento dos dados de entrada.

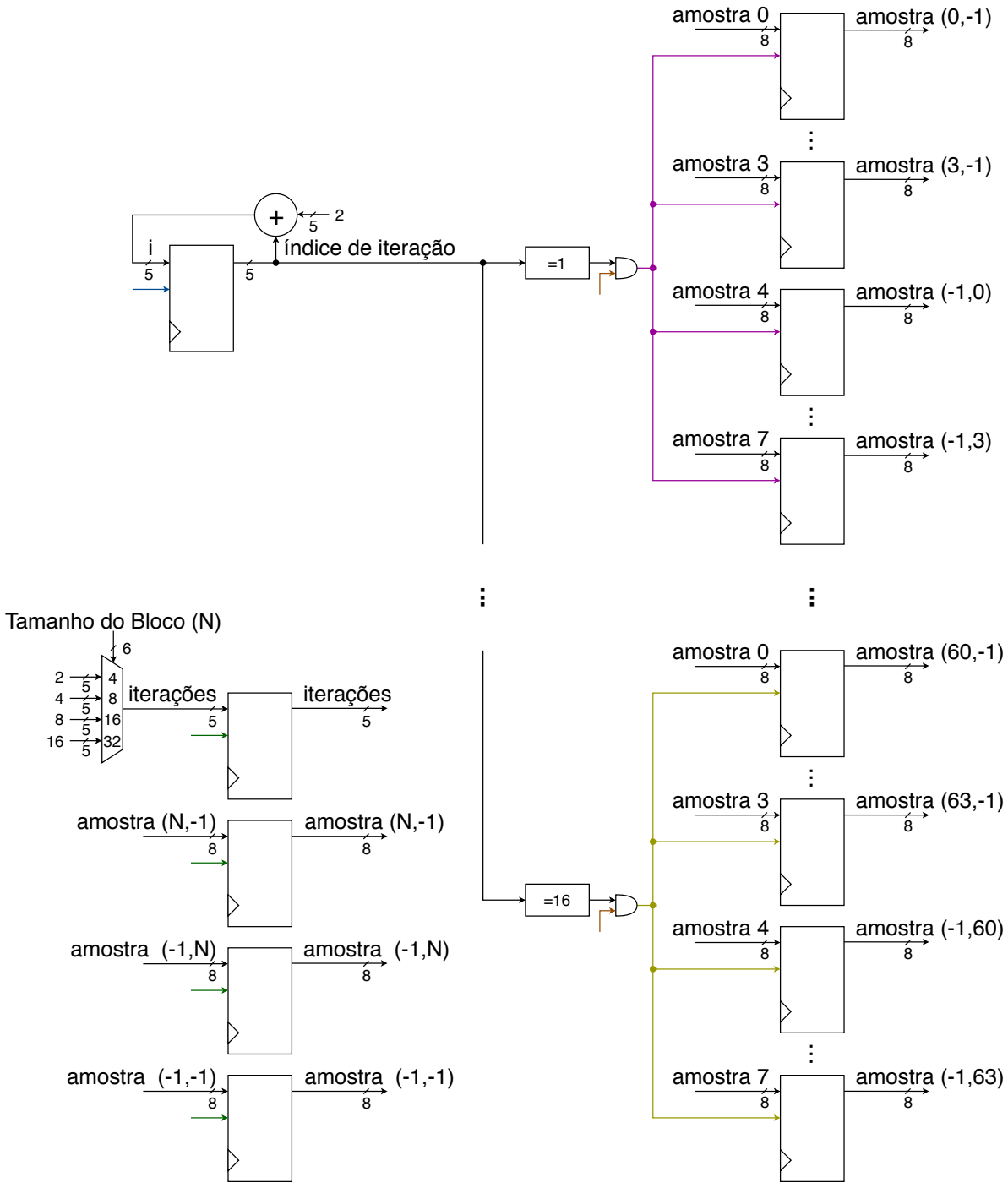
Como a leitura é realizada de 8 em 8 amostras (4 horizontais e 4 verticais), foi necessária a utilização de uma lógica para manter a consistência das amostras nesses registradores. No total, existem 131 registradores de amostras, onde: 64 são para amostras horizontais, 64 para as verticais, 1 para a posição (-1,-1), 1 que mantém uma cópia da amostra (N,-1) e outro para uma cópia da amostra (-1,N). As cópias são mantidas simplesmente para facilitar a lógica de distribuição para o preditor planar.

O controle realizado para manter a leitura e armazenamento consistente das amostras é feito através da operação "AND" entre dois sinais de controle: o "*enable*" dos registradores e o sinal "índice de iteração" que, por sua vez, define o número de iterações necessário para a ler todas as amostras de referência. Assim, o índice de iterações faz com que o sinal de "*enable*" de um conjunto de 8 registradores só fique ativo quando é a iteração deles serem atualizados, enquanto que, o sinal de "*enable*" dos demais registradores permanece inativo. Exceto pelos registradores de cópia e o da amostra na posição (-1,-1), cuja leitura é sempre realizada em um único ciclo, a quantidade de ciclos utilizada para realizar a leitura de todas as amostras de referência depende do tamanho do bloco. Para blocos de tamanho 4×4, 8×8, 16×16 e 32×32 são necessários 2, 4, 8 e 16 ciclos para a leitura de amostras, respectivamente.

A Figura 23 mostra como é realizada a distribuição de amostras para o preditor planar.

A distribuição é realizada com base nos dois seletores de amostras apresentados na Figura 17. Ilustrativamente, as amostras na horizontal são divididas em 8 conjuntos de 4 amostras sequenciais, totalizando 32 amostras (total que é utilizado na predição planar). Logo, a primeira amostra selecionada no "MUX" da Figura 23 será uma entre as amostras de posição 1

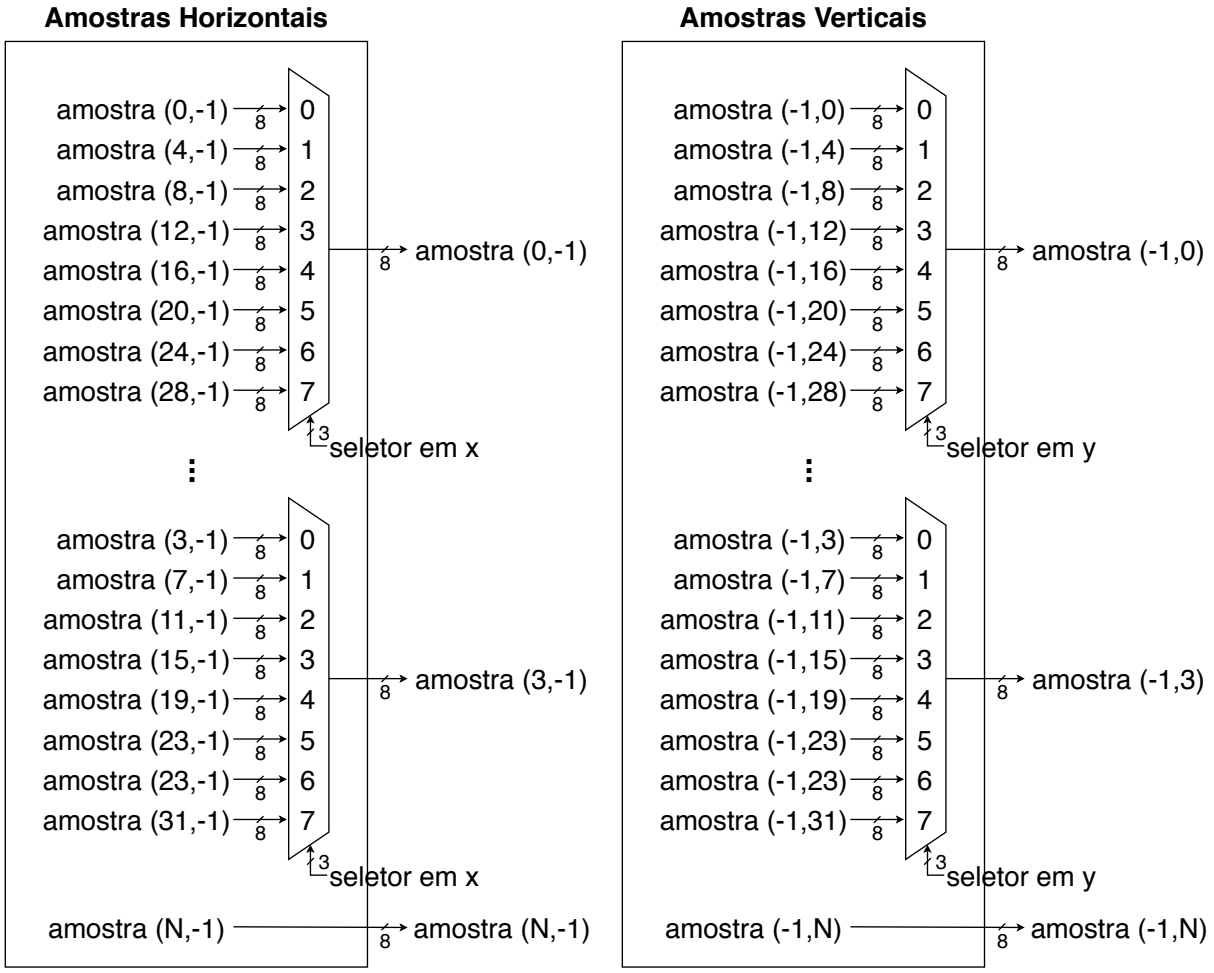
Figura 22 – Parte do bloco operativo do módulo de gerência de amostras responsável pela leitura e armazenamento dos dados de entrada.



Fonte: o autor.

dos 8 conjuntos, a segunda amostra selecionada será uma entre as amostras de posição 2 dos 8 conjuntos e assim por diante. Para as amostras na vertical a lógica é a mesma. Finalmente, as amostras de posição (N,-1) e (-1,N) serão enviadas ao preditor planar.

Figura 23 – Parte do bloco operativo do módulo de gerência responsável pela distribuição de amostras de acordo com a demanda do preditor planar.



Fonte: o autor.

4.2 PREDITOR DC

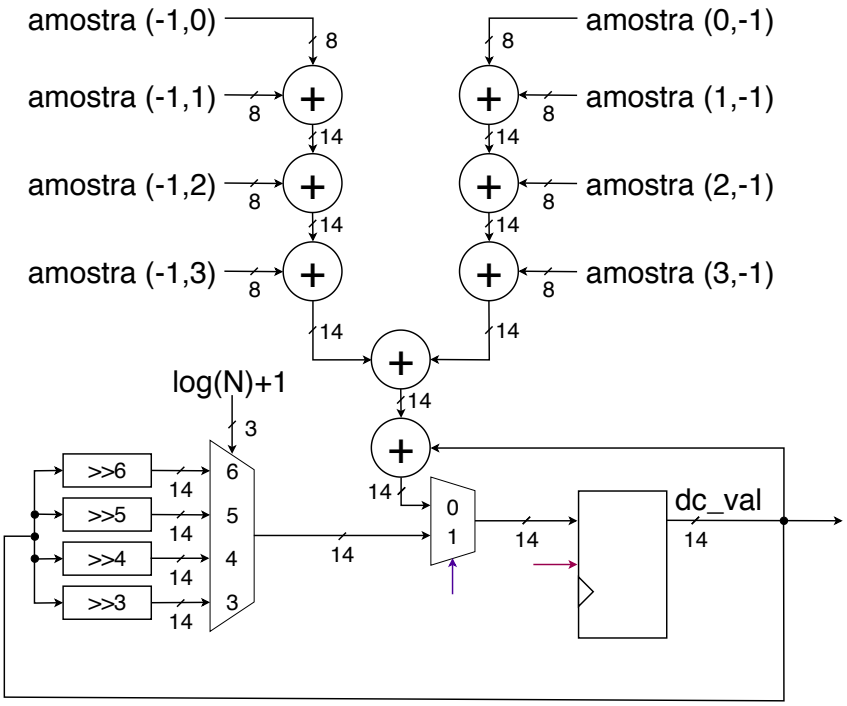
Bem como no estado 1 do preditor planar, no estado 1 do preditor DC São carregados os dados de entrada e selecionados iteradores e máscaras. Também, no estado 1 o registrador "dc_val" é inicializado com o valor de N.

Como ilustra a Figura 24, no estado 2 será realizada a leitura de 8 amostras de referência e a soma acumulativa delas no registrador "dc_val". Vale citar que, assim como o preditor planar, o preditor DC inclui lógicas de controle do número de sub-blocos e como eles são gerados;

No estado 3 será finalizado o cálculo do valor do "dc_val", i.e., o valor acumulado no registrador será deslocado log(N)+1 *bits* à direita, tal como define a Equação (10) e ilustra a Figura 24.

Como o padrão prevê a aplicação de filtro pós-predição para o modo DC, no estado 4 serão gerados todos os sub-blocos do candidato, tanto os que terão as bordas filtradas como os que não precisam de filtro. Os "K" sub-blocos de um candidato são gerados tendo como

Figura 24 – Parte do bloco operativo do preditor DC responsável pelo cálculo do “dc_val” de acordo com a Equação (10).



Fonte: o autor.

controladores os sinais “x” e “y”. Como pode ser visto na Figura 25, tais controladores são gerados a partir de uma máscara e do índice “i” que, por sua vez, controla o número de iterações. Assim sendo, “x” tem valor 1 apenas quando o sub-bloco coincide com a borda superior do candidato e, de forma análoga, “y” tem valor 1 apenas quando o sub-bloco coincide com a borda esquerda do candidato.

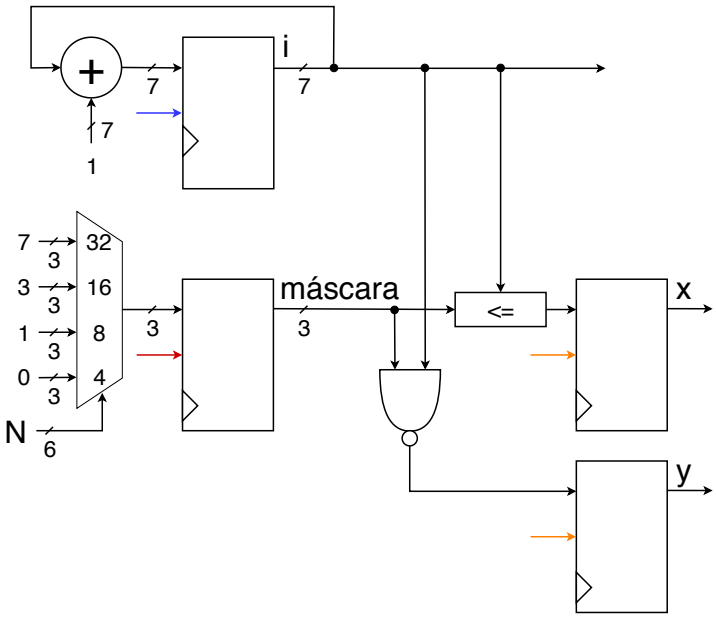
A Figura 26 permite visualizar como os sinais de controle são utilizados para selecionar o valor que será atribuído a cada posição do sub-bloco. As posições que não fazem parte da borda do sub-bloco não podem coincidir com uma borda do candidato, portanto, simplesmente recebem o valor do “dc_val”. Nos demais caso, os sinais de controle permitem selecionar se o valor que que será salvo no registrador é simplesmente o valor do “dc_val”, quando não coincide com a borda do candidato, ou então o valor do filtro quando coincide com uma borda. Ainda, para a posição (0,0), que pode tanto estar em apenas uma das bordas como em ambas ou nenhuma delas, a combinação dos sinais de controle através da soma permite definir qual o correto valor a ser atribuído no registrador em detrimento dos quatro possíveis casos.

Os demais estados não citados nesta seção possuem os mesmos comportamentos dos respectivos estados do preditor planar.

4.2.1 Seleção de amostras

A interação do preditor DC com o módulo de gerência de amostras é bastante similar da interação do preditor planar, exceto por duas características. Primeiro, como mostra a Figura

Figura 25 – Parte do bloco operativo do preditor DC responsável pela seleção da máscara e, também, do cálculo dos sinais de controle “x” e “y”.



Fonte: o autor.

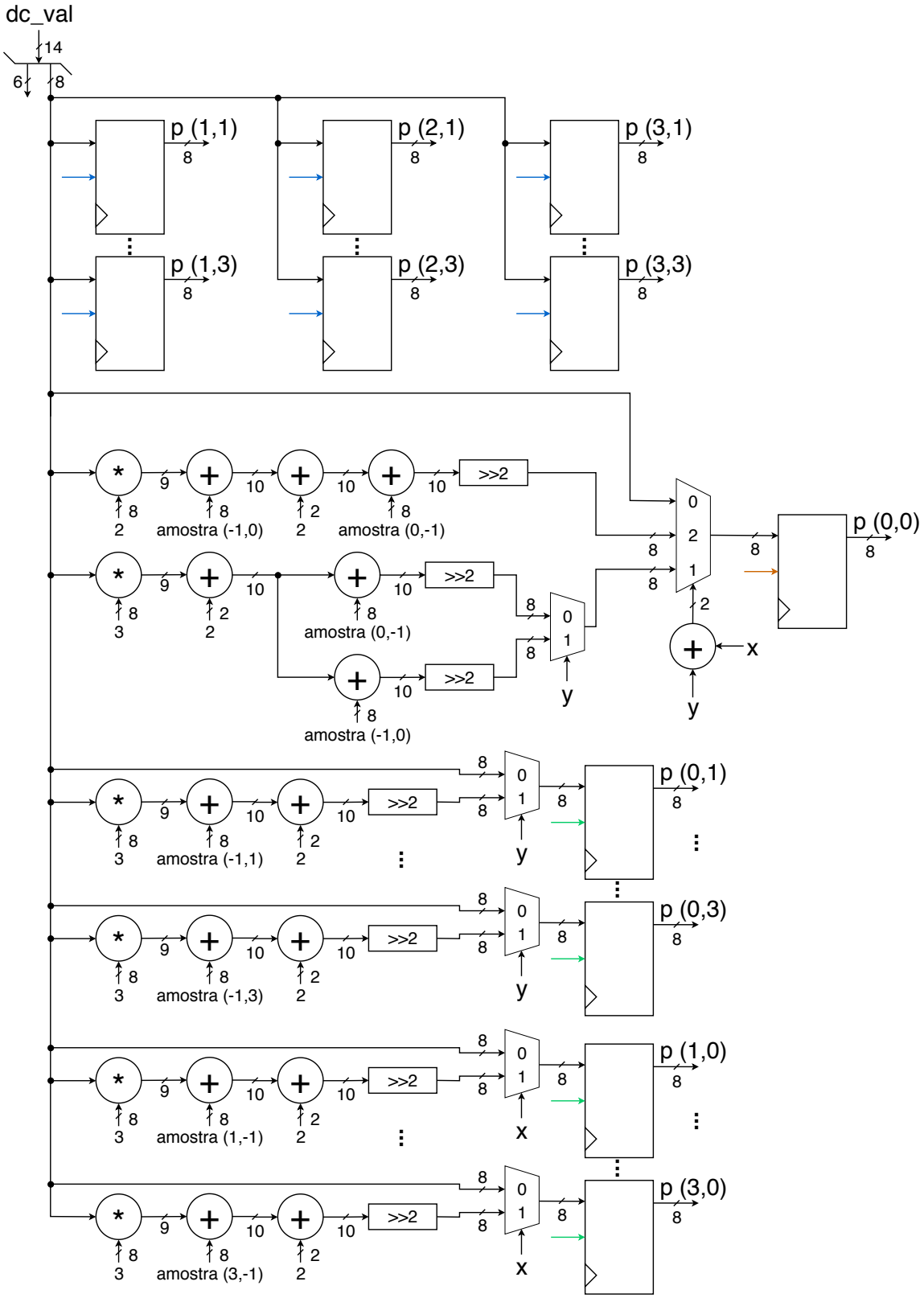
23, o preditor planar utiliza as amostras nas posições (-1,N) e (N,-1), as quais não são utilizadas no preditor DC. Segundo, devido à aplicação dos filtros pós-processamento e, para evitar armazenar novamente as amostras dentro do preditor, durante a construção dos sub-blocos, as amostras de referências são lidas novamente. Logo, existem duas etapas de leitura de amostras do preditor DC, uma para o cálculo do “dc_val” e outra na aplicação do filtro durante a geração dos sub-blocos.

4.3 PREDITORES ANGULARES

Para simplificar a implementação dos modos de predição angular, foram criados dois preditores genéricos: um para os modos diretamente horizontal e vertical (10 e 26, respectivamente) e outro para os demais modos angulares. Essa decisão foi tomada em decorrência da predição ser mais simples para os modos diretamente horizontal e vertical e, também, por eles terem aplicação de filtros pós-predição. O genérico desses preditores diz respeito apenas a sua implementação, i.e., ele é parametrizado para que, durante a instanciação do módulo, seja possível definir para qual modo de predição ele será especializado. A partir daí, o modo passa a ser específico para um único modo de predição. Logo, existem 33 instâncias de preditores angulares, cada qual definindo um ângulo de predição diferente.

Como ambos os preditores angulares são bem parecidos, a seguir será discutido o preditor angular que abrange a maioria dos modos e, ao mesmo tempo, serão apontadas suas diferenças em relação ao preditor dos modos 10 e 26. A quantidade de estados e as principais atividades realizadas de ambos os preditores são iguais, tal como ilustra a Figura 15, mudando apenas alguns detalhes.

Figura 26 – Parte do bloco operativo do preditor DC responsável pela seleção dos valores que compõem o sub-bloco sendo gerado.

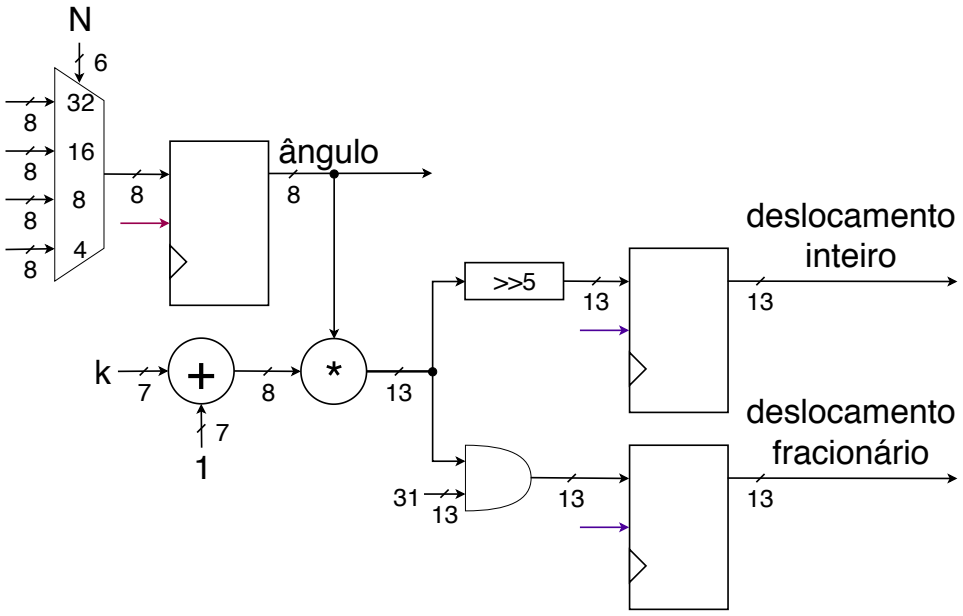


Fonte: o autor.

No estado 1, além de carregar os dados de entrada, também são calculados os deslocamentos inteiro e fracionário, tal como definem as Equações (13) - (16) e ilustra a Figura 27.

Na Figura 27, “k” representa o índice “x” quando o modo angular é horizontal e “y” quando vertical. Esse cálculo pode ser feito logo no início já que a operação depende apenas do ângulo (carregado no estado 0) e dos índices “x” e “y” que, por sua vez, serão sempre zero na construção do primeiro sub-bloco. Quanto ao preditor para os modos 10 e 26, estes índices não precisam ser calculados, já que os deslocamentos são sempre nulos.

Figura 27 – Parte do bloco operativo do preditor angular responsável pelo cálculo dos deslocamentos inteiro e fracionário.



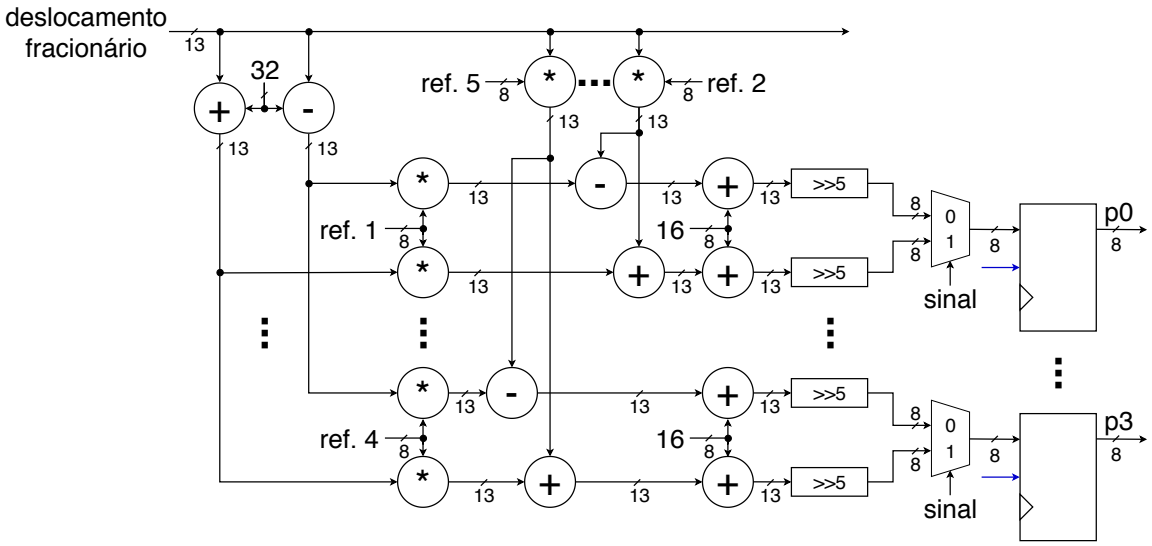
Fonte: o autor.

Tal como o preditor DC, é possível construir uma linha ou coluna do sub-bloco, quando o modo é vertical ou horizontal, respectivamente, fazendo o reuso de dados pré-calculados. Como as amostras utilizadas para o cálculo de uma posição do sub-bloco podem ser utilizadas para o cálculo das posições seguintes (de uma mesma linha ou coluna), são necessárias apenas 5 amostras de referência. Por conta disso, em cada iteração no estado 2 serão lidas cinco amostras de referência e construído uma linha ou coluna do sub-bloco, tal como ilustra a Figura 28. Como o deslocamento é sempre nulo no preditor dos modos 10 e 26 (tal como definido nas Equações (17) e (18)), o cálculo é simplificado por não fazer uso da segunda amostra de referência. Para evitar cálculos com números sinalizados, foi criado um sinal que indica quando o deslocamento é positivo ou negativo.

Tal como o preditor planar, é necessário manter os resultados das linhas ou colunas sendo construídas, onde foi utilizado o mesmo *shift buffer* do bloco intermediário *ph*, ilustrado na Figura 19.

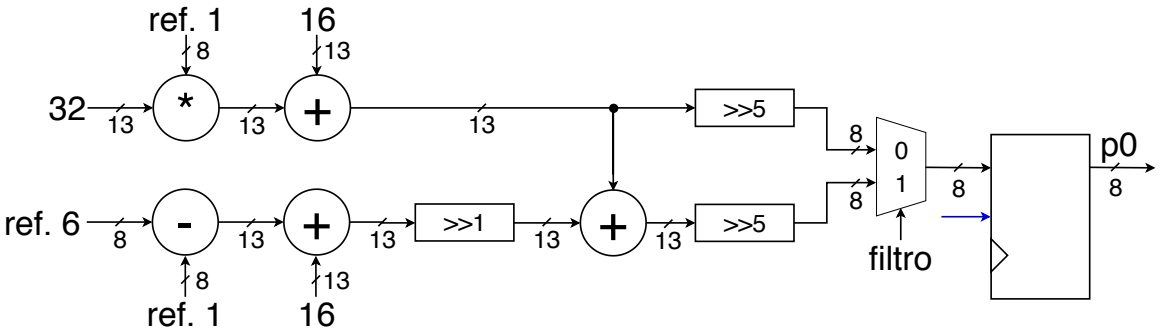
Ainda no estado 2, o preditor dos modos 10 e 26 faz a aplicação do filtro. Para isso, o preditor recebe uma sexta referência, tal como mostram as Equações (22) e (23) e ilustra a Figura 29. Portanto, é utilizado um sinal de filtro que fica ativo quando o sub-bloco está em uma das bordas do candidato.

Figura 28 – Parte do bloco operativo do preditor angular responsável pelo cálculo de uma linha ou coluna do sub-bloco, dependendo se o modo angular é vertical ou horizontal, respectivamente.



Fonte: o autor.

Figura 29 – Parte do bloco operativo do preditor angular dos modos 10 e 26 responsável pela aplicação do filtro no início de uma linha ou coluna do sub-bloco, dependendo se o modo angular é vertical ou horizontal, respectivamente.



Fonte: o autor.

No estado 4 é realizada apenas a atualização de índices “x” e “y”, uma vez que no estado 1 serão calculados os deslocamentos novamente.

Os estados 3 e 5 são equivalentes aos estados 5 e 6 dos preditores planar e DC, respectivamente.

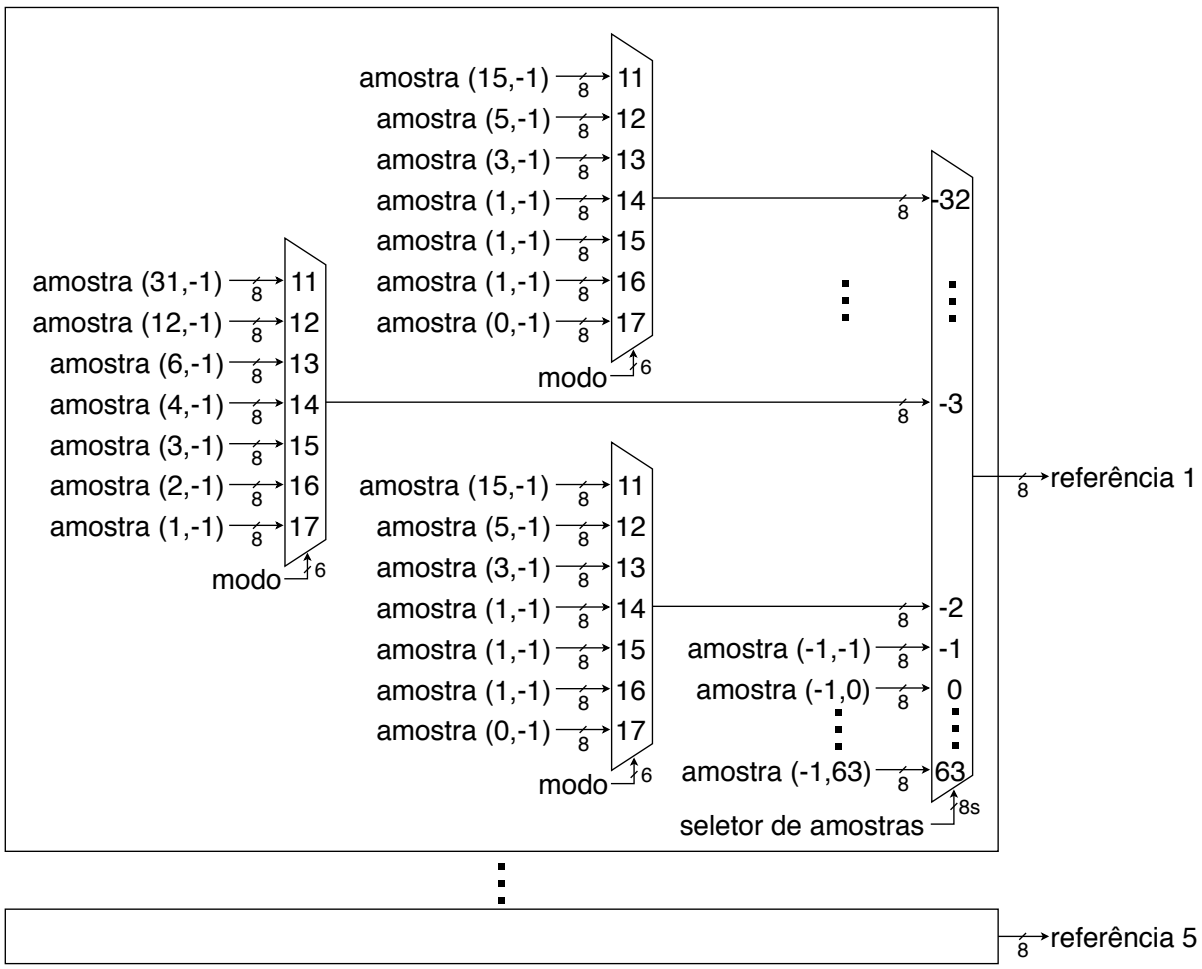
4.3.1 Seleção de amostras

Existe uma grande diferença na forma de acesso das amostras de referência dos modos angulares em relação aos demais modos. O padrão HEVC define um mapeamento das amostras de referência para um vetor unidimensional que pode ser acessado tanto em posições positivas quanto negativas. Porém, para evitar o intermédio no acesso das amostras, neste trabalho foi realizado um mapeamento direto das amostras para as posições do sub-bloco sendo construído.

A lógica utilizada nesse mapeamento leva em consideração a forma de construção do vetor unidimensional e, portanto, não altera as amostras utilizadas na predição de um bloco.

Diferente dos modos planar e DC, existe um único sinal de seleção de amostras para os preditores angulares. Como é realizada a construção linha a linha ou coluna a caluna do sub-bloco, dependendo do modo, o índice "y" ou "x", respectivamente, é suficiente para definir as amostras necessárias para construção desse sub-bloco em uma dada linha ou coluna. Assim, o sinal de controle foi gerado a partir do índice "k" (representa "x" quando o modo é horizontal e "y" quando vertical) e do sinal "i" (valor do deslocamento inteiro). A soma dos índices "k" e "i" permite discriminar todas as amostras acessadas em posições positivas, mas nem todas as amostras acessadas em posições negativas. Uma vez que a construção do vetor unidimensional em posições negativas leva em consideração o inverso do ângulo do modo, a lógica de seleção das amostras levou em consideração os modos também. A Figura 30 ilustra a forma de seleção das amostras para os modos angulares.

Figura 30 – Parte do bloco operativo do módulo de gerência de amostras responsável pela distribuição de amostras de acordo com a demanda dos preditores angulares, exceto os diretamente horizontal e vertical.



Fonte: o autor.

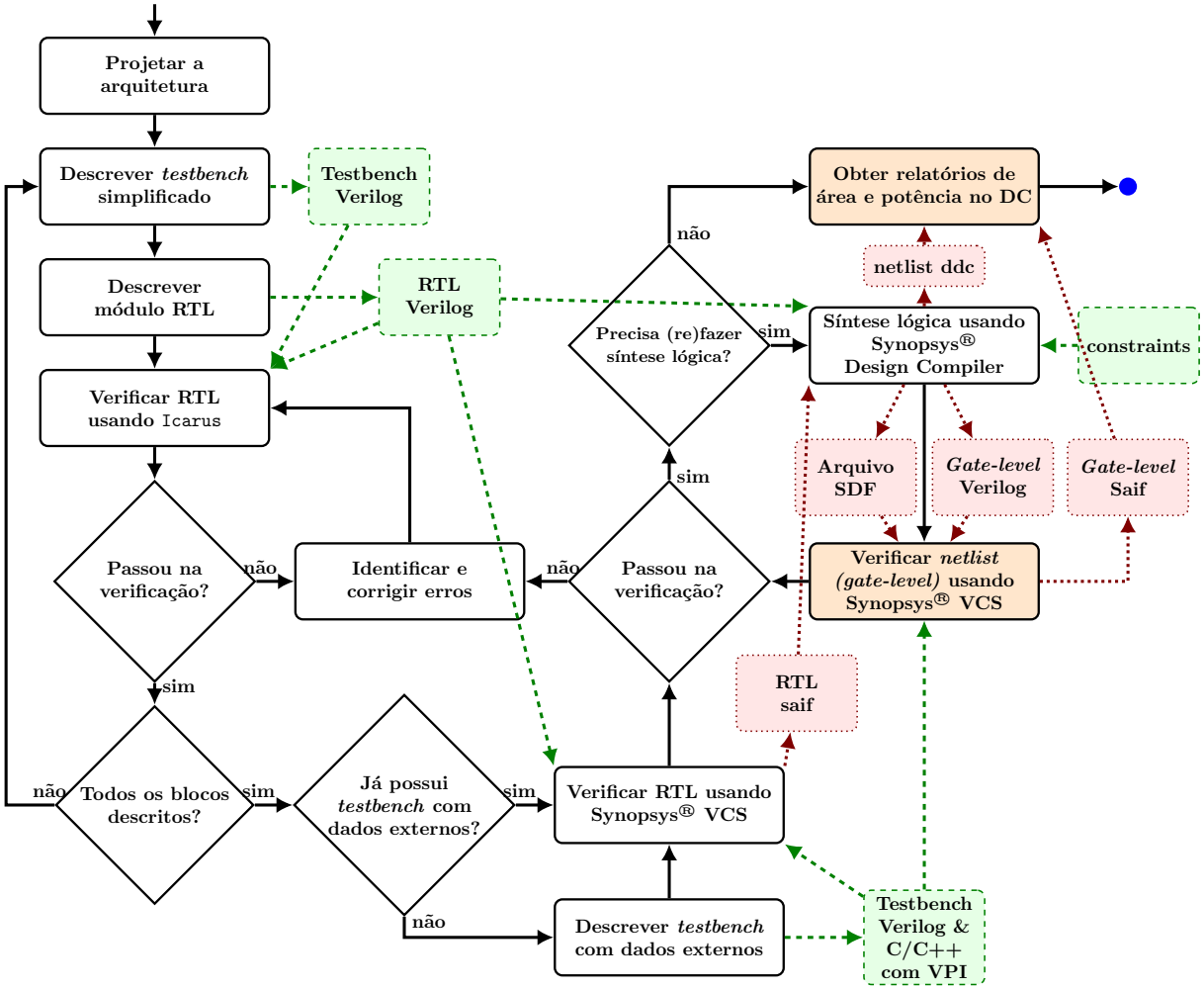
É possível notar na Figura 30 que quando o seletor de amostras (soma de "i" mais "k") é maior ou igual a -1 as amostras de referência são únicas e independem no modo. No

entanto, quando o ângulo do modo é negativo (nesse caso o seletor de amostras é menor que -1) é possível notar que o valor da amostra varia de acordo com o modo. Supondo que durante a predição angular de um bloco a soma dos sinais “i” e “k” é -3, por exemplo, o valor da “referência 1” será o valor da amostra na posição (31,-1) se o modo for 11, o valor da amostra na posição (12,-1) se o modo for 12 e assim por diante.

5 MÉTODO DE TRABALHO

O desenvolvimento deste trabalho foi realizado baseando-se no método utilizado por Seidel et al. (2019). A Figura 31 ilustra cada uma das etapas desse método de trabalho, as quais foram de suma importância para o desenvolvimento e avaliação realista da arquitetura proposta neste trabalho. Na Figura 31, os arquivos em verde foram criados pelo projetista e os vermelhos foram gerados pelas ferramentas comerciais. Também, as etapas em laranja não foram realizadas neste trabalho, onde foram consideradas como possíveis trabalhos futuros. Cada uma das etapas do fluxograma será descrita em detalhes nas seções seguintes.

Figura 31 – Método de síntese e projeto para garantir confiabilidade nas estimativas de área e potência.



Fonte: adaptado de Seidel et al. (2019).

5.1 PROJETAR A ARQUITETURA

Antes de qualquer etapa que envolvesse implementação de *hardware*, foi necessária a realização de uma etapa de projeto. Nesta etapa é onde a arquitetura de *hardware* proposta foi

especificada detalhadamente, conforme mostrado no Capítulo 4. Primeiramente, foram especificados os módulos internos da arquitetura, qual o objetivo de cada um deles, os algoritmos que eles deveriam ser responsáveis por implementar e quais os sinais de entrada e saída seriam necessários para tal. Para cada módulo anteriormente especificado foram elaborados os seguintes blocos: de controle, responsável por coordenar as operações do módulo; operativo, responsável por executar as operações necessárias do algoritmo; sub-módulos, neste trabalho tratando-se apenas de *buffers*; *top level*, responsável por interconectar os sinais entre os demais blocos e servir de interface do módulo.

5.2 DESCREVER *TESTBENCH* SIMPLIFICADO E MÓDULO RTL

Estas etapas correspondem ao início das implementações necessárias para cada módulo e, portanto, cada iteração está relacionada a um módulo diferente da arquitetura.

Através do projeto, que especifica as entradas e saídas do módulo em questão, na primeira etapa foi realizada a implementação de um *testbench* simplificado para o mesmo. Tal *testbench* fica responsável por fornecer estímulos para o módulo e verificar se suas saídas correspondem ao esperado. Neste trabalho, os estímulos fornecidos pelo *testbench* simplificado foram especificados em seu próprio código. Ainda, os estímulos utilizados nesta etapa buscaram avaliar o correto funcionamento do módulo sob três condições: maiores, menores e valores médios dos sinais de entrada. Vale citar que toda implementação foi realizada utilizando *Verilog Hardware Description Language (HDL)* (Verilog HDL) (Thomas; Moorby, 2010).

Com base no projeto do módulo, na segunda etapa foi realizada a implementação dos blocos, tais como citados anteriormente. Por fim, como resultados destas etapas têm-se a descrição, em Verilog HDL, do módulo ao qual se trata a iteração e o *testbench* simplificado para testá-lo.

5.3 VERIFICAR RTL USANDO ICARUS

Tendo como entrada as descrições de um módulo e o *testbench* simplificado para testá-lo, nesta etapa é realizada a verificação do módulo. Para isso, foi utilizada a ferramenta gratuita *Icarus Verilog* onde é possível realizar uma primeira verificação de forma rápida e simplificada. Embora esta etapa de verificação não estimule o módulo de forma precisa (e nem utilize dados reais para tal), ela ainda permite visualizar o funcionamento dele e localizar inconsistências, bem como erros de sintaxe da linguagem.

Ao final da verificação utilizando o *Icarus Verilog*, o método de síntese adotado apresenta quatro possíveis alternativas para continuar o fluxo. Abaixo são resumidas as condições para seleção de qual das etapas deve ser adotada a seguir:

1. **Identificar e corrigir erros:** adotada quando a verificação do modo falha.

2. **Descrever *testbench* simplificado:** adotada quando a verificação do modo atual não falha, mas nem todos os módulos foram implementados. Neste caso, reinicia-se o ciclo de implementações para um outro módulo.

3. **Descrever *testbench* com dados externos:** adotada quando todos os módulos foram implementados e verificados corretamente, mas ainda não se tem o *testbench* que utiliza dados externos.

4. **Verificar RTL usando o *Synopsys Verilog Compiler Simulator*® (VCS®):** adotada quando todos os módulos foram validados corretamente e já existe o *testbench* que utiliza dados externos.

5.4 IDENTIFICAR E CORRIGIR ERROS

Esta etapa é utilizada em duas situações: a primeira durante a implementação do módulo, havendo erros durante a verificação com o *Icarus Verilog*; a segunda quando houveram erros durante a verificação da arquitetura utilizando o VCS®.

No primeiro caso, e mais simples de resolver já que o problema está mais isolado, basta depurar o módulo e corrigir o problema, bem como os erros de sintaxe da linguagem. No segundo caso, primeiro é necessário identificar onde está ocorrendo o problema, que pode complicar dependendo do tamanho da arquitetura e das decisões de projeto adotadas. Vale lembrar que, em algumas situações, um erro pode gerar alterações no projeto da arquitetura devido a problemas que não haviam sido percebidos até o momento da integração dos módulos.

Finalizadas as correções, o fluxo segue para uma nova etapa de verificação utilizando o *Icarus Verilog* e assim por diante, conforme descrito anteriormente.

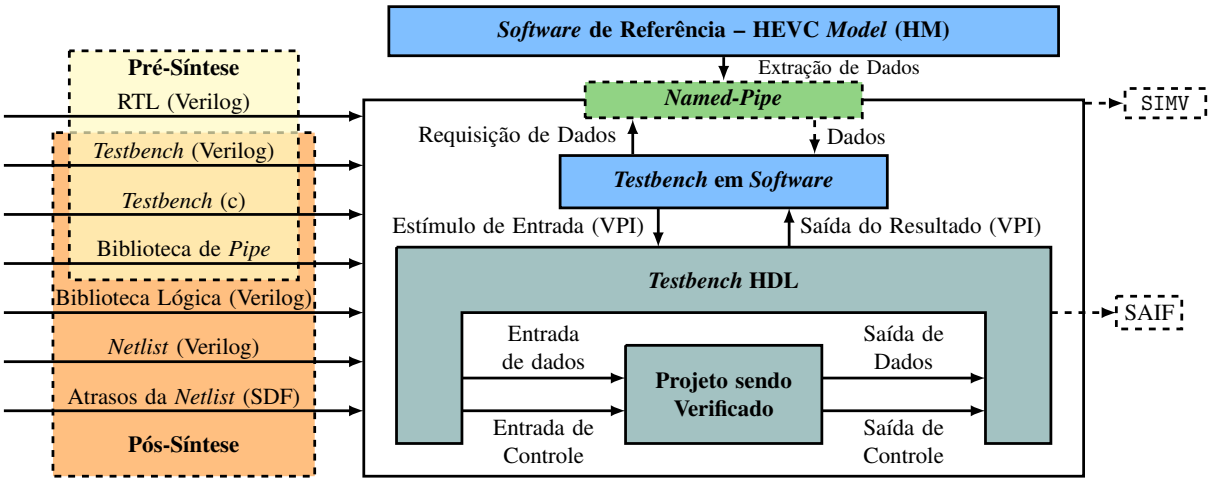
5.5 DESCREVER *TESTBENCH* COM DADOS EXTERNOS

Para realizar a verificação e validação da arquitetura, é necessário um *testbench* que forneça dados realistas a ela. Nesta etapa do método de síntese, é realizada a implementação desse *testbench* que, por fim, é responsável por ser a interface da arquitetura proposta com o mundo exterior. Para isso, neste trabalho foi utilizado o *framework* proposto por Bonotto et al. (2018), tal como ilustra a Figura 32. Este *framework* define uma estrutura básica para verificação de projetos de *hardware* (chamado de "projeto sendo verificado" na Figura 32).

Observe, na Figura 32, que existem dois *testbenches* no *framework*: um *Testbench* HDL (tb.v) e um *Testbench software* (tb.c). Abaixo serão descritas as características e atribuições de cada um dos *testbenches*:

- **tb.v**:
  - Módulo de *hardware* onde foi instanciada a arquitetura proposta neste trabalho;

Figura 32 – Modelo de simulação utilizando o *framework* proposto por Bonotto et al. (2018).



Fonte: adaptada de Seidel et al. (2019).

  - Fornece os dados e comandos de entrada (estímulos) necessários para simular a arquitetura;
  - Monitora os dados e controles de saída da arquitetura;
  - Monitora os sinais da arquitetura para gerar o arquivo, no *Switching Activity Interchange Format* (SAIF), com a atividade de chaveamento da arquitetura durante a simulação.

- **tb.c**:
  - Lê os dados realísticos necessários (fornecidos por um *software* de referência) dos *named-pipes* e os salva nos registradores do tb.v.
  - Monitora os sinais do tb.v e trata os eventos necessários, como por exemplo, comparar a resposta da arquitetura sendo simulada com a resposta fornecida pelo *software* de referência.

5.6 VERIFICAR RTL USANDO SYNOPSYS®VCS

A verificação do RTL utilizando o VCS®é realizada através de simulações, as quais possuem duas principais finalidades: viabilizar a verificação funcional da arquitetura e obter atividades de chaveamento da mesma durante as simulações.

Os dados de entrada de todas as simulações foram extraídos do vídeo *BQTerrace* (*Full* HD - 1080p), uma amostra de vídeo do conjunto de amostras especificadas pelas *Common Test Conditions* (CTC). As CTC, por sua vez, constituem um *benchmark* planejado para testar codificadores (Bossen, 2012) e, portanto, são adequadas para serem utilizadas na simulação realista de *hardware* para codificação de vídeo. Também, foi utilizado o HM (versão 16.16) como *software* de referência e de onde foram capturados os dados necessários durante a codificação da amostra de vídeo citada acima.

Em cada simulação foram utilizados um total de 100 mil vetores de teste, i.e., 100 mil conjuntos mínimos de dados necessários para que a arquitetura possa processá-los e gerar uma saída correspondente. Segundo experimentos realizados por Silveira et al. (2017), constatou-se que 50 mil vetores de teste são suficientes para estimular todas as partes do *hardware* e, portanto, estabilizar as estimativas de consumo de energia. Logo, é utilizado o dobro de vetores de teste propostos por Silveira et al. (2017) como garantia de que toda a arquitetura será estimulada.

A compilação da arquitetura foi realizada na ferramenta VCS®, gerando um único arquivo executável, o qual é chamado de "simv" na Figura 32. Nesta etapa, a compilação tem como entrada os itens que, na Figura 32, são chamados de "Pré-Síntese", i.e., a descrição RTL da arquitetura, o tb.v, o tb.c e a biblioteca de *pipe* (implementada em linguagem C). A biblioteca de *pipes*, disponibilizada por Bonotto et al. (2018), é o que permite a comunicação entre o *software* de referência e o "simv" da arquitetura proposta.

À fim de definir os períodos de *clock* para os quais a arquitetura foi simulada e sintetizada, foram necessárias três principais etapas:

1. Determinar o número de ciclos de *clock* necessários para gerar um bloco de tamanho 4×4 e 32×32 para todos os modos de predição. Gerar blocos de tamanho 8×8 e 16×16 demandam uma quantidade de ciclos de *clock* entre as quantidades necessárias para gerar blocos de tamanho 4×4 e 8×8. Neste sentido, o período máximo do *clock* para gerar os blocos de tamanho 8×8 e 16×16 também estariam entre os períodos dos blocos de demais tamanhos. A Tabela 4 resume os valores obtidos. Nota-se que o pior caso deste trabalho ocorre quando o tamanho do bloco é 4×4 e o modo é planar;

2. Calcular o período máximo do *clock* da arquitetura para uma dada resolução de vídeo, taxa de qps e tamanho de bloco, o que pode ser feito através da Função (F1), sendo: "ciclos" uma função que retorna o número de ciclos necessários para gerar um bloco de tamanho N através de um modo M. A Tabela 5 resume os principais resultados obtidos com a Função F1 para as resoluções *Full* HD e 4k e taxas de 30 e 60 qps. Ainda, como a geração de blocos de tamanho 4×4 através do modo planar constituem o pior caso da arquitetura aqui proposta, os parâmetros N e M foram fixados para este caso em específico.

3. Definir as frequências alvo para síntese e simulação e quais QPs utilizados no HM para codificar a amostra de vídeo selecionada. Neste trabalho, as frequências alvo definidas estão na terceira coluna da Tabela 5 e os QPs utilizados são os sugeridos pelas CTC, i.e., 22, 27, 32 e 37.

$$\text{Período(Largura, Altura, qps, N, M)} = \frac{N^2 * 10^9}{\text{Largura} * \text{Altura} * \text{qps} * \text{ciclos(N, M)}} \quad \text{(F1)}$$

Tabela 4 – Número de ciclos de *clock* necessários, por módulo de predição, para geração de um bloco de tamanho 4×4 e 32×32.

Modo	**Tamanho do Bloco**	**Número de ciclos**
Planar	4×4	23
	32×32	1283
DC	4×4	9
	32×32	343
Angular	4×4	16
	32×32	897

Tabela 5 – Resultados da aplicação da Função F1 tendo como parâmetros as resoluções *Full* HD e 4k e as taxas de 30 e 60 qps. N e M estão fixados em 4 e planar, respectivamente.

Resolução	**qps**	**Período do *clock*** (ns)	**Período alvo** (ns)
Full HD	30	11,183	10
	60	5,591	5
4k	30	2,796	2,5
	60	1,398	1,25

5.7 SÍNTESE LÓGICA USANDO SYNOPSYS®DC

A etapa de síntese lógica é responsável pelo mapeamento da descrição da arquitetura de alto nível (RTL) para a representação em *gate-level* otimizado. Esse processo foi realizado automaticamente pela ferramenta comercial DC®. A *netlist* gerada pela ferramenta já leva em consideração a atividade de chaveamento, restrições de projeto e também da tecnologia alvo, i.e., uma biblioteca *standard-cell*. Neste trabalho, foi utilizada uma biblioteca *standard-cell* de 45nm. Ainda, a ferramenta também realiza avaliações sobre a arquitetura, permitindo a obtenção de estimativas de área, potência, análises de tempo, entre outras informações.

Assim como pode ser visto no fluxo da Figura 31, a etapa de síntese lógica utiliza como entrada a descrição RTL da arquitetura, o SAIF em nível RTL e as *constraints* do *clock*, i.e., período, *delay*, latência etc. Como saída, já é possível obter estimativas de área e potência, o arquivo de atrasos do circuito em *Standard Delay Format* (SDF), o *gate-level* da arquitetura e a *netlist* DDC que contém o *gate-level* junto das *constraints*.

A arquitetura proposta neste trabalho foi sintetizada utilizando a técnica *clock-gating* para reduzir o consumo de energia. A técnica *clock-gating* tem por objetivo evitar o chaveamento desnecessário do *clock* (Keating et al., 2002). Em circuitos sequenciais, o sinal de *clock* controla toda a questão de temporização e, portanto, permite a sincronização dos dispositivos. Para distribuir este sinal a todas as regiões do circuito, é necessário uma árvore de *clock* que propague este sinal (Qing Wu; Pedram; Xunwei Wu, 2000). No entanto, todo o chaveamento realizado pela árvore de *clock* tem um impacto significativo na potência final do circuito, podendo ser responsável por mais de 40% dela (DONNO et al., 2003) e, portanto, do consumo de energia.

O critério para a utilização da técnica é satisfeito quando existe algum módulo da arquitetura que depende de um sinal de controle (*enable*) para funcionar. Assim, quando o módulo não estiver operando, o sinal de controle pode ser utilizado para evitar a propagação do *clock* e, dessa forma, reduzir o consumo de energia. Para isso, basta que seja realizada a operação lógica "AND" entre os sinais de *enable* e *clock* para evitar o chaveamento do *clock* quando uma determinada parte do circuito não mudará de estado.

Embora a técnica permita uma redução considerável na potência, isso não vem sem custos: a adição de novas células, além de aumentar a área, pode também aumentar o atraso do circuito. Porém, algumas ferramentas são capazes de aplicar *clock-gating* de forma automática (e.g. DC®) e, portanto, desaconselha-se a aplicação da técnica de forma manual devido ao grande potencial de falhas (Keating et al., 2002). Também, através do DC®é possível obter informações à respeito da quantidade de elementos adicionados para realizar *clock-gating* e o número de registradores afetados por eles.

5.8 VERIFICAR *NETLIST* (*GATE-LEVEL*) USANDO *SYNOPSYS*®VCS

Assim como a verificação do RTL, a verificação da *netlist* é realizada através de simulações. Neste caso, a compilação da arquitetura para geração do "simv" tem como entrada os arquivos da "Pós-Síntese" mostrados na Figura 32, i.e., o arquivo SDF com os atrasos, a biblioteca lógica de células e todos os arquivos da "Pré-Síntese" exceto a descrição RTL da arquitetura que, agora, é substituída pela *netlist*.

Diferente das primeiras simulações, estas utilizam os atrasos gerados pelo DC®, fazendo com que seja possível capturar aspectos temporais (como *glitches* temporais na *netlist*) que podem influenciar nas estimativas de potência (Silveira et al., 2017). Logo, através do *gate-level* SAIF, gerado durante esta etapa, é possível obter estimativas mais realistas de potência.

Durante a execução dessa etapa, embora a primeira síntese não tenha acusado nenhuma violação de tempo, o VCS®reportou algumas violações de tempo. Em consequência, a simulação utilizando os atrasos não foi bem sucedida, onde os dados de saída gerados pela arquitetura não correspondiam aos obtidos no *software* de referência.

Após uma análise do preditor planar, verificou-se de que os dados estavam sendo calculados corretamente. Porém, os registradores da saída do preditor, responsáveis por ar-

mazenar os resultados até eles serem utilizados, não estavam sendo atualizados corretamente. Foi investigado o sinal de *enable* desses registradores e concluiu-se que estes se tratavam de *gated-registers*, i.e., registradores onde foi aplicado *clock-gating*. Em virtude desse problema, a primeira síntese foi realizada novamente e, desta vez, não foi utilizada a otimização de *clock-gating*.

Ao simular a *netlist* novamente verificou-se outro problema, a simulação entrava em *loop* infinito executando por horas mesmo para poucos casos de teste. Neste caso, por não encerrar a simulação, não foi possível gerar um arquivo *Value Change Dump* (VCD) com os sinais do circuito, impossibilitando a verificação dos sinais do *hardware* durante a simulação. Pequenas modificações foram realizadas na arquitetura para tentar resolver o problema, tais como: tornar síncrono todos os *resets* assíncronos; remover a definição da escala de tempo do *clock* das descrições em Verilog HDL e defini-la como parâmetro no comando de compilação com o VCS®; entre outras pequenas modificações.

As modificações realizadas possibilitaram que a arquitetura obtivesse sucesso em alguns casos de teste, mas ainda sem encerrar a simulação. Para poder gerar o arquivo VCD das simulações executadas, foi criado um "monitor" no tb.v para finalizar a execução da arquitetura quando um sinal de *status* fosse acionado. Avaliando o VCD gerado dessa pequena execução, foi possível notar que diversos registradores não tinham seu valor zerado corretamente no início da simulação. Logo, toda a execução da arquitetura partia de um estado indefinido.

Tal como anteriormente, foi investigado o motivo que estava levando à essa inconsistência da arquitetura. Novas modificações foram feitas, como por exemplo: definir explicitamente o estado indefinido das máquinas de estado (*default*); alterar os operadores de atribuições de valores aos registradores; entre outros. Ainda com essas correções o valor de alguns registradores permaneceram indefinidos. Como última solução, tentou-se construir o circuito de um registrador baseado na *netlist* e na descrição das células da biblioteca utilizada. Isolando o caso, foi possível notar que o registrador analisado tinha o sinal *reset* como parte do valor sendo atualizado. No entanto, alguns sinais utilizados na lógica estavam indefinidos e, portanto, os resultados armazenados no registrador não eram consistentes.

Dado que foi possível obter estimativas de área e potência durante a a síntese do RTL, optou-se por não realizar a síntese e simulação da *netlist*, ficando essas etapas como possíveis trabalhos futuros. Sabe-se que as estimativas obtidas através da síntese do RTL não são tão precisas, porém, como a ferramenta não reportou nenhum erro ou violação de tempo durante a síntese do RTL, a mesma pode ser utilizada como resultados preliminares.

5.9 OBTER RELATÓRIOS DE ÁREA E POTÊNCIA NO SYNOPSYS®DC

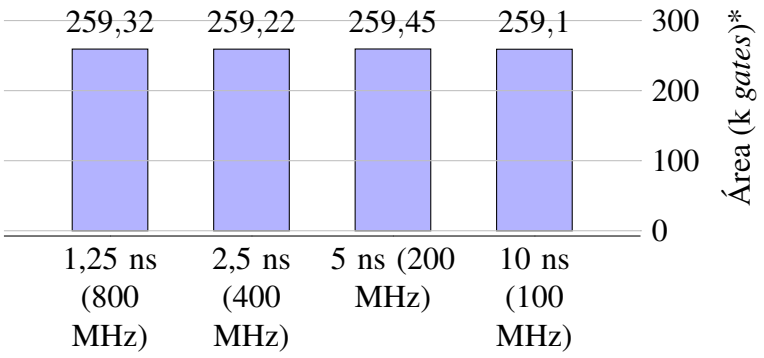
Tendo os *gate-level* SAIFs como saída das simulações da *netlist*, através da ferramenta DC®é possível obter precisas estimativas de área e consumo de energia. Assim sendo, é necessário passar como entrada para a ferramenta dois arquivos além das bibliotecas de células: a *netlist* DDC e o *gate-level* SAIF.

6 RESULTADOS OBTIDOS

Tal como definido na Tabela 5, a arquitetura proposta foi simulada para quatro períodos de *clock*. Cada período, por sua vez, foi definido para que a arquitetura fosse capaz de gerar os blocos candidatos para uma dada resolução e qps. Para a verificação e validação da arquitetura, foi considerado apenas o processamento de blocos de tamanho 4×4, cenário em que a arquitetura demanda mais ciclos de *clock* para geração de candidatos. Além disso, os vetores de teste utilizados nas simulações foram extraídos do HM durante a codificação da amostra *BQTerrace* para quatro QPs (22, 27, 34 e 37). Também, a síntese foi realizada utilizando a biblioteca *standard-cell* TSMC de 45nm.

A Figura 33 resume as estimativas de área para os experimentos citados. Uma vez que o DC®fornece as estimativas em μm^2, foi feito o cálculo da área em *gate equivalent*, i.e., área total do *hardware* / área de uma porta NAND com duas entradas e capacidade de corrente igual a 1 (0,7056 μm^2). Dessa maneira foi possível manter a unidade de área compatível com a fornecida nos trabalhos correlatos.

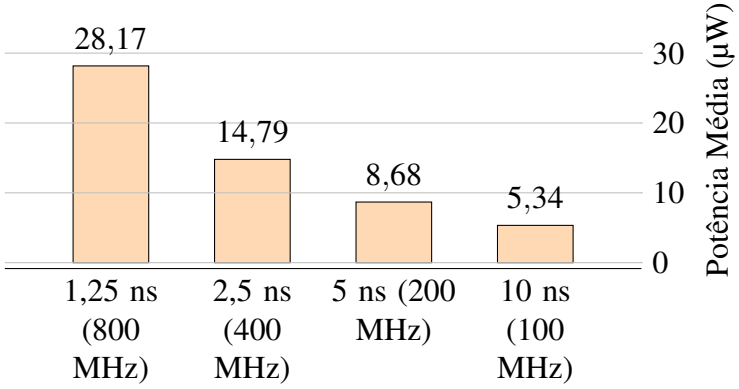
Figura 33 – Resultados das estimativas de área fornecidas pelo DC®. **Gate equivalent*.



Observou-se que as estimativas de área se mantiveram as mesmas para diferentes QPs, mas haviam uma pequena variação dependendo do período *clock*, que pode ser vista na Figura 33. Nesse contexto, nota-se que a área tende a aumentar conforme reduz-se o período do *clock*. Isso acontece devido a seleção de células pelo DC®uma vez que células maiores (que também são mais rápidas) precisam ser utilizadas para cumprir as restrições de tempo do circuito. Essa pequena variação na área indica que a arquitetura está distante da frequência máxima, pois é esperado um aumento considerável na área quando a arquitetura opera próxima à sua frequência máxima, justamente pelo fato de células mais rápidas serem maiores.

Com relação as estimativas de potência, observou-se uma pequena variação dependendo do QP (maior variância foi de, aproximadamente, 0,004). Por conta disso, foi calculada a média simples dessas estimativas. A Figura 34 resumem as estimativas médias de potência por QP.

Figura 34 – Resultados médios das estimativas de potência fornecidas pelo DC®.



Tal como esperado, a medida que o período aumenta em 50%, o consume do potência reduz em, aproximadamente, 50%. Como mostra a Equação (25), a frequência (f) é um dos componentes utilizados para o cálculo da potência dinâmica (P_{dyn}), onde C_L é a *load capacitance*, V_{dd}^2 é a tensão e α é a atividade de chaveamento. Logo, como a frequência é inversamente proporcional ao período, à medida que aumenta-se o período, reduz-se a frequência e, consequentemente, reduz-se a potência.

$$P_{dyn} = C_L * V_{dd}^2 * f * \alpha \tag{25}$$

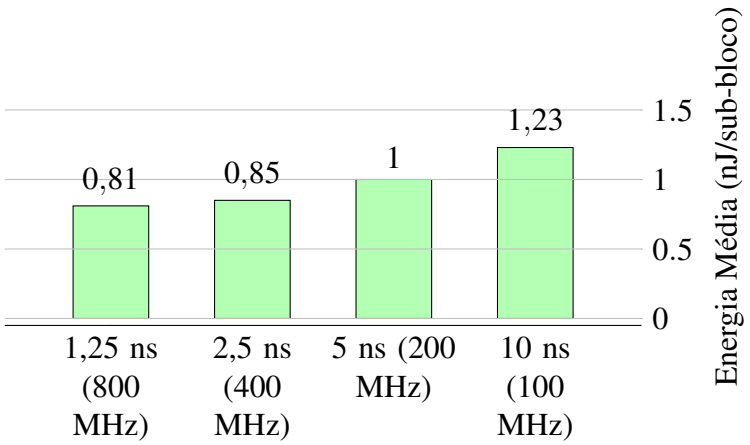
A Figura 35 sintetiza os resultados de consumo de energia por sub-bloco (bloco de tamanho 4×4) para os experimentos mencionados acima. Tal como nas estimativas de potência, também foi realizada a média por QP em decorrência a baixa variação no consumo de energia.

$$\text{Energia(N, M, Período, Potência)} = \text{ciclos(N, M)} * \text{Período} * \text{Potência} * 10^9 \tag{F2}$$

O cálculo da energia pode ser visto na Função (F2), onde: “ciclos” é uma função que retorna o número de ciclos necessários para o preditor do modo “M” construir um candidato de tamanho “N”. Foi considerado nessa função apenas o número de ciclos necessário para a geração de um candidato de tamanho 4×4 com o modo planar, cenário em que a predição demanda maior número de ciclos. Como é possível notar na Figura 35, o consumo de energia aumenta de acordo com o aumento do período. Isso ocorre em virtude da potência reduzir em uma proporção menor do que a proporção em que o período aumenta.

Quanto à aplicação de *clock-gating*, a ferramenta DC®reportou que cerca de 86% dos registradores são *gated-registers*. Logo, a maior parte dos registradores foram otimizados para reduzir a atividade de chaveamento, restando apenas 16% que mantiveram a lógica de atualização não afetada pela técnica. Ainda, foi necessário adicionar 302 unidades de *clock-*

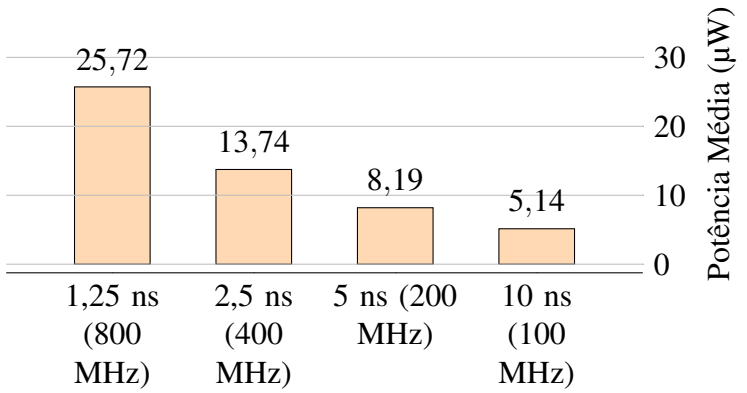
Figura 35 – Resultados médios de consumo de energia por sub-bloco.



gating (portas “AND”, “NAND”, “OR” e “NOR” utilizadas para associar o chaveamento do *clock* ao sinal de *enable* dos registradores).

Além dos experimentos relatados acima, outros foram realizados para avaliar as estimas de área e potência da arquitetura durante o processamento de blocos maiores do que 4×4. Neste caso, o *software* de referência foi executado apenas com QP 22 e foram codificados apenas blocos de tamanho 8×8, 16×16 e 32×32. As Figuras 36 e 37 sintetizam os resultados de potência e energia obtidos através destes experimentos.

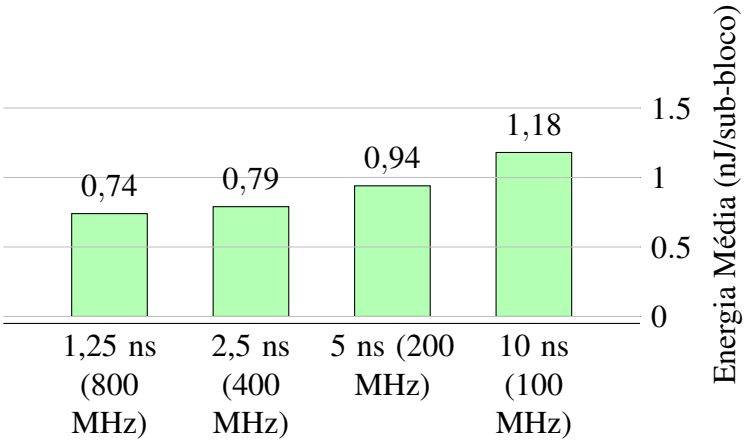
Figura 36 – Resultados médios das estimativas de potência fornecidas pelo DC®.



Tal como nos experimentos anteriores o QP não teve influência sobre as estimativas de área onde, neste caso, os resultados se mantiveram idênticos aos obtidos anteriormente. Também, por ter uma variância pequena em relação ao tamanho de bloco (maior variância foi, aproximadamente, 0,83) foi calculada a média dos valores.

Em virtude dos estados iniciais e finais dos preditores serem executados uma única vez por candidato, gerar quatro candidatos de tamanho 4×4 demanda mais ciclos do que gerar um candidato de tamanho 8×8, por exemplo. Essa característica da arquitetura se reflete cla-

Figura 37 – Resultados médios de consumo de energia por sub-bloco.



ramente nas estimativas de potência e energia mostradas nas Figuras 36 e 37. Processar blocos de tamanho maior reduz a potência e, consequentemente, o consumo de energia por sub-bloco. Ainda, o relatório do DC®em relação ao *clock-gating* foi idêntico ao reportado nos experimentos anteriores.

6.1 COMPARAÇÃO COM TRABALHOS CORRELATOS

A Tabela 6 resume as principais características das arquiteturas propostas em trabalhos correlatos e da proposta neste trabalho. Tal como os demais trabalhos, exceto o de Zhou et al. (2013), a arquitetura proposta implementa a etapa de pós-filtro prevista pelo HEVC. Quanto à decisão de modo, como não foi implementada a busca de candidato, não há necessidade de avaliar o impacto da arquitetura em termos de BD-Rate. Também é importante apontar que poucos dos trabalhos realizam a simulação.

A frequência necessária para gerar candidatos para vídeos em resolução 4K à 60 qps faz com que a arquitetura proposta neste trabalho tenha que operar a 800 MHz. Com relação à frequência da arquitetura proposta neste trabalho, é possível notar que ela opera a uma frequência 1,5 vezes maior se comparada com a proposta por Corrêa (2017). Logo, a arquitetura proposta neste trabalho consome mais energia que o de Corrêa (2017) se ambas funcionarem por um mesmo período de tempo.

Ainda que não tenha sido feita a segunda etapa de simulações e os trabalhos na Tabela 6 tenham sido sintetizados para tecnologias diferentes, nota-se uma grande diferença entre as potências deste trabalho e o de Corrêa (2017). Essa diferença, onde a potência deste trabalho é aproximadamente 13 vezes menor, está relacionada diretamente ao tipo de resolução alvo do trabalho de Corrêa (2017). A arquitetura proposta neste trabalho processa vídeo em resolução 4K (3840×2160) à 60 qps, enquanto que a de Corrêa (2017) processa quadros 8K (7680×4320) à 120 qps. Logo a arquitetura de Corrêa (2017) processa 8 vezes mais dados, já que a resolução

Tabela 6 – Resumo das características de soluções para predição intra quadros do padrão HEVC. “n/a” significa “não se aplica” e “n/d” “não disponibilizado(a)”.

Critérios	**Zhou (2013)**	**Cong Liu (2013)**	**Abramowski (2014)**	**Corrêa (2017)**	**Este trabalho**
pós-filtros	Não	Sim	Sim	Sim	sim
Decisão de Modo	Único Melhor SAD	Único Melhor SATD	n/d	Oito Melhores SADs	não
BD-Rate	n/d	10%	n/d	0.17%	n/a
Simulação	n/d	n/d	Sim	Não	Sim
Tecnologia	TSMC 130 nm	TSMC 65 nm	TSMC 130 nm	*NanGate* 45 nm	TSMC 45 nm
Área (k ***gates***)	324	77	127	4952	260
Frequência	400 MHz	600 MHz	200 MHz	529 MHz	800 MHz
Potência	n/d	n/d	n/d	363,23 mW	26,46 mW
Taxa de Proces.	HD 1080p 60 qps	HD 1080p 30 qps	HD 1080p 35 qps	UHD 8K 120 qps	UHD 4K 60 qps

8K possui 4 vezes mais dados que a 4K e a taxa de 120 qps tem o dobro de quadros que a de 60 qps. Levando essas diferenças em consideração, a arquitetura proposta neste trabalho teria uma potência 8 vezes maior, i.e., uma potência de 225,36 µW. Com isso, é possível dizer que a arquitetura proposta neste trabalho possui, aproximadamente, 62% da potência da arquitetura de Corrêa (2017). Ainda, essa diferença não leva em consideração que este trabalho não realiza a busca de melhor candidato que certamente aumentaria a potência.

7 CONCLUSÃO

Neste trabalho foi explicado sobre o processo de particionamento de blocos do padrão HEVC, uma vez que nas etapas do fluxo de codificação híbrida são aplicadas técnicas e algoritmos nos blocos dos quadros. Foi apresentado o fluxo de codificação híbrido e explicada a principal função de cada uma das etapas deste fluxo. Uma vez que este trabalho se baseia na predição intra quadros do padrão de codificação HEVC, esta foi apresentada em detalhes. Mostrou-se que a predição intra quadros do HEVC é composta por três principais etapas, sendo elas: pré-processamento de amostras de referência, uma etapa opcional onde é realizado o tratamento inicial das amostras utilizadas na predição; predição de amostras, onde o padrão de codificação define três principais tipos de preditores, cada qual para capturar uma característica presente em vídeos; por fim, foi apresentada a etapa de pós-processamento de amostras preditas, outra etapa opcional onde é aplicado um filtro para suavizar transições de blocos.

No Capítulo 4 foi apresentado o projeto e implementação do bloco acelerador em *hardware* para a geração dos blocos candidatos da predição intra quadros do padrão de codificação HEVC. Considerando os objetivos deste trabalho, descritos na Seção 1.1, obteve-se êxito quanto ao projeto e implementação. Quanto da avaliação, alguns problemas impediram a extração das estimativas de área e potência com alta precisão. A principal característica que diferencia este trabalho dos correlatos é justamente a capacidade de compor qualquer tamanho de bloco a partir de sub-blocos, i.e., blocos de tamanho 4×4. Para isso, lógicas de controle foram elaboradas para ser possível a predição de blocos maiores do que 4×4, dada que a predição de uma amostra depende de uma série de quesitos (e.g. tamanho do bloco, modo de predição e posição da amostra sendo predita).

Finalmente, no Capítulo 6 foram realizadas análises acerca do acelerador em *hardware* proposto neste trabalho. Verificou-se que a arquitetura proposta neste trabalho consome cerca de 62% da potência em relação a solução apresentada por Corrêa (2017) ao custo de um aumento de 1,5 vezes na frequência de operação. As comparações entre os trabalhos citados no Capítulo 3 não são muito justas, já que cada um foi sintetizado por ferramentas diferentes e para tecnologias diferentes. Ainda assim, a arquitetura aqui proposta apresentou resultados similares aos presentes nos trabalhos da literatura.

7.1 TRABALHOS FUTUROS

Realizar as etapas em laranja no fluxo apresentado na Figura 31 permitiam melhorar as estimativas de área e potência. Nesse sentido, ainda que demande bastante tempo, outra forma de melhorar a avaliação da arquitetura proposta seria a realização de simulações com diferentes amostras de vídeo da CTC (neste trabalho, cada uma das 28 simulações durou cerca de 4 horas).

Embora a arquitetura tenha sido planejada para compor blocos maiores a partir dos blocos de tamanho 4×4, algumas lógicas de controle comuns aos preditores não foram abstraídas. Um possível trabalho futuro seria justamente explorar otimizações que poderiam ser realizadas

na arquitetura proposta e, dessa forma melhorar seu desempenho. Junto com as otimizações, poderia ser realizada a implementação de um módulo de decisão (busca do melhor candidato), permitindo avaliar qual o impacto da lógica de decisão de modo.

Ainda em relação à avaliação, pode ser feito uma análise da frequência máxima da arquitetura, podendo até definir outras resoluções alvos das quais a arquitetura seja capaz de processar.

Também, um possível trabalho seria explorar a estratégia de geração dos candidatos através de sub-blocos no futuro padrão de codificação estado-da-arte, conhecido como *Versatile Video Coding* (VVC).

REFERÊNCIAS

Abramowski, A.; Pastuszak, G. A double-path intra prediction architecture for the hardware h.265/hevc encoder. In: **17th International Symposium on Design and Diagnostics of Electronic Circuits Systems**. [S.l.: s.n.], 2014. p. 27–32.

Bonotto, B. et al. A Named-Pipe Library for Hardware Simulation. In: **SIM2018**. [S.l.]: SBC, 2018.

Bossen, F. **Common Test Conditions and software reference configurations**. Shanghai, 2012.

CISCO. **VNI Complete Forecast Highlights: Global - 2016 Year in Review**. [S.l.], 2017.

Conti, M. et al. The dark side(-channel) of mobile devices: A survey on network traffic analysis. **IEEE Communications Surveys Tutorials**, v. 20, n. 4, p. 2658–2713, 10 2018. ISSN 1553-877X.

Corrêa, M. et al. High-throughput hevc intrapicture prediction hardware design targeting uhd 8k videos. In: **2017 IEEE International Symposium on Circuits and Systems (ISCAS)**. [S.l.: s.n.], 2017. p. 1–4.

CORRêA, M. M.

Desenvolvimento arquitetural para predição intraquadro do padrão HEVC de codificação de vídeos — Dissertação de Mestrado. Universidade Católica de Pelotas, 2017.

DONNO, M. et al. Clock-tree power optimization based on rtl clock-gating. In: **Proceedings of the 40th Annual Design Automation Conference**. New York, NY, USA: ACM, 2003. (DAC '03), p. 622–627. ISBN 1-58113-688-9.

Herglotz, C.; Springer, D.; Kaup, A. Modeling the energy consumption of hevc p- and b-frame decoding. In: **2014 IEEE International Conference on Image Processing (ICIP)**. [S.l.: s.n.], 2014. p. 3661–3665.

Keating, M. et al. **Low Power Methodology Manual for System-on-Chip Design**. New York, USA: Springer Science+Business Media, Inc, 2002.

Kim, C.; Shih, H.-H.; Kuo, C.-C. J. Fast h.264 intra-prediction mode selection using joint spatial and transform domain features. **Journal of Visual Communication and Image Representation**, v. 17, n. 2, p. 291 – 310, 2006. ISSN 1047-3203.

Kim, I. et al. Block partitioning structure in the hevc standard. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 22, n. 12, p. 1697–1706, 12 2012. ISSN 1051-8215.

Lainema, J. et al. Intra coding of the hevc standard. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 22, n. 12, p. 1792–1801, 12 2012. ISSN 1051-8215.

Ling, N. High efficiency video coding and its 3d extension: A research perspective. In: **2012 7th IEEE Conference on Industrial Electronics and Applications (ICIEA)**. [S.l.: s.n.], 2012. p. 2150–2155. ISSN 2156-2318.

Liu, C. et al. A highly pipelined vlsi architecture for all modes and block sizes intra prediction in hevc encoder. In: **2013 IEEE 10th International Conference on ASIC**. [S.l.: s.n.], 2013. p. 1–4. ISSN 2162-755X.

Qing Wu; Pedram, M.; Xunwei Wu. Clock-gating and its application to low power design of sequential circuits. **IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications**, v. 47, n. 3, p. 415–420, 3 2000.

Seidel, I. et al. Energy-efficient hadamard-based satd hardware architectures through calculation reuse. **IEEE Transactions on Circuits and Systems I: Regular Papers**, v. 66, n. 6, p. 2102–2115, 6 2019.

Silveira, B. et al. Power-Efficient Sum of Absolute Differences Hardware Architecture Using Adder Compressors for Integer Motion Estimation Design. **IEEE Transections Circuits Systems I, Regular Papers**, v. 64, n. 12, p. 3126–3137, 12 2017. ISSN 1549-8328.

SULLIVAN, G. et al. Overview of the high efficiency video coding (HEVC) standard. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 22, n. 12, p. 1649–1668, 12 2012.

Sze, V. et al. **High Efficiency Video Coding (HEVC)**: Algorithms and Architectures. US: Springer Publishing Company, 2014.

Thomas, D.; Moorby, P. **The Verilog®Hardware Description Language**. New York, USA: Springer Science+Business Media, Inc, 2010.

Wiegand, T.; Girod, B. Lagrange multiplier selection in hybrid video coder control. In: **Proceedings 2001 International Conference on Image Processing (Cat. No.01CH37205)**. [S.l.: s.n.], 2001. v. 3, p. 542–545 vol.3.

Yang, F.; Wan, S.; Izquierdo, E. Lagrange multiplier selection for 3-d wavelet based scalable video coding. In: **2007 IEEE International Conference on Image Processing**. [S.l.: s.n.], 2007. v. 2. ISSN 2381-8549.

Zhou, N. et al. On hardware architecture and processing order of hevc intra prediction module. In: **2013 Picture Coding Symposium (PCS)**. [S.l.: s.n.], 2013. p. 101–104.

ANEXO A – ARTIGO CIENTÍFICO DO TCC

A seguir será apresentado o artigo científico elaborado com base no conteúdo desta monografia.

Bloco Acelerador em Hardware para Predição Intra Quadros do Padrão HEVC

Bruno Bonotto[1], **Ismael Seidel**[1], **José Luís Güntzel**[1]

[1]Departamento de Informática e Estatística
Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brasil

Abstract. *O aumento na utilização de aplicações com streaming de vídeo demanda a criação de novas técnicas de compressão de vídeo para viabilizar a transmissão via rede. A compressão de vídeo é indispensável quando se trata do armazenamento desse tipo de dado, já estes chegam a ocupar até dezenas de gigabytes por minuto. Para aumentar a taxa de compressão, codificadores de padrões do estado-da-arte apresentam complexidade 10 vezes maiores que aqueles de padrões anteriores. Com o objetivo de viabilizar a compressão em dispositivos móveis operados à bateria, as etapas mais complexas do codificador devem ser executadas em hardware dedicado que, por sua vez, deve ser projetado para ser energeticamente eficiente. Este trabalho apresenta o projeto, descrição e avaliação de um bloco acelerador em hardware otimizado para realizar o processo de predição intra quadros para o padrão de codificação High Efficiency Video Coding (HEVC). O acelerador proposto possui uma frequência de operação de 800* MHz*, sendo capaz de processar vídeos em resolução 4K com 60 quadros por segundo (qps) à um consumo médio de potência de 28,17* mW.

Palavras-chave: *Codificação de vídeo digital, HEVC, Predição intra quadros, Arquitetura de hardware.*

1. Introducão

As últimas décadas têm sido marcadas pela crescente demanda por dispositivos móveis (e.g. *smartphones* e *notebooks*). [Conti et al. 2018] constatou que em 2015 o número de usuários de *smartphones* e *tablets* correspondiam a cerca de 25% e 14% da população mundial, respectivamente, podendo aumentar para 37% e 19% até 2020. A popularização desses dispositivos deve-se à maior facilidade de acesso à internet e a grande variedade de aplicações disponíveis para esses dispositivos [Conti et al. 2018]. Dentre as aplicações mais populares encontram-se aquelas que permitem a captura, armazenamento e transmissão via rede (*streaming*) de arquivos de vídeo. O uso em massa dessas aplicações ajuda a entender o fato de que, em 2016, aproximadamente 73% de todo o tráfego na internet estava relacionado a vídeos, podendo aumentar para 83% em 2021 [Cisco 2017].

Além da ampla demanda por aplicações multimídia, existe uma crescente demanda por resoluções maiores [Ling 2012], fazendo com que seja necessário um volume de dados maior para representar vídeos em tais resoluções. Por exemplo, um vídeo sem compressão de 30 minutos, amostragem de 30 qps, 3 *bytes* para representar cada *pixel* e resolução *Super Video Graphics Array* (SVGA) teria, aproximadamente, 77,8 *gigabytes*

de dados. Aumentando a resolução para *Full* HD e mantendo-se as demais configurações, o mesmo vídeo necessitaria quase 4,32 vezes mais dados.

Tal volume dificulta o armazenamento de vídeos digitais sem compressão, bem como a transmissão deles, já que seria necessário uma banda de 1,5 *gigabits* por segundo para transmitir vídeos em resolução *Full* HD, por exemplo.

Assim, a demanda por resoluções cada vez maiores faz com que aumente, também, a demanda por novas ferramentas de compressão. Por outro lado, a introdução de novas ferramentas também aumenta a complexidade computacional dos codificadores. O atual padrão de codificação de vídeo digital HEVC, por exemplo, embora permita uma compressão até 50% maior que seu antecessor (*Advanced Video Coding* (AVC)) mantendo a mesma qualidade visual, teve também um aumento estimado de 2 a 10 vezes em sua complexidade [Sullivan et al. 2012]. Ainda, a demanda por codificação em tempo real dificulta que esta possa ser realizada em *software* executando sobre um processador de propósito geral, fazendo com que as etapas mais complexas sejam executadas em *hardware*. Estes, por sua vez, precisam ser energeticamente eficientes, uma vez que o codificador pode ser embarcado em dispositivos móveis operados à bateria [Herglotz et al. 2014].

As etapas executadas na codificação buscam explorar redundâncias existentes neste tipo de arquivo, tais como: espacial (intra quadro), existente entre regiões semelhantes dentro de um mesmo quadro; temporal (inter quadros), existente entre regiões semelhantes entre quadros próximos no tempo; e entrópica, relacionada com a representação binária da informação. Dentre as etapas de compressão de um codificador, as predições intra (foco deste trabalho) e inter quadros são responsáveis por reduzir redundâncias espaciais e temporais, respectivamente.

O restante deste artigo está organizado da forma a seguir. No Capítulo 2 será explicada a predição intra quadros do padrão HEVC. No Capítulo 3 será apresentado o bloco acelerador em *hardware* proposto neste trabalho. No Capítulo 4 será apresentado o método de avaliação utilizado neste trabalho. Por fim, nos Capítulos 5 e 6 serão apresentados os resultados conclusões deste trabalho, respectivamente.

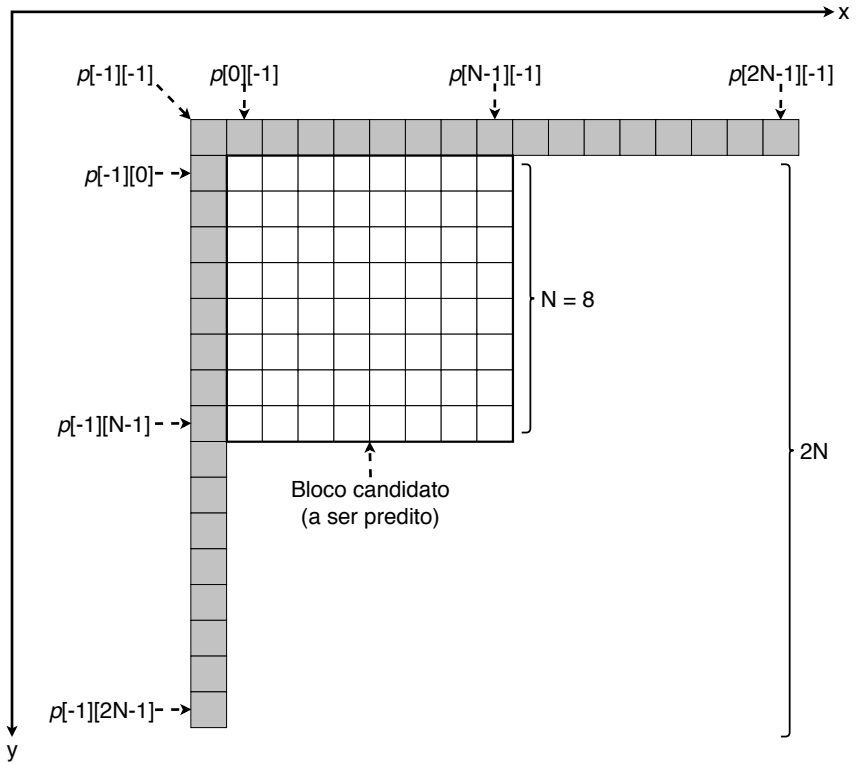
2. Conceitos Básicos

No padrão de codificação HEVC, a predição intra quadros é realizada em quatro principais etapas: pré-processamento de amostras de referência, onde estas são preparadas para a realização da predição; predição de amostras, onde blocos candidatos serão gerados para predizer o bloco original; pós-processamento de amostras preditas, onde os blocos candidatos serão filtrados para gerar imagens mais; busca do melhor candidato, onde será realizada a seleção do bloco candidato mais parecido com o bloco original. Cada umas dessas etapas será melhor explicada a seguir.

2.1. Pré-processamento de amostras de referência

A predição intra quadros de um bloco original de tamanho N$\times$N, onde N $\in \{4, 8, 16, 32\}$, é realizada utilizando-se amostras de referência de fora do bloco. A Figura 1 ilustra o número máximo de amostras de referência que podem ser utilizadas para a construção de um bloco candidato de tamanho N=8 (4N+1).

Figura 1. Blocos em cinza representam as amostra de referência *p*[x][y] para um bloco de tamanho N=8.



Fonte: adaptada de [Sze et al. 2014].

Note que, nem sempre todas as amostras de referência estarão disponíveis. Isso acontece, por exemplo, sempre que for realizada a predição do primeiro bloco de cada quadro original. Para esses casos particulares, o padrão define duas formas para preencher as amostras indisponíveis: quando nenhuma amostra estiver disponível todas são preenchidas com o valor nominal médio (2^{n-1}) de acordo com o número de *bits* (n) por amostra; quando ao menos uma amostra esta disponível, as demais são preenchidas copiando o valor das amostras válidas.

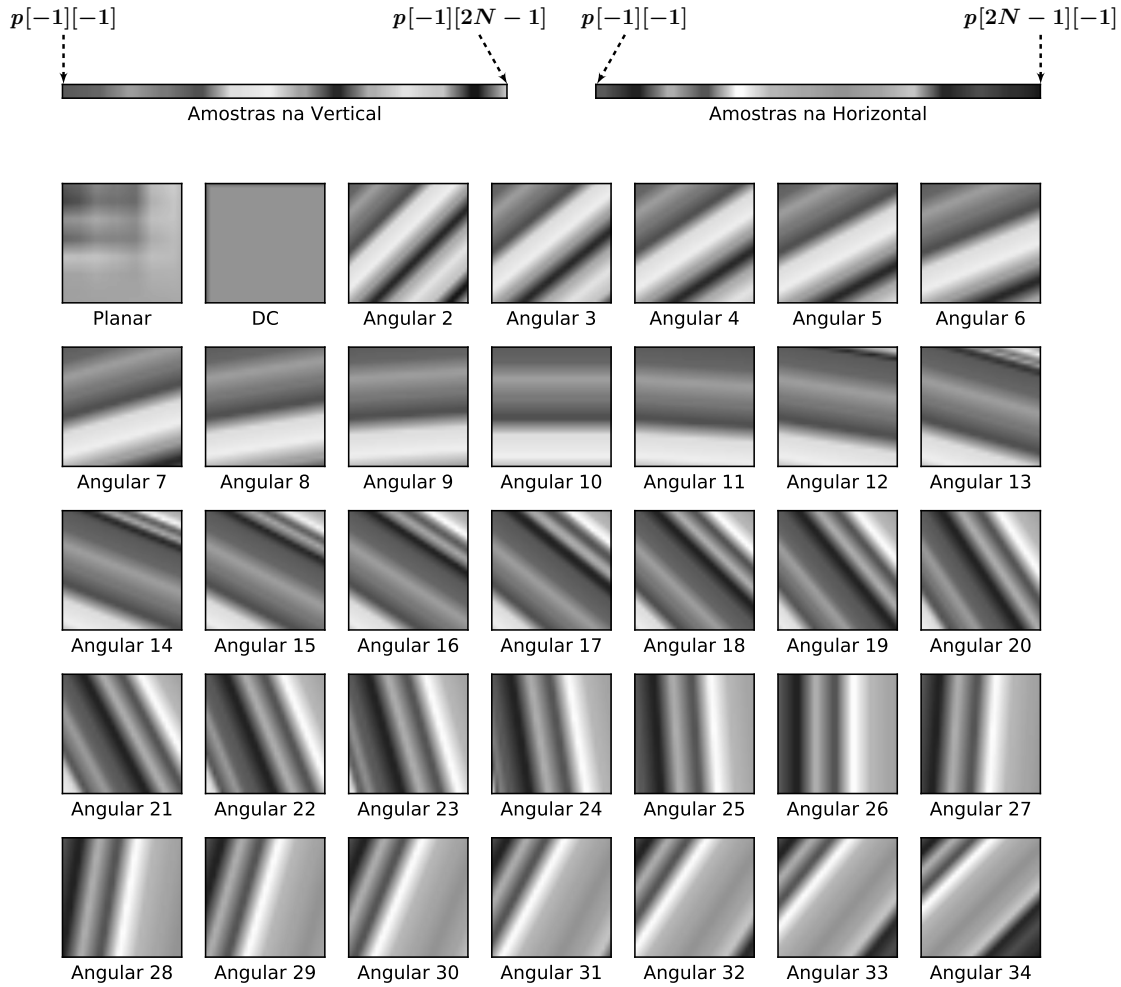
Após tornar disponíveis todas as amostras de referência, o padrão define uma etapa opcional de filtros que podem ser utilizados para suavizar a transição entre blocos vizinhos, i.e., evitar bordas indesejadas.

2.2. Predição de amostras

O processo de predição de um bloco pode ser realizado através de 35 modos distintos, onde cada um permite gerar candidatos com conteúdos diferenciados, tipicamente presentes em imagens [Sze et al. 2014]. O modo 0, denominado planar, é utilizado para predizer blocos com conteúdo contínuo mas com suaves transições. Já o modo 1, denominado DC, permite predizer blocos com conteúdo neutro, i.e., com nenhuma variação. Os demais modos, denominados angulares, foram elaborados para predizer blocos com diferentes características direcionais e, por conta disso, os modos 2 a 17 são também chamados de horizontais e os restantes de verticais. A Figura 2 ilustra a geração dos 35 blocos candidatos de acordo com os modos de predição intra para um bloco original de

tamanho 32×32.

Figura 2. 35 blocos candidatos gerados a partir da predição intra quadros para um bloco original de tamanho N=32. No topo estão as amostras de referência utilizadas na construção dos candidatos.



Fonte: o autor.

2.3. Pós-processamento de amostras preditas

Visando suavizar descontinuidades, o padrão define filtros opcionais que podem ser aplicados aos blocos candidatos após a sua construção. Esses casos são mais visíveis nas bordas superior e esquerda do modo DC e nas bordas esquerda e superior dos modos diretamente vertical e horizontal, respectivamente. Caso optado pelo uso dos filtros pós-predição, sua aplicação será decidida com base no modo de predição e no tamanho do bloco. Os filtros são aplicados apenas para o modo de predição DC (em todos os tamanhos de bloco, exceto 32×32) ou para os modos diretamente horizontal e vertical (em todos os tamanhos de bloco).

2.4. Busca do melhor candidato

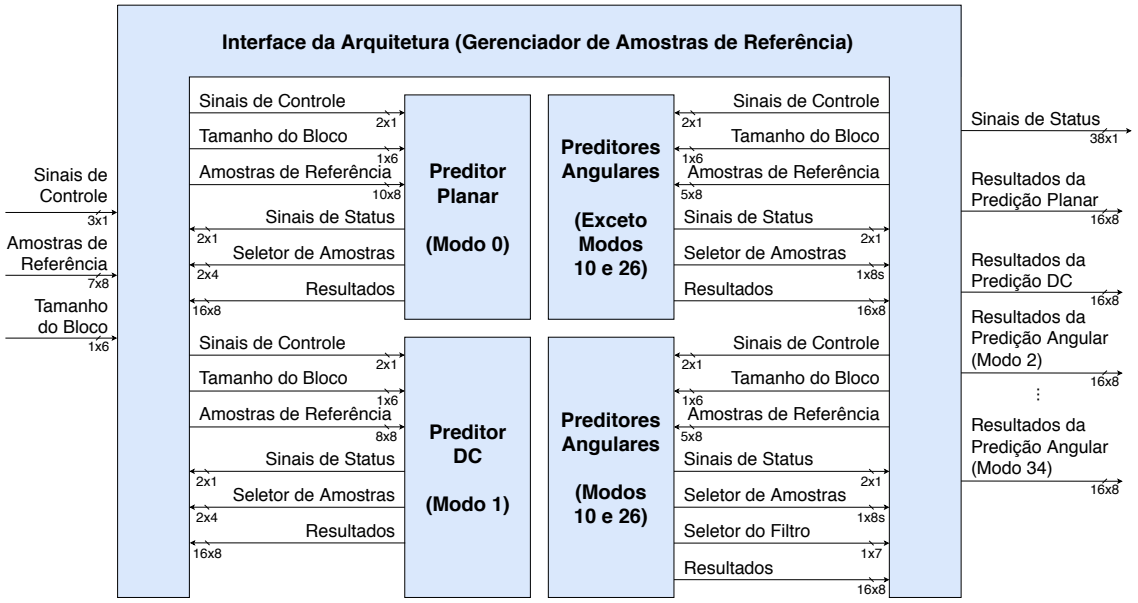
Nessa etapa é realizada a seleção do melhor candidato, i.e., aquele que for mais parecido com o bloco original. As formas mais comuns de avaliar os candidatos consiste

em estimar o custo dos modos utilizando métricas de distorção (e.g. *Sum of Absolute Differences* (SAD) e *Sum of Absolute Transformed Differences* (SATD)).

3. Proposta

Neste trabalho, a arquitetura foi elaborada para construir candidatos através da composição de blocos de tamanho 4×4 (por convenção serão chamados de sub-blocos). Optou-se pela construção de preditores específicos para cada modo de predição e genéricos para tamanhos de blocos. A Figura 3 ilustra o diagrama de blocos da arquitetura proposta neste trabalho.

Figura 3. Diagrama de blocos da arquitetura proposta.



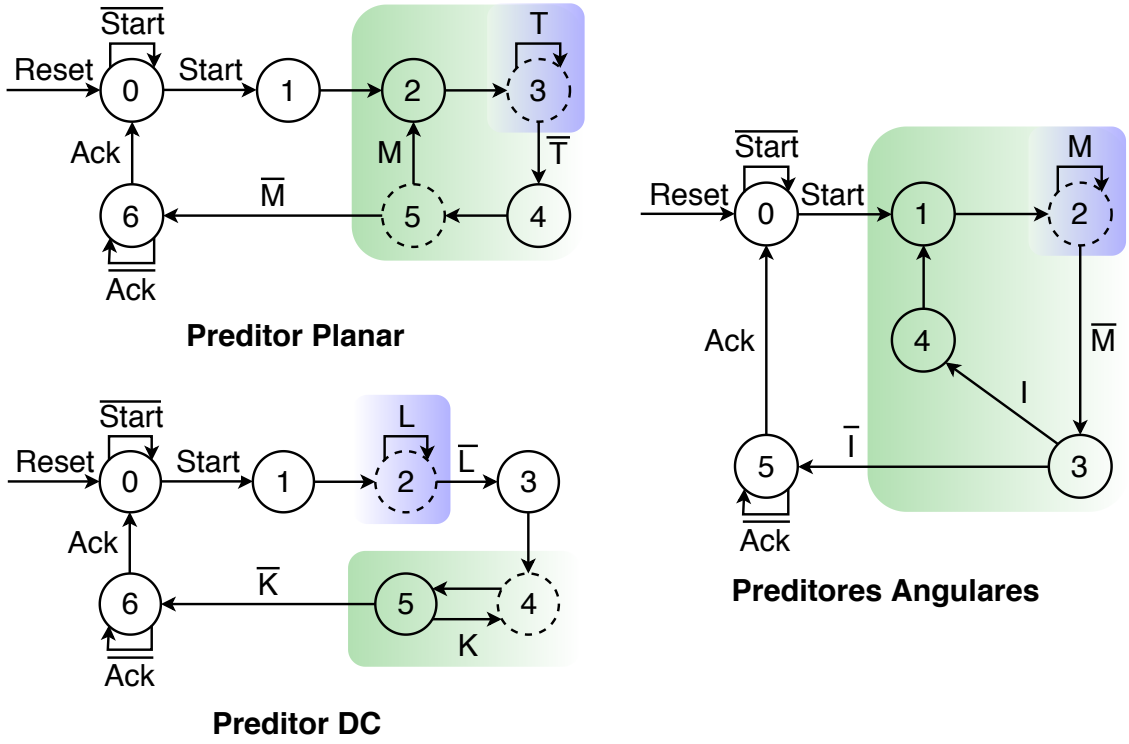
Fonte: o autor.

Como mostra a Figura 3, foram descritos cinco módulos em *hardware*: um responsável pela leitura e distribuição das amostras de referência, que também é a interface do acelerador; dois preditores específicos para os modos planar e DC; dois preditores para os modos angulares, onde um é específico para a predição com os modos diretamente horizontal e vertical e o outro para os demais modos angulares.

O módulo de gerência de amostras é fundamental para o correto funcionamento dos preditores. A leitura é feita de 8 em 8 amostras (4 horizontais e 4 verticais) sendo necessário um total de 131 registradores de amostras, onde: 128 são para amostras horizontais e verticais, 1 para a posição $(-1,-1)$ e 2 de cópia das amostras (N,-1) e (-1,N). Para blocos de tamanho 4×4, 8×8, 16×16 e 32×32 são necessários 2, 4, 8 e 16 ciclos para a leitura de amostras, respectivamente.

A lógica de construção dos candidatos reflete-se diretamente nas máquinas de estado dos preditores. Como pode ser visto na Figura 4, todas as máquinas de estados possui um *loop* para a construção de um sub-bloco (ilustrado em azul). Também, existe um *loop* maior para construção de todos os sub-blocos necessários para a compor o candidato de tamanho maior do que 4×4 (ilustrado em verde).

Figura 4. Máquinas de estados simplificadas dos preditores.



Fonte: o autor.

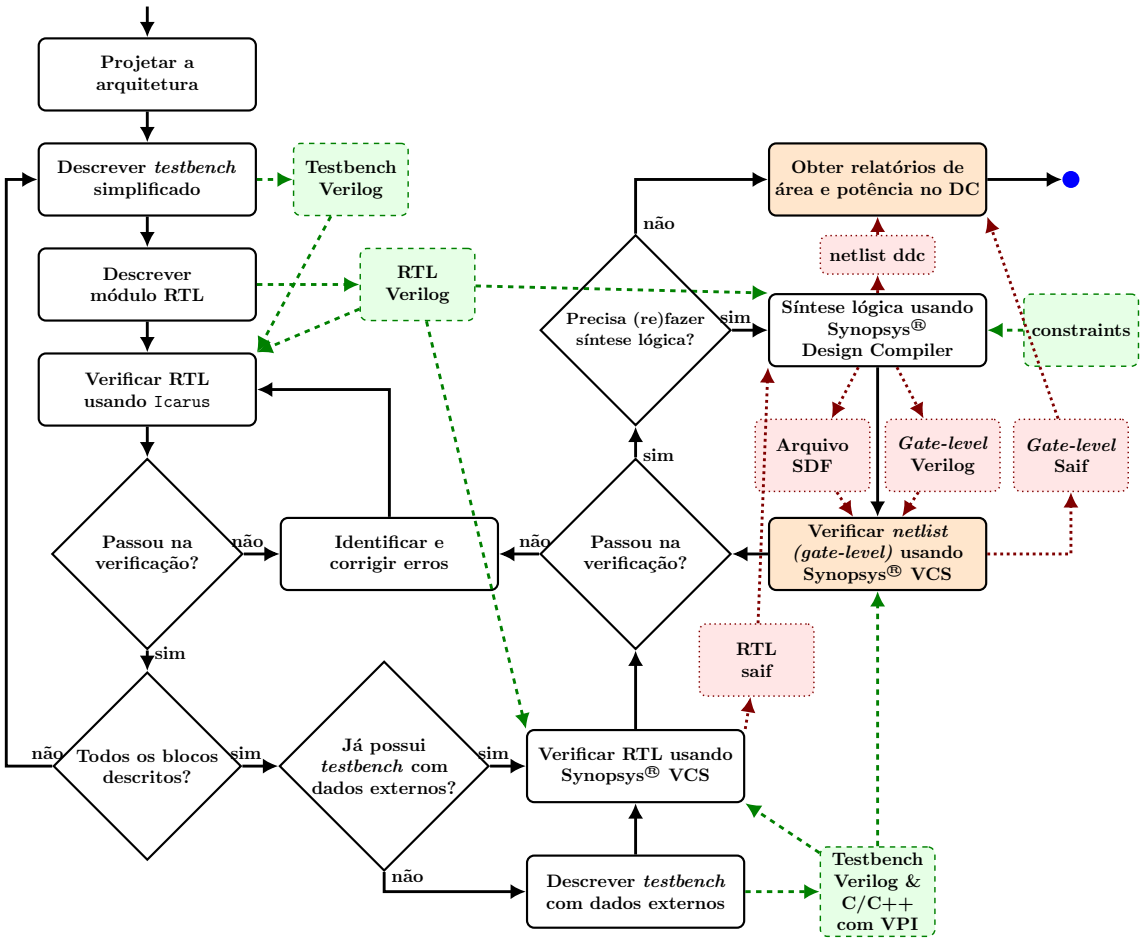
Além dos *loops* onde são construídos os sub-blocos, no estado inicial o sinal "*Start*" determina se os preditores permanecem neste estado, onde todos os registradores recebem o valor zero, ou se iniciam a predição. Nos estados 1 são realizadas a leitura do tamanho do bloco e das amostras necessárias para a construção do sub-bloco e no estado final é sinalizado o fim da construção de um candidato. Finalmente, quando todos os sub-blocos foram construídos os preditores vão ao estado final e permanece nele enquanto os sinais "Ack" ou "*Reset*" não forem ativados.

4. Método

O desenvolvimento deste trabalho foi realizado baseando-se no método utilizado por [Seidel et al. 2019]. A Figura 5 ilustra cada uma das etapas desse método de trabalho, as quais foram de suma importância para o desenvolvimento e avaliação realista da arquitetura proposta neste trabalho. Na Figura 5, os arquivos em verde foram criados pelo projetista e os vermelhos foram gerados pelas ferramentas comerciais. Também, as etapas em laranja não foram realizadas neste trabalho, onde foram consideradas como possíveis trabalhos futuros.

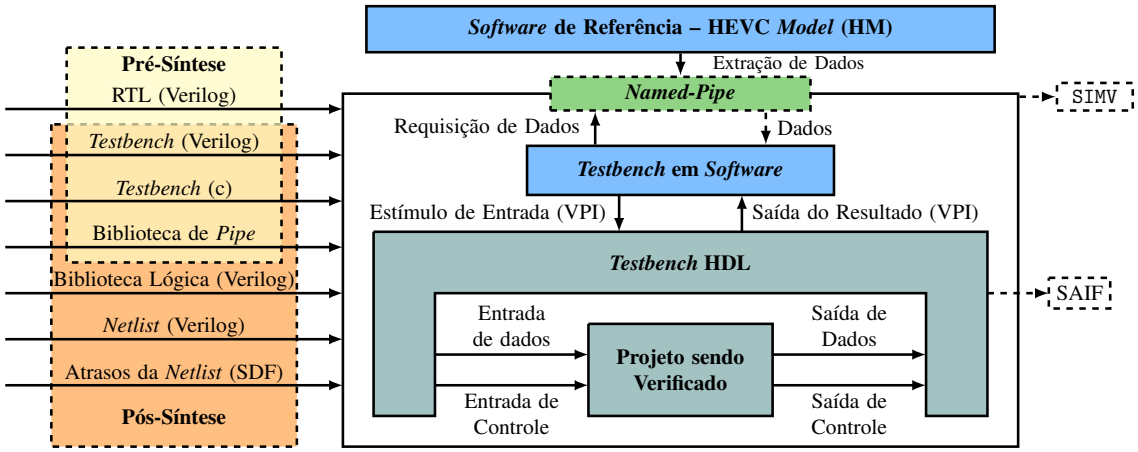
Na etapa "Projetar a arquitetura" é onde a arquitetura de *hardware* proposta foi especificada detalhadamente. As etapas "Descrever *testbench* simplificado e módulo RTL", "Verificar RTL usando Icarus" e "Identificar e corrigir erros" correspondem às etapas de descrição HDL dos módulos em *hardware* e teste simplificado. Na etapa "Descrever *testbench* com dados externos" é implementado o *testbench* que forneçe dados realistas à arquitetura. Neste trabalho foi utilizado o *framework* proposto por [Bonotto et al. 2018], tal como ilustra a Figura 6.

Figura 5. Método de síntese e projeto.



Fonte: adaptado de [Seidel et al. 2019].

Figura 6. *Framework* proposto por [Bonotto et al. 2018].



Fonte: adaptada de [Seidel et al. 2019].

A etapas "Verificar RTL usando Synopsys®VCS" e "Verificar *netlist* (*gate-level*) usando *Synopsys*®VCS" são utilizadas para realziar a verificação funcional da arquitetura, uma em nível *Register-Transfer Level* (RTL) outra em nível *netlist*, respectivamente.

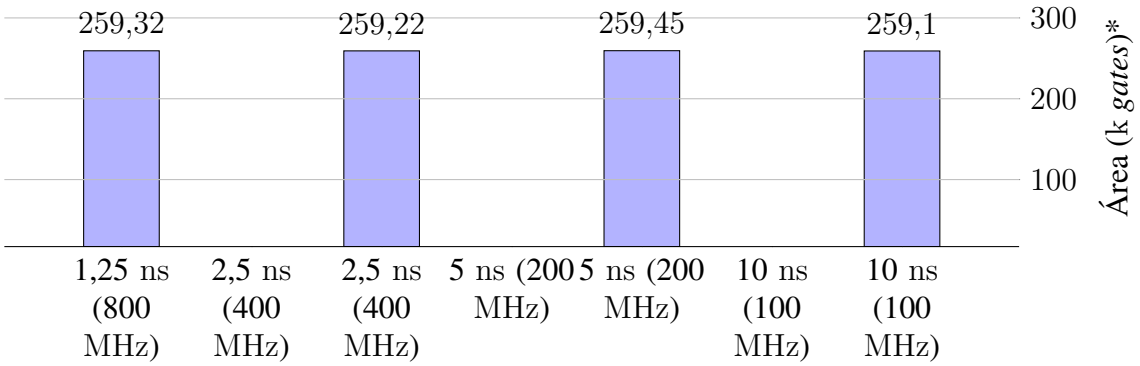
Finalmente, as etapas “Síntese lógica usando Synopsys®DC” e “Obter relatórios de área e potência no Synopsys®DC” são utilizadas para a obtenção de estimativas de área e potência.

5. Resultados

A arquitetura proposta foi simulada para quatro períodos de *clock*. Cada período foi definido para que a arquitetura fosse capaz de gerar os blocos candidatos para uma dada resolução e qps. Para a verificação e validação da arquitetura, foi considerado apenas o processamento de blocos de tamanho 4×4. Além disso, os vetores de teste utilizados nas simulações foram extraídos do *software* de referência do HEVC, chamado HEVC *Model* (HM) durante a codificação da amostra *BQTerrace* para quatro *Quantization Parameter* (QP)s (22, 27, 34 e 37). A síntese foi realizada utilizando a biblioteca *standard-cell* TSMC de 45nm.

A Figura 7 resume as estimativas de área para os experimentos citados.

Figura 7. Resultados das estimativas de área fornecidas pelo *Synopsys®Design Compiler®* (DC®).



A área tende a aumentar conforme reduz-se o período do *clock* devido a seleção de células pelo DC® uma vez que células maiores (que também são mais rápidas) precisam ser utilizadas para cumprir as restrições de tempo do circuito.

Com relação as estimativas de potência, a Figura 8 resume as estimativas médias de potência por QP.

Tal como esperado, a medida que o período aumenta em 50%, o consume do potência reduz em, aproximadamente, 50%. Como mostra a Equação (1), a frequência (f) é um dos componentes utilizados para o cálculo da potência dinâmica (P_{dyn}), onde C_L é a *load capacitance*, V_{dd}^2 é a tensão e α é a atividade de chaveamento. Logo, como a frequência é inversamente proporcional ao período, à medida que aumenta-se o período, reduz-se a frequência e, consequentemente, reduz-se a potência.

$$P_{dyn} = C_L * V_{dd}^2 * f * \alpha \quad (1)$$

A Figura 9 sintetiza os resultados de consumo de energia por sub-bloco (bloco de tamanho 4×4) para os experimentos mencionados acima.

Figura 8. Resultados médios das estimativas de potência fornecidas pelo DC®.

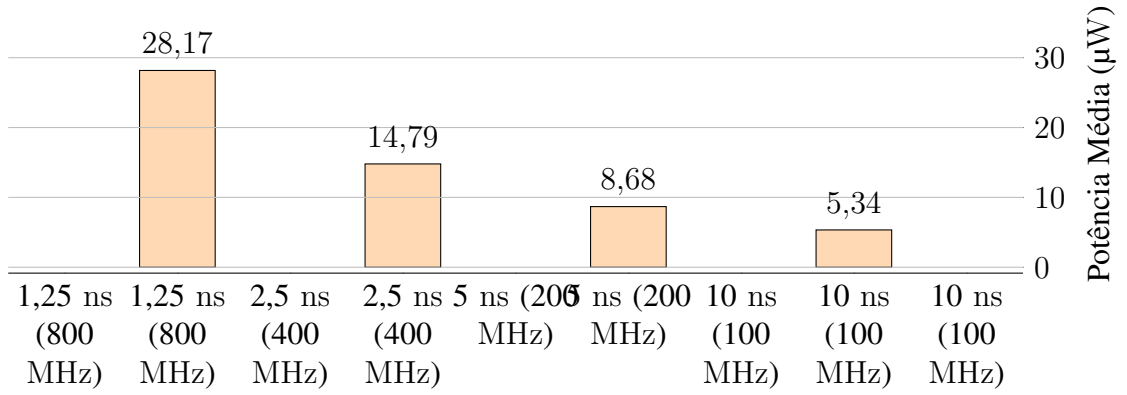
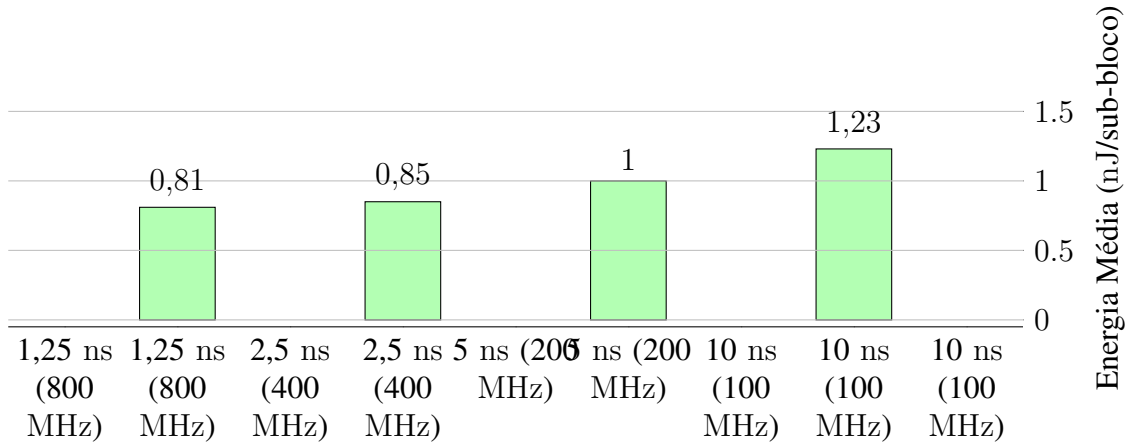


Figura 9. Resultados médios de consumo de energia por sub-bloco.



Como é possível notar na Figura 9, o consumo de energia aumenta de acordo com o aumento do período. Isso ocorre em virtude da potência reduzir em uma proporção menor do que a proporção em que o período aumenta.

Quanto à aplicação de *clock-gating*, a ferramenta DC® reportou que cerca de 86% dos registradores são *gated-registers*. Ainda, foi necessário adicionar 302 unidades de *clock-gating* (portas "AND", "NAND", "OR" e "NOR" utilizadas para associar o chaveamento do *clock* ao sinal de *enable* dos registradores).

Além dos experimentos relatados acima, outros foram realizados com o processamento de blocos maiores do que 4×4 e apenas QP 22. As Figuras 10 e 11 sintetizam os resultados obtidos.

Em virtude dos estados iniciais e finais dos preditores serem executados uma única vez por candidato, gerar quatro candidatos de tamanho 4×4 demanda mais ciclos do que gerar um candidato de tamanho 8×8, por exemplo. Essa característica da arquitetura se reflete claramente nas estimativas de potência e energia mostradas nas Figuras 10 e 11. Processar blocos de tamanho maior reduz a potência e, consequentemente, o consumo de energia por sub-bloco. Ainda, o relatório do DC® em relação ao *clock-gating* foi idêntico

Figura 10. Resultados médios das estimativas de potência fornecidas pelo DC®.

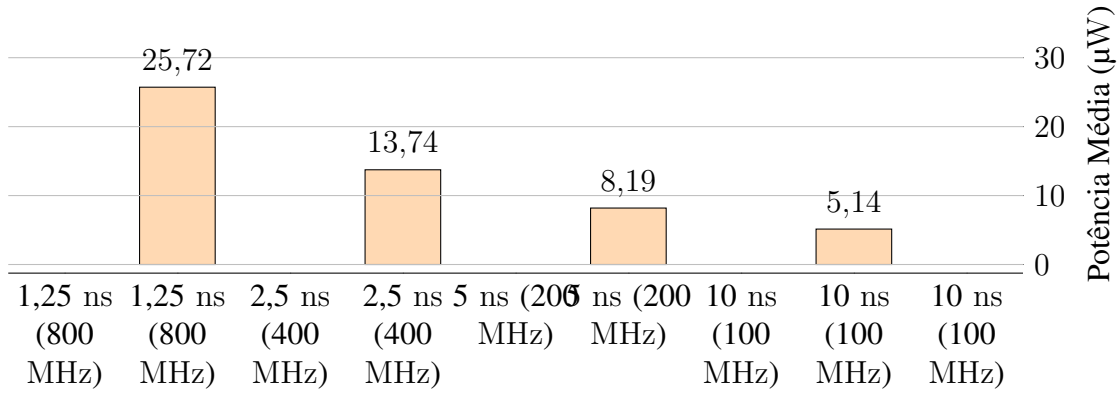
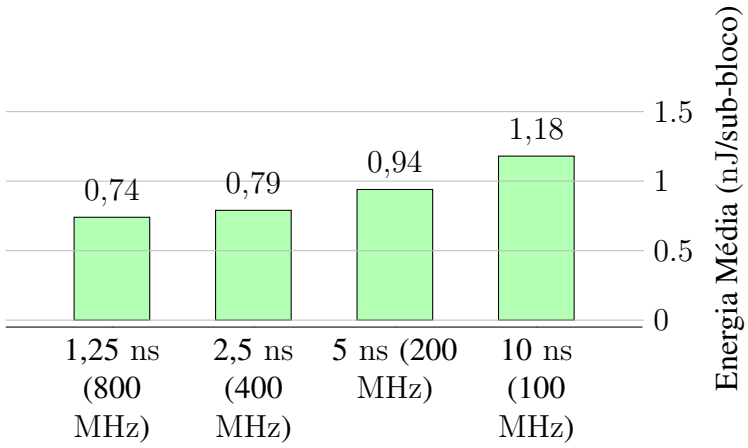


Figura 11. Resultados médios de consumo de energia por sub-bloco.



ao reportado nos experimentos anteriores.

5.1. Comparação com trabalhos correlatos

A Tabela 1 resume as principais características das arquiteturas propostas em trabalhos correlatos e da proposta neste trabalho. Tal como os demais trabalhos, exceto o de [Zhou et al. 2013], a arquitetura proposta implementa a etapa de pós-filtro prevista pelo HEVC. Quanto à decisão de modo, como não foi implementada a busca de candidato, não há necessidade de avaliar o impacto da arquitetura em termos de *Bjøntegaard Delta Rate* (BD-Rate). Também é importante apontar que poucos dos trabalhos realizam a simulação.

A frequência necessária para gerar candidatos para vídeos em resolução 4K à 60 qps faz com que a arquitetura proposta neste trabalho tenha que operar a 800 MHz. Ainda que não tenha sido feita a segunda etapa de simulações e os trabalhos na Tabela 1 tenham sido sintetizados para tecnologias diferentes, nota-se uma grande diferença entre as potências deste trabalho e o de [Corrêa 2017]. Essa diferença, onde a potência deste trabalho é aproximadamente 13 vezes menor, está relacionada diretamente ao tipo de resolução alvo do trabalho de [Corrêa 2017]. A arquitetura proposta neste trabalho processa vídeo em resolução 4K (3840×2160) à 60 qps, enquanto que a de [Corrêa 2017] processa qua-

Tabela 1. Resumo das características de soluções para predição intra quadros do padrão HEVC.

Critérios	Zhou (2013)	Cong Liu (2013)	Abram-owski (2014)	Corrêa (2017)	Este trabalho
pós-filtros	Não	Sim	Sim	Sim	sim
Decisão de Modo	Único Melhor SAD	Único Melhor SATD	n/d	Oito Melhores SADs	não
BD-Rate	n/d	10%	n/d	0.17%	n/a
Simulação	n/d	n/d	Sim	Não	Sim
Tecnologia	TSMC 130 nm	TSMC 65 nm	TSMC 130 nm	*NanGate* 45 nm	TSMC 45 nm
Área (k *gates*)	324	77	127	4952	260
Frequência	400 MHz	600 MHz	200 MHz	529 MHz	800 MHz
Potência	n/d	n/d	n/d	363,23 mW	26,46 mW
Taxa de Proces.	HD 1080p 60 qps	HD 1080p 30 qps	HD 1080p 35 qps	UHD 8K 120 qps	UHD 4K 60 qps

dros 8K (7680×4320) à 120 qps. Logo a arquitetura de [Corrêa 2017] processa 8 vezes mais dados, já que a resolução 8K possui 4 vezes mais dados que a 4K e a taxa de 120 qps tem o dobro de quadros que a de 60 qps. Levando essas diferenças em consideração, a arquitetura proposta neste trabalho teria uma potência 8 vezes maior, i.e., uma potência de 225,36 µW. Com isso, é possível dizer que a arquitetura proposta neste trabalho possui, aproximadamente, 62% da potência da arquitetura de [Corrêa 2017]. Ainda, essa diferença não leva em consideração que este trabalho não realiza a busca de melhor candidato que certamente aumentaria a potência.

6. Conclusões e Trabalhos Futuros

Uma vez que este trabalho se baseia na predição intra quadros do padrão de codificação HEVC, esta foi apresentada em detalhes. Mostrou-se que a predição intra quadros do HEVC é composta por três principais etapas, sendo elas: pré-processamento de amostras de referência, uma etapa opcional onde é realizado o tratamento inicial das amostras utilizadas na predição; predição de amostras, onde o padrão de codificação define três principais tipos de preditores, cada qual para capturar uma característica presente em vídeos; por fim, foi apresentada a etapa de pós-processamento de amostras preditas, outra etapa opcional onde é aplicado um filtro para suavizar transições de blocos.

Na Seção 3 foi apresentado o projeto e implementação do bloco acelerador em *hardware* para a geração dos blocos candidatos da predição intra quadros do padrão de codificação HEVC. A principal característica que diferencia este trabalho dos correlatos é justamente a capacidade de compor qualquer tamanho de bloco a partir de sub-blocos, i.e., blocos de tamanho 4×4. Para isso, lógicas de controle foram elaboradas para ser possível a predição de blocos maiores do que 4×4, dada que a predição de uma amostra depende de uma série de quesitos (e.g. tamanho do bloco, modo de predição e posição da amostra sendo predita).

Finalmente, na Seção 5 foram realizadas análises acerca do acelerador em *hardware* proposto neste trabalho. Verificou-se que a arquitetura proposta neste trabalho consome cerca de 62% da potência em relação a solução apresentada por Corrêa 2017 ao custo de um aumento de 1,5 vezes na frequência de operação. Comparações entre os trabalhos citados não são muito justas, já que cada um foi sintetizado por ferramentas diferentes e para tecnologias diferentes. Ainda assim, a arquitetura aqui proposta apresentou resultados similares aos presentes nos trabalhos da literatura.

7. Trabalhos futuros

Realizar as etapas em laranja no fluxo apresentado na Figura 5 permitiam melhorar as estimativas de área e potência. Nesse sentido, ainda que demande bastante tempo, outra forma de melhorar a avaliação da arquitetura proposta seria a realização de simulações com diferentes amostras de vídeo da *Common Test Conditions* (CTC) (neste trabalho, cada uma das 28 simulações durou cerca de 4 horas).

Embora a arquitetura tenha sido planejada para compor blocos maiores a partir dos blocos de tamanho 4×4, algumas lógicas de controle comuns aos preditores não foram abstraídas. Um possível trabalho futuro seria justamente explorar otimizações que poderiam ser realizadas na arquitetura proposta e, dessa forma melhorar seu desempenho. Junto com as otimizações, poderia ser realizada a implementação de um módulo de decisão (busca do melhor candidato), permitindo avaliar qual o impacto da lógica de decisão de modo.

Ainda em relação à avaliação, pode ser feito uma análise da frequência máxima da arquitetura, podendo até definir outras resoluções alvos das quais a arquitetura seja capaz de processar.

Também, um possível trabalho seria explorar a estratégia de geração dos candidatos através de sub-blocos no futuro padrão de codificação estado-da-arte, conhecido como *Versatile Video Coding* (VVC).

Referências

Bonotto, B., Cancellier, L. H., Monteiro, M., Seidel, I., and Güntzel, J. L. A. (2018). A Named-Pipe Library for Hardware Simulation. In *SIM2018*. SBC.

Cisco (2017). VNI complete forecast highlights: Global - 2016 year in review. Technical report, Cisco.

Conti, M., Li, Q. Q., Maragno, A., and Spolaor, R. (2018). The dark side(-channel) of mobile devices: A survey on network traffic analysis. *IEEE Communications Surveys Tutorials*, 20(4):2658–2713.

Corrêa, M. M. (2017). Desenvolvimento arquitetural para predição intraquadro do padrão hevc de codificação de vídeos.

Herglotz, C., Springer, D., and Kaup, A. (2014). Modeling the energy consumption of hevc p- and b-frame decoding. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 3661–3665.

Ling, N. (2012). High efficiency video coding and its 3d extension: A research perspective. In *2012 7th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pages 2150–2155.

Seidel, I., Monteiro, M., Bonotto, B., Agostini, L. V., and Güntzel, J. L. (2019). Energy-efficient hadamard-based satd hardware architectures through calculation reuse. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 66(6):2102–2115.

Sullivan, G., Ohm, J., Han, W.-J., and Wiegand, T. (2012). Overview of the high efficiency video coding (HEVC) standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1649–1668.

Sze, V., Budagavi, M., Sullivan, G. J., and and (2014). *High Efficiency Video Coding (HEVC)*. Springer Publishing Company, US.

Zhou, N., Ding, D., Yu, L., and and (2013). On hardware architecture and processing order of hevc intra prediction module. In *2013 Picture Coding Symposium (PCS)*, pages 101–104.

ANEXO B – CÓDIGO FONTE

B.1 *SOFTWARE* DE REFERÊNCIA (HM)

Listing B.1 – Alterações em “TComPrediction.cpp”

```
--- TComPrediction_original.cpp  2018-06-15 14:18:12.000000000 -0300
+++ TComPrediction_modified.cpp  2019-11-07 16:48:56.000000000 -0300
@@ -35,11 +35,54 @@
     \brief    prediction class
 */

+
+/* ################################################### */
+/* ############### BONOTTO'S INCLUDES ############### */
+/* ################################################### */
+
+
+#include <iostream>
+#include <eclpipes.h>
+#include <ctime>
+
+
+/* ################################################### */
+/* ############### BONOTTO'S INCLUDES ############### */
+/* ################################################### */
+
+
+using namespace std;
+
 #include <memory.h>
 #include "TComPrediction.h"
 #include "TComPic.h"
 #include "TComTU.h"

+
+/* ################################################### */
+/* ############### CHECK FRAME CHANGE ############### */
+/* ################################################### */
+
+
+namespace encoding_frame
+{
+  B_Frame* B_Frame::b_frame = 0;
+
+  B_Frame* B_Frame::get_frame()
+  {
+    if(!b_frame)
+        b_frame = new B_Frame();
+
+    return b_frame;
+  }
+}
+
+
```

```
+/* ################################################## */
+/* ############### CHECK FRAME CHANGE ############### */
+/* ################################################## */
+
+
 //! \ingroup TLibCommon
 //! \{

@@ -74,6 +117,42 @@
 : m_pLumaRecBuffer(0)
 , m_iLumaRecStride(0)
 {
+
+
+  /*
   ################################################################################################
   */
+  /* ################## BEGIN HERE THE BONOTTO'S INITIAL MODIFICATIONS
   ################## */
+  /*
   ################################################################################################
   */
+
+
+  cout << endl << "################ BEGINNING INITIALIZATION ################"
   << endl;
+
+  srand (time(NULL));
+
+  short auxiliary = 0;
+
+  write_to("../pipes/frames", &auxiliary, sizeof(short));
+  write_to("../pipes/samples", &auxiliary, sizeof(short));
+  write_to("../pipes/default", &auxiliary, sizeof(short));
+  write_to("../pipes/dc_results", &auxiliary, sizeof(short));
+  write_to("../pipes/planar_results", &auxiliary, sizeof(short));
+
+  for (int i = 2; i < 35; ++i)
+  {
+    ostringstream pipe_path;
+
+    pipe_path << "../pipes/angular_mode" << i << "_results";
+
+    write_to(pipe_path.str().c_str(), &auxiliary, sizeof(short));
+  }
+
+  cout << endl << "################ ENDING INITIALIZATION ################" <<
   endl;
+
+
+  /*
   ################################################################################################
   */
+  /* ################## END HERE THE BONOTTO'S INITIAL MODIFICATIONS
   ################## */
+  /*
   ################################################################################################
```

```
     */
96  +
97  +
98     for(UInt ch=0; ch<MAX_NUM_COMPONENT; ch++)
99     {
100      for(UInt buf=0; buf<2; buf++)
101 @@ -387,7 +466,9 @@
102    }
103  }
104
105 -Void TComPrediction::predIntraAng( const ComponentID compID, UInt uiDirMode,
       Pel* piOrg /* Will be null for decoding */, UInt uiOrgStride, Pel* piPred,
       UInt uiStride, TComTU &rTu, const Bool bUseFilteredPredSamples, const Bool
       bUseLosslessDPCM )
106 +unsigned blockCounter = 1;
107 +
108 +Void TComPrediction::predIntraAng( const ComponentID compID, UInt uiDirMode,
       Pel* piOrg /* Will be null for decoding */, UInt uiOrgStride, Pel* piPred,
       UInt uiStride, TComTU &rTu, const Bool bUseFilteredPredSamples, const Bool
       bUseLosslessDPCM ) /*<------------------------- ALL HERE
       ----------------------*/
109  {
110    const ChannelType    channelType = toChannelType(compID);
111    const TComRectangle &rect        = rTu.getRect(isLuma(compID) ? COMPONENT_Y :
         COMPONENT_Cb);
112 @@ -444,8 +525,63 @@
113    }
114    else
115    {
116 +
117      const Pel *ptrSrc = getPredictorPtr( compID, bUseFilteredPredSamples );
118
119 +
120 +    /*
       ################################################################################
       */
121 +    /* ################## BEGIN HERE THE BONOTTO'S MODIFICATIONS
       ################## */
122 +    /*
       ################################################################################
       */
123 +
124 +    if (iWidth == To_place && iHeight == To_place)
125 +    {
126 +      if (blockCounter < 101)
127 +      {
128 +        printf("\nBlock number: %u\n", blockCounter++);
129 +        fflush(stdout);
130 +      }
131 +
132 +      write_to("../pipes/default", &iWidth, sizeof(short));
133 +      write_to("../pipes/default", &((ptrSrc + sw + 1)[iWidth - sw]), sizeof(
       short));
134 +      write_to("../pipes/default", &((ptrSrc + sw + 1)[iWidth * sw - 1]),
       sizeof(short));
135 +      write_to("../pipes/default", &((ptrSrc + sw + 1)[-sw - 1]), sizeof(short)
       );
```

```
136 +
137 +       for (int p = 0; p < iWidth / 2; ++p)
138 +       {
139 +         for (int k = p * 4; k < p * 4 + 4; ++k)
140 +           write_to("../pipes/samples", &((ptrSrc + sw + 1)[k - sw]), sizeof(
    short));
141 +
142 +         for (int k = p * 4; k < p * 4 + 4; ++k)
143 +           write_to("../pipes/samples", &((ptrSrc + sw + 1)[k * sw - 1]), sizeof
    (short));
144 +       }
145 +
146 +       #if 00043_BEST_EFFORT_DECODING
147 +         const Int bitDept = rTu.getCU()->getSlice()->getSPS()->
    getStreamBitDepth(channelType);
148 +       #else
149 +         const Int bitDept = rTu.getCU()->getSlice()->getSPS()->getBitDepth(
    channelType);
150 +       #endif
151 +
152 +       MyPredIntraPlanar(ptrSrc + sw + 1, sw, iWidth, iHeight);
153 +       MyPredIntraDC(ptrSrc + sw + 1, sw, iWidth, iHeight);
154 +
155 +       for (int myMode = 2; myMode < 35; ++myMode)
156 +         MyPredIntraAngular(bitDept, ptrSrc + sw + 1, sw, pDst, uiStride, iWidth
    , iHeight, myMode);
157 +
158 +       encoding_frame::B_Frame* frame_pointer = encoding_frame::B_Frame::
    get_frame();
159 +
160 +       bool frame_changed = frame_pointer->get_frame_changed();
161 +
162 +       short frame = 0 + frame_changed;
163 +
164 +       write_to("../pipes/frames", &frame, sizeof(short));
165 +     }
166 +
167 +
168 +     /*
    ////////////////////////////////////////////////////////////////////////////
    */
169 +     /* ################## END HERE THE BONOTTO'S MODIFICATIONS
    ################## */
170 +     /*
    ////////////////////////////////////////////////////////////////////////////
    */
171 +
172 +
173       if ( uiDirMode == PLANAR_IDX )
174       {
175         xPredIntraPlanar( ptrSrc+sw+1, sw, pDst, uiStride, iWidth, iHeight );
176 @@ -472,6 +608,215 @@
177
178   }
179
180 +Void TComPrediction::MyPredIntraPlanar(const Pel* pSrc, Int srcStride, UInt
    width, UInt height)
```

```
+{
+   Int leftColumn[MAX_CU_SIZE+1];
+   Int topRow[MAX_CU_SIZE+1];
+   Int bottomRow[MAX_CU_SIZE];
+   Int rightColumn[MAX_CU_SIZE];
+
+   UInt shift1Dhor = g_aucConvertToBit[width] + 2;
+   UInt shift1Dver = g_aucConvertToBit[height] + 2;
+
+   for(Int k = 0; k < width + 1; ++k)
+      topRow[k] = pSrc[k - srcStride];
+
+   for (Int k = 0; k < height + 1; ++k)
+      leftColumn[k] = pSrc[k * srcStride - 1];
+
+   Int topRight = topRow[width];
+   Int bottomLeft = leftColumn[height];
+
+   for(Int k = 0; k < width; ++k)
+   {
+      bottomRow[k] = bottomLeft - topRow[k];
+      topRow[k] <<= shift1Dver;
+   }
+
+   for(Int k = 0; k < height; ++k)
+   {
+      rightColumn[k] = topRight - leftColumn[k];
+      leftColumn[k] <<= shift1Dhor;
+   }
+
+      const UInt topRowShift = 0;
+
+      short _result[height][width];
+
+   for (Int y = 0; y < height; ++y)
+   {
+      Int horPred = leftColumn[y] + width;
+
+      for (Int x = 0; x < width; ++x)
+      {
+         horPred += rightColumn[y];
+         topRow[x] += bottomRow[x];
+
+         Int vertPred = (topRow[x] + topRowShift) >> topRowShift;
+
+         _result[y][x] = (horPred + vertPred) >> (shift1Dhor + 1);
+      }
+   }
+
+   for (Int p = 0; p < height / 4; ++p)
+      for (Int q = 0; q < width / 4; ++q)
+         for (Int k = p * 4; k < p * 4 + 4; ++k)
+            for (Int l = q * 4; l < q * 4 + 4; ++l)
+               write_to("../pipes/planar_results", &(_result[k][l]), sizeof(
    short));
+}
+
```

```
237 +Void TComPrediction::MyPredIntraDC(const Pel* pSrc, Int srcStride, UInt width,
        UInt height)
238 +{
239 +  Int iSum = 0;
240 +
241 +  for (Int k = 0; k < width; ++k)
242 +    iSum += pSrc[k - srcStride];
243 +
244 +  for (Int k = 0; k < height; ++k)
245 +    iSum += pSrc[k * srcStride - 1];
246 +
247 +  Pel dcval = (iSum + width) / (width + height);
248 +
249 +  Pel _result[height][width];
250 +
251 +  for (Int y = 0; y < height; ++y)
252 +    for (Int x = 0; x < width; ++x)
253 +      _result[y][x] = dcval;
254 +
255 +  _result[0][0] = (pSrc[-srcStride] + pSrc[-1] + 2 + 2 * dcval) >> 2;
256 +
257 +  for (Int x = 1; x < width; ++x)
258 +    _result[0][x] = (pSrc[x - srcStride] + 2 + 3 * dcval) >> 2;
259 +
260 +  for (Int y = 1; y < height; ++y)
261 +    _result[y][0] = (pSrc[y * srcStride - 1] + 2 + 3 * dcval) >> 2;
262 +
263 +  for (Int p = 0; p < height / 4; ++p)
264 +    for (Int q = 0; q < width / 4; ++q)
265 +      for (Int k = p * 4; k < p * 4 + 4; ++k)
266 +        for (Int l = q * 4; l < q * 4 + 4; ++l)
267 +          write_to("../pipes/dc_results", &(_result[k][l]), sizeof(short));
268 +}
269 +
270 +Void TComPrediction::MyPredIntraAngular(Int bitDepth, const Pel* pSrc, Int
        srcStride, Pel* pTrueDst, Int dstStrideTrue, UInt width, UInt height, int
        myMode)
271 +{
272 +  static const Int angTable[9]    = {0,    2,    5,   9,  13,  17,  21,  26,
        32};
273 +  static const Int invAngTable[9] = {0, 4096, 1638, 910, 630, 482, 390, 315,
        256};
274 +
275 +  const Bool bIsModeVer         = myMode >= 18;
276 +  const Int  intraPredAngleMode = bIsModeVer ? (Int) myMode - VER_IDX :  -((Int
        ) myMode - HOR_IDX);
277 +  const Int  signAng            = intraPredAngleMode < 0 ? -1 : 1;
278 +  const Int  absAngMode         = abs(intraPredAngleMode);
279 +
280 +  Int invAngle       = invAngTable[absAngMode];
281 +  Int absAng         = angTable[absAngMode];
282 +  Int intraPredAngle = signAng * absAng;
283 +
284 +  Pel* refMain;
285 +  Pel* refSide;
286 +
287 +  Pel refAbove[3 * MAX_CU_SIZE + 1];
```

```
288 +  Pel refLeft[3 * MAX_CU_SIZE + 1];
289 +
290 +  if (intraPredAngle < 0)
291 +  {
292 +    const Int refMainOffsetPreScale = (bIsModeVer ? height : width ) - 1;
293 +    const Int refMainOffset         = height - 1;
294 +
295 +    for (Int x = 0; x < 2 * width + 1; ++x)
296 +      refAbove[x + refMainOffset] = pSrc[x - srcStride - 1];
297 +
298 +    for (Int y = 0; y < 2 * height + 1; ++y)
299 +      refLeft[y + refMainOffset] = pSrc[(y - 1) * srcStride - 1];
300 +
301 +    refMain = (bIsModeVer ? refAbove : refLeft)  + refMainOffset;
302 +    refSide = (bIsModeVer ? refLeft  : refAbove) + refMainOffset;
303 +
304 +    Int invAngleSum    = 128;
305 +
306 +    for (Int k = -1; k > (refMainOffsetPreScale + 1) * intraPredAngle >> 5; --k
    )
307 +    {
308 +      invAngleSum += invAngle;
309 +      refMain[k] = refSide[invAngleSum >> 8];
310 +    }
311 +  }
312 +  else
313 +  {
314 +    for (Int x = 0; x < 2 * width + 1; ++x)
315 +      refAbove[x] = pSrc[x - srcStride - 1];
316 +
317 +    for (Int y = 0; y < 2 * height + 1; ++y)
318 +      refLeft[y] = pSrc[(y - 1) * srcStride - 1];
319 +
320 +    refMain = bIsModeVer ? refAbove : refLeft;
321 +    refSide = bIsModeVer ? refLeft  : refAbove;
322 +  }
323 +
324 +  const Int dstStride = MAX_CU_SIZE;
325 +
326 +  Pel  tempArray[MAX_CU_SIZE * MAX_CU_SIZE];
327 +  Pel* pDst = tempArray;
328 +
329 +  if (intraPredAngle == 0)
330 +  {
331 +    for (Int y = 0; y < height; ++y)
332 +      for (Int x = 0; x < width ; ++x)
333 +        if (x == 0)
334 +          pDst[y * dstStride + x] = ((32 * refMain[x + 1] + 16) / 32) + ((
    refSide[y + 1] - refSide[0]) >> 1);
335 +        else
336 +          pDst[y * dstStride + x] = (32 * refMain[x + 1] + 16) / 32;
337 +  }
338 +  else
339 +  {
340 +    Pel* pDsty = pDst;
341 +
```

```
342 +    for (Int y = 0, deltaPos = intraPredAngle; y < height; ++y, deltaPos +=
        intraPredAngle, pDsty += dstStride)
343 +    {
344 +      const Int deltaInt   = deltaPos / 32;
345 +      const Int deltaFract = deltaPos < 0 ? ((deltaPos * -1) & 31) * -1 :
        deltaPos & 31;
346 +
347 +      if (deltaFract)
348 +      {
349 +        const Pel *pRM = refMain + deltaInt + 1;
350 +
351 +        Int lastRefMainPel = *pRM++;
352 +
353 +        for (Int x = 0; x < width; ++pRM, ++x)
354 +        {
355 +          Int thisRefMainPel = *pRM;
356 +          pDsty[x + 0] = (Pel) (((32 - deltaFract) * lastRefMainPel +
        deltaFract * thisRefMainPel + 16) >> 5);
357 +          lastRefMainPel = thisRefMainPel;
358 +        }
359 +      }
360 +      else
361 +      {
362 +        for (Int x = 0; x < width; ++x)
363 +          pDsty[x] = refMain[x + deltaInt + 1];
364 +      }
365 +    }
366 +  }
367 +
368 +  Pel _result[height][width];
369 +
370 +  for (int x = 0, k = 0; x < width; ++x, k += MAX_CU_SIZE - width)
371 +    for (int y = 0; y < height; ++y, ++k)
372 +    {
373 +      _result[x][y] = tempArray[k] % 256;
374 +      if (_result[x][y] < 0)
375 +        _result[x][y] += 256;
376 +    }
377 +
378 +  ostringstream pipe_path;
379 +
380 +  pipe_path << "../pipes/angular_mode" << myMode << "_results";
381 +
382 +  for (Int q = 0; q < width / 4; ++q)
383 +    for (Int p = 0; p < height / 4; ++p)
384 +      for (Int k = p * 4; k < p * 4 + 4; ++k)
385 +        for (Int l = q * 4; l < q * 4 + 4; ++l)
386 +          write_to(pipe_path.str().c_str(), &(_result[k][l]), sizeof(short));
387 +}
388 +
389   /** Check for identical motion in both motion vector direction of a bi-
        directional predicted CU
390    * \returns true, if motion vectors and reference pictures match
391    */
```

Listing B.2 – Alterações em “TComPrediction.hpp”

```
--- TComPrediction_o.hpp   2018-06-15 14:18:12.000000000 -0300
+++ TComPrediction_m.hpp   2019-07-26 14:20:35.000000000 -0300
@@ -51,6 +51,42 @@
 //! \ingroup TLibCommon
 //! \{

+
+/* ################################################### */
+/* ############### CHECK FRAME CHANGE ############### */
+/* ################################################### */
+
+
+namespace encoding_frame
+{
+  class B_Frame
+  {
+  private:
+    B_Frame()
+    {
+      frame_changed = false;
+    }
+
+  public:
+    bool frame_changed;
+    static B_Frame* b_frame;
+    static B_Frame* get_frame();
+
+    ~B_Frame()
+    {}
+
+    bool get_frame_changed()
+    {
+      if (!frame_changed)
+        return false;
+
+      frame_changed = false;
+      return true;
+    }
+  };
+}
+
+
 //
     ================================================================================================

 // Class definition
 //
     ================================================================================================

@@ -86,6 +122,22 @@

   Void xPredIntraAng            ( Int bitDepth, const Pel* pSrc, Int srcStride,
       Pel* pDst, Int dstStride, UInt width, UInt height, ChannelType
      channelType, UInt dirMode, const Bool bEnableEdgeFilters );
   Void xPredIntraPlanar         ( const Pel* pSrc, Int srcStride, Pel* rpDst,
      Int dstStride, UInt width, UInt height );
+
```

```
51 +
52 +  /* ############################################################################
       */
53 +  /* ################### BEGIN HERE THE BONOTTO'S FUNCTIONS ###################
       */
54 +  /* ############################################################################
       */
55 +
56 +
57 +  Void MyPredIntraPlanar(const Pel* pSrc, Int srcStride, UInt width, UInt
       height);
58 +  Void MyPredIntraDC(const Pel* pSrc, Int srcStride, UInt uiWidth, UInt
       uiHeight);
59 +  Void MyPredIntraAngular(Int bitDepth, const Pel* pSrc, Int srcStride, Pel*
       pTrueDst, Int dstStrideTrue, UInt width, UInt height, int myMode);
60 +
61 +
62 +  /* ############################################################################
       */
63 +  /* ##################### END HERE THE BONOTTO'S FUNCTIONS ####################
       */
64 +  /* ############################################################################
       */
65 +
66
67     // motion compensation functions
68     Void xPredInterUni             ( TComDataCU* pcCU,
          UInt uiPartAddr,                Int iWidth, Int iHeight, RefPicList
          eRefPicList, TComYuv* pcYuvPred, Bool bi=false          );
```

B.2 *TESTBENCH* EM *SOFTWARE*

Listing B.3 – Código fonte do *testbench* em *software*

```
1
2  /* ############################################################ */
3  /* ###################### Used Libraries ###################### */
4  /* ############################################################ */
5
6
7  // Internal
8  #include <stdio.h>
9  #include <stdlib.h>
10 #include <string.h>
11
12 // External
13 #include "acc_user.h"
14 #include "./src/eclpipes.h"
15
16
17 /* ############################################################ */
18 /* ################### Internal Definitions ################### */
19 /* ############################################################ */
20
21
22 // Type used for basis convertions
```

```
23 typedef char *string;
24
25 // Common types used
26 #define set_value s_setval_value
27 #define get_value p_acc_value
28
29 // Default block size
30 #define height 4
31 #define width 4
32
33 // Total of frames to be simulated
34 // #define total_tests 100000
35 // #define total_tests 10000
36 // #define total_tests 1000
37 #define total_tests 50
38
39 // Total of blocks to be simulated
40 #define total_frames 600
41
42 // Debugg condition for verbose
43 #define debugg false
44
45
46 /* ############################################################ */
47 /* ##################### Global Definitions ##################### */
48 /* ############################################################ */
49
50
51 // Global delay for set on registers writing or reading
52 s_setval_delay delay;
53
54 // Predictors block step counter
55 unsigned angular_mode34_step_counter;
56 unsigned angular_mode33_step_counter;
57 unsigned angular_mode32_step_counter;
58 unsigned angular_mode31_step_counter;
59 unsigned angular_mode30_step_counter;
60 unsigned angular_mode29_step_counter;
61 unsigned angular_mode28_step_counter;
62 unsigned angular_mode27_step_counter;
63 unsigned angular_mode26_step_counter;
64 unsigned angular_mode25_step_counter;
65 unsigned angular_mode24_step_counter;
66 unsigned angular_mode23_step_counter;
67 unsigned angular_mode22_step_counter;
68 unsigned angular_mode21_step_counter;
69 unsigned angular_mode20_step_counter;
70 unsigned angular_mode19_step_counter;
71 unsigned angular_mode18_step_counter;
72 unsigned angular_mode17_step_counter;
73 unsigned angular_mode16_step_counter;
74 unsigned angular_mode15_step_counter;
75 unsigned angular_mode14_step_counter;
76 unsigned angular_mode13_step_counter;
77 unsigned angular_mode12_step_counter;
78 unsigned angular_mode11_step_counter;
79 unsigned angular_mode10_step_counter;
```

```
80  unsigned angular_mode9_step_counter;
81  unsigned angular_mode8_step_counter;
82  unsigned angular_mode7_step_counter;
83  unsigned angular_mode6_step_counter;
84  unsigned angular_mode5_step_counter;
85  unsigned angular_mode4_step_counter;
86  unsigned angular_mode3_step_counter;
87  unsigned angular_mode2_step_counter;
88  unsigned planar_step_counter;
89  unsigned dc_step_counter;
90
91  // Frame counter
92  long long unsigned frame_counter;
93
94  // Test counter
95  long long unsigned test_counter;
96
97  // Counter of accepted tests
98  long long unsigned right_tests_counter;
99
100 // Counter of simulation test
101 long long unsigned simulation_counter;
102
103 // Condition used to verify if the current test was accepted
104 bool test_passed;
105
106 // Global handlers for registers
107 handle read_def;
108 handle def_readed;
109 handle read_samples;
110 handle samples_readed;
111 handle check_planar_step;
112 handle planar_step_checked;
113 handle check_dc_step;
114 handle dc_step_checked;
115 handle check_angular_mode2_step;
116 handle angular_mode2_step_checked;
117 handle check_angular_mode3_step;
118 handle angular_mode3_step_checked;
119 handle check_angular_mode4_step;
120 handle angular_mode4_step_checked;
121 handle check_angular_mode5_step;
122 handle angular_mode5_step_checked;
123 handle check_angular_mode6_step;
124 handle angular_mode6_step_checked;
125 handle check_angular_mode7_step;
126 handle angular_mode7_step_checked;
127 handle check_angular_mode8_step;
128 handle angular_mode8_step_checked;
129 handle check_angular_mode9_step;
130 handle angular_mode9_step_checked;
131 handle check_angular_mode10_step;
132 handle angular_mode10_step_checked;
133 handle check_angular_mode11_step;
134 handle angular_mode11_step_checked;
135 handle check_angular_mode12_step;
136 handle angular_mode12_step_checked;
```

```
137 handle check_angular_mode13_step;
138 handle angular_mode13_step_checked;
139 handle check_angular_mode14_step;
140 handle angular_mode14_step_checked;
141 handle check_angular_mode15_step;
142 handle angular_mode15_step_checked;
143 handle check_angular_mode16_step;
144 handle angular_mode16_step_checked;
145 handle check_angular_mode17_step;
146 handle angular_mode17_step_checked;
147 handle check_angular_mode18_step;
148 handle angular_mode18_step_checked;
149 handle check_angular_mode19_step;
150 handle angular_mode19_step_checked;
151 handle check_angular_mode20_step;
152 handle angular_mode20_step_checked;
153 handle check_angular_mode21_step;
154 handle angular_mode21_step_checked;
155 handle check_angular_mode22_step;
156 handle angular_mode22_step_checked;
157 handle check_angular_mode23_step;
158 handle angular_mode23_step_checked;
159 handle check_angular_mode24_step;
160 handle angular_mode24_step_checked;
161 handle check_angular_mode25_step;
162 handle angular_mode25_step_checked;
163 handle check_angular_mode26_step;
164 handle angular_mode26_step_checked;
165 handle check_angular_mode27_step;
166 handle angular_mode27_step_checked;
167 handle check_angular_mode28_step;
168 handle angular_mode28_step_checked;
169 handle check_angular_mode29_step;
170 handle angular_mode29_step_checked;
171 handle check_angular_mode30_step;
172 handle angular_mode30_step_checked;
173 handle check_angular_mode31_step;
174 handle angular_mode31_step_checked;
175 handle check_angular_mode32_step;
176 handle angular_mode32_step_checked;
177 handle check_angular_mode33_step;
178 handle angular_mode33_step_checked;
179 handle check_angular_mode34_step;
180 handle angular_mode34_step_checked;
181 handle done;
182
183 handle samples_kn[width];
184 handle samples_nk[width];
185 handle sample_Nn;
186 handle sample_nN;
187 handle sample_NN;
188 handle N;
189
190 handle planar_p[height][width];
191 handle dc_p[height][width];
192 handle angular_mode2_p[height][width];
193 handle angular_mode3_p[height][width];
```

```
194 handle angular_mode4_p[height][width];
195 handle angular_mode5_p[height][width];
196 handle angular_mode6_p[height][width];
197 handle angular_mode7_p[height][width];
198 handle angular_mode8_p[height][width];
199 handle angular_mode9_p[height][width];
200 handle angular_mode10_p[height][width];
201 handle angular_mode11_p[height][width];
202 handle angular_mode12_p[height][width];
203 handle angular_mode13_p[height][width];
204 handle angular_mode14_p[height][width];
205 handle angular_mode15_p[height][width];
206 handle angular_mode16_p[height][width];
207 handle angular_mode17_p[height][width];
208 handle angular_mode18_p[height][width];
209 handle angular_mode19_p[height][width];
210 handle angular_mode20_p[height][width];
211 handle angular_mode21_p[height][width];
212 handle angular_mode22_p[height][width];
213 handle angular_mode23_p[height][width];
214 handle angular_mode24_p[height][width];
215 handle angular_mode25_p[height][width];
216 handle angular_mode26_p[height][width];
217 handle angular_mode27_p[height][width];
218 handle angular_mode28_p[height][width];
219 handle angular_mode29_p[height][width];
220 handle angular_mode30_p[height][width];
221 handle angular_mode31_p[height][width];
222 handle angular_mode32_p[height][width];
223 handle angular_mode33_p[height][width];
224 handle angular_mode34_p[height][width];
225
226 handle end_simulation;
227
228
229 /* ################################################################ */
230 /* ##################### Testbench Functions ###################### */
231 /* ################################################################ */
232
233
234 // Converts a binary string to a int value
235 int convert (string content, int no_bits)
236 {
237    int conv = 0;
238    int i = 0;
239    int j = 0;
240    int bin = 0;
241
242    for (i = no_bits - 1; i >= 0; i = i - 1)
243    {
244       if (*(content + i) == 49)
245          bin = 1;
246
247       else if (*(content + i) == 120)
248          bin = 0;
249
250       else if (*(content + i) != 122)
```

```
251             bin = 0;
252
253         conv = conv + (1 << j) * bin;
254         j++;
255     }
256
257     return conv;
258 }
259
260 // Turn true a signal register
261 void set_true (handle * hand)
262 {
263     set_value container;
264
265     container.format = accIntVal;
266     container.value.integer = 1;
267
268     acc_set_value(*hand, &container, &delay);
269 }
270
271 // Shows the simulation resume
272 void make_statistics ()
273 {
274     printf("\n#################␣ENDING␣SIMULATION␣################\n\n");
275
276     printf("Total␣frames␣simulated:␣%llu\n", frame_counter);
277     printf("Total␣tests:␣%llu\n\n", test_counter);
278     printf("Total␣tests␣passed:␣%llu\n", right_tests_counter);
279     printf("Total␣tests␣don't␣passed:␣%llu\n\n", test_counter -
        right_tests_counter);
280 }
281
282 // Initialize the variables of testbench and open the pipes
283 void default_initialize ()
284 {
285     printf("\n#################␣BEGINNING␣INITIALIZATION␣################\n");
286
287     delay.model = accNoDelay;
288
289     test_counter = 0;
290     dc_step_counter = 0;
291     simulation_counter = 0;
292     planar_step_counter = 0;
293     angular_mode2_step_counter = 0;
294     angular_mode3_step_counter = 0;
295     angular_mode4_step_counter = 0;
296     angular_mode5_step_counter = 0;
297     angular_mode6_step_counter = 0;
298     angular_mode7_step_counter = 0;
299     angular_mode8_step_counter = 0;
300     angular_mode9_step_counter = 0;
301     angular_mode10_step_counter = 0;
302     angular_mode11_step_counter = 0;
303     angular_mode12_step_counter = 0;
304     angular_mode13_step_counter = 0;
305     angular_mode14_step_counter = 0;
306     angular_mode15_step_counter = 0;
```

```
307     angular_mode16_step_counter = 0;
308     angular_mode17_step_counter = 0;
309     angular_mode18_step_counter = 0;
310     angular_mode19_step_counter = 0;
311     angular_mode20_step_counter = 0;
312     angular_mode21_step_counter = 0;
313     angular_mode22_step_counter = 0;
314     angular_mode23_step_counter = 0;
315     angular_mode24_step_counter = 0;
316     angular_mode25_step_counter = 0;
317     angular_mode26_step_counter = 0;
318     angular_mode27_step_counter = 0;
319     angular_mode28_step_counter = 0;
320     angular_mode29_step_counter = 0;
321     angular_mode30_step_counter = 0;
322     angular_mode31_step_counter = 0;
323     angular_mode32_step_counter = 0;
324     angular_mode33_step_counter = 0;
325     angular_mode34_step_counter = 0;
326     right_tests_counter = 0;
327 
328     test_passed = true;
329 
330     short auxiliary;
331 
332     read_from("../pipes/frames", &auxiliary, sizeof(short));
333     read_from("../pipes/samples", &auxiliary, sizeof(short));
334     read_from("../pipes/default", &auxiliary, sizeof(short));
335     read_from("../pipes/dc_results", &auxiliary, sizeof(short));
336     read_from("../pipes/planar_results", &auxiliary, sizeof(short));
337 
338     for (int i = 2; i < 35; ++i)
339     {
340         char pipe_path[50] = "../pipes/angular_mode";
341 
342         char number[3];
343         sprintf(number, "%d", i);
344 
345         strcat(pipe_path, number);
346 
347         strcat(pipe_path, "_results");
348 
349         read_from(pipe_path, &auxiliary, sizeof(short));
350     }
351 
352     printf("\n#################␣ENDING␣INITIALIZATION␣################\n");
353 }
354 
355 // Verifies if the simulation reached the objective
356 void check_final ()
357 {
358     short changed;
359 
360     read_from("../pipes/frames", &changed, sizeof(short));
361 
362     if (changed)
363         frame_counter++;
```

```
364
365     if (frame_counter >= total_frames || simulation_counter++ == total_tests - 1)
366     {
367         set_true(&end_simulation);
368         make_statistics();
369     }
370 }
371
372 // Counts the result iteration
373 void loop_finalization ()
374 {
375     get_value value;
376
377     int condition = convert(acc_fetch_value(done, "%b", value), 1);
378
379     if (condition)
380     {
381         angular_mode34_step_counter = 0;
382         angular_mode33_step_counter = 0;
383         angular_mode32_step_counter = 0;
384         angular_mode31_step_counter = 0;
385         angular_mode30_step_counter = 0;
386         angular_mode29_step_counter = 0;
387         angular_mode28_step_counter = 0;
388         angular_mode27_step_counter = 0;
389         angular_mode26_step_counter = 0;
390         angular_mode25_step_counter = 0;
391         angular_mode24_step_counter = 0;
392         angular_mode23_step_counter = 0;
393         angular_mode22_step_counter = 0;
394         angular_mode21_step_counter = 0;
395         angular_mode20_step_counter = 0;
396         angular_mode19_step_counter = 0;
397         angular_mode18_step_counter = 0;
398         angular_mode17_step_counter = 0;
399         angular_mode16_step_counter = 0;
400         angular_mode15_step_counter = 0;
401         angular_mode14_step_counter = 0;
402         angular_mode13_step_counter = 0;
403         angular_mode12_step_counter = 0;
404         angular_mode11_step_counter = 0;
405         angular_mode10_step_counter = 0;
406         angular_mode9_step_counter = 0;
407         angular_mode8_step_counter = 0;
408         angular_mode7_step_counter = 0;
409         angular_mode6_step_counter = 0;
410         angular_mode5_step_counter = 0;
411         angular_mode4_step_counter = 0;
412         angular_mode3_step_counter = 0;
413         angular_mode2_step_counter = 0;
414         planar_step_counter = 0;
415         dc_step_counter = 0;
416         test_counter++;
417
418         if (test_passed)
419             right_tests_counter++;
420         else
```

```
421             test_passed = true;
422
423         // if (!(test_counter % 1000000))
424         printf("Current Test: %u\n", test_counter);
425
426         check_final();
427     }
428 }
429
430 // Gets and compare the angular mode 2 predicted block
431 void get_angular_mode2_results ()
432 {
433     get_value value;
434
435     int condition = convert(acc_fetch_value(check_angular_mode2_step, "%b", value
        ), 1);
436
437     if (condition)
438     {
439         short arch_result[height][width];
440
441         for (int i = 0; i < height; ++i)
442             for (int j = 0; j < width; ++j)
443                 arch_result[i][j] = convert(acc_fetch_value(angular_mode2_p[i][j], "
                    %b", value), 8);
444
445         short reff_result = 0;
446
447         for (int i = 0; i < height; ++i)
448             for (int j = 0; j < width; ++j)
449             {
450                 read_from("../pipes/angular_mode2_results", &reff_result, sizeof(
                    short));
451
452                 if (reff_result != arch_result[i][j])
453                 {
454                     test_passed = false;
455
456                     printf("\n################ ANGULAR MODE 2 ERROR DETECTED : TEST %
                        llu - Step %u - INDEX [%d, %d] ################\n",
                        test_counter, angular_mode2_step_counter, i, j);
457                     printf("Expected Value  = %d\nGenerated Value = %d\n",
                        reff_result, arch_result[i][j]);
458                 }
459                 else if (debugg)
460                 {
461                     printf("\n################ CORRECT ON ANGULAR MODE 2: TEST %llu -
                        Step %u - INDEX [%d, %d] ################\n", test_counter,
                        angular_mode2_step_counter, i, j);
462                 }
463             }
464
465         angular_mode2_step_counter++;
466
467         // set_true(&angular_mode2_step_checked);
468     }
469 }
```

```
470
471 // Gets and compare the angular mode 3 predicted block
472 void get_angular_mode3_results ()
473 {
474    get_value value;
475
476    int condition = convert(acc_fetch_value(check_angular_mode3_step, "%b", value
        ), 1);
477
478    if (condition)
479    {
480       short arch_result[height][width];
481
482       for (int i = 0; i < height; ++i)
483          for (int j = 0; j < width; ++j)
484             arch_result[i][j] = convert(acc_fetch_value(angular_mode3_p[i][j], "
                %b", value), 8);
485
486       short reff_result = 0;
487
488       for (int i = 0; i < height; ++i)
489          for (int j = 0; j < width; ++j)
490          {
491             read_from("../pipes/angular_mode3_results", &reff_result, sizeof(
                 short));
492
493             if (reff_result != arch_result[i][j])
494             {
495                test_passed = false;
496
497                printf("\n#################␣ANGULAR␣MODE␣3␣ERROR␣DETECTED:␣TEST␣%
                    llu␣-␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣#################\n",
                    test_counter, angular_mode3_step_counter, i, j);
498                printf("Expected␣Value␣␣=␣%d\nGenerated␣Value␣=␣%d\n",
                    reff_result, arch_result[i][j]);
499             }
500             else if (debugg)
501             {
502                printf("\n#################␣CORRECT␣ON␣ANGULAR␣MODE␣3:␣TEST␣%llu␣-
                    ␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣#################\n", test_counter,
                    angular_mode3_step_counter, i, j);
503             }
504          }
505
506       angular_mode3_step_counter++;
507
508       // set_true(&angular_mode3_step_checked);
509    }
510 }
511
512 // Gets and compare the angular mode 4 predicted block
513 void get_angular_mode4_results ()
514 {
515    get_value value;
516
517    int condition = convert(acc_fetch_value(check_angular_mode4_step, "%b", value
        ), 1);
```

```
518
519     if (condition)
520     {
521         short arch_result[height][width];
522
523         for (int i = 0; i < height; ++i)
524             for (int j = 0; j < width; ++j)
525                 arch_result[i][j] = convert(acc_fetch_value(angular_mode4_p[i][j], "
                        %b", value), 8);
526
527         short reff_result = 0;
528
529         for (int i = 0; i < height; ++i)
530             for (int j = 0; j < width; ++j)
531             {
532                 read_from("../pipes/angular_mode4_results", &reff_result, sizeof(
                        short));
533
534                 if (reff_result != arch_result[i][j])
535                 {
536                     test_passed = false;
537
538                     printf("\n##################␣ANGULAR␣MODE␣4␣ERROR␣DETECTED:␣TEST␣%
                            llu␣-␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣#################\n",
                            test_counter, angular_mode4_step_counter, i, j);
539                     printf("Expected␣Value␣␣=␣%d\nGenerated␣Value␣=␣%d\n",
                            reff_result, arch_result[i][j]);
540                 }
541                 else if (debugg)
542                 {
543                     printf("\n##################␣CORRECT␣ON␣ANGULAR␣MODE␣4:␣TEST␣%llu␣-
                            ␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣#################\n", test_counter,
                            angular_mode4_step_counter, i, j);
544                 }
545             }
546
547         angular_mode4_step_counter++;
548
549         // set_true(&angular_mode4_step_checked);
550     }
551 }
552
553 // Gets and compare the angular mode 5 predicted block
554 void get_angular_mode5_results ()
555 {
556     get_value value;
557
558     int condition = convert(acc_fetch_value(check_angular_mode5_step, "%b", value
            ), 1);
559
560     if (condition)
561     {
562         short arch_result[height][width];
563
564         for (int i = 0; i < height; ++i)
565             for (int j = 0; j < width; ++j)
566                 arch_result[i][j] = convert(acc_fetch_value(angular_mode5_p[i][j], "
```

```
                %b", value), 8);
567
568         short reff_result = 0;
569
570         for (int i = 0; i < height; ++i)
571            for (int j = 0; j < width; ++j)
572            {
573               read_from("../pipes/angular_mode5_results", &reff_result, sizeof(
                   short));
574
575               if (reff_result != arch_result[i][j])
576               {
577                  test_passed = false;
578
579                  printf("\n##################␣ANGULAR␣MODE␣5␣ERROR␣DETECTED:␣TEST␣%
                      llu␣-␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣################\n",
                      test_counter, angular_mode5_step_counter, i, j);
580                  printf("Expected␣Value␣␣=␣%d\nGenerated␣Value␣=␣%d\n",
                      reff_result, arch_result[i][j]);
581               }
582               else if (debugg)
583               {
584                  printf("\n##################␣CORRECT␣ON␣ANGULAR␣MODE␣5:␣TEST␣%llu␣-
                      ␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣################\n", test_counter,
                      angular_mode5_step_counter, i, j);
585               }
586            }
587
588         angular_mode5_step_counter++;
589
590         // set_true(&angular_mode5_step_checked);
591      }
592   }
593
594   // Gets and compare the angular mode 6 predicted block
595   void get_angular_mode6_results ()
596   {
597      get_value value;
598
599      int condition = convert(acc_fetch_value(check_angular_mode6_step, "%b", value
          ), 1);
600
601      if (condition)
602      {
603         short arch_result[height][width];
604
605         for (int i = 0; i < height; ++i)
606            for (int j = 0; j < width; ++j)
607               arch_result[i][j] = convert(acc_fetch_value(angular_mode6_p[i][j], "
                   %b", value), 8);
608
609         short reff_result = 0;
610
611         for (int i = 0; i < height; ++i)
612            for (int j = 0; j < width; ++j)
613            {
614               read_from("../pipes/angular_mode6_results", &reff_result, sizeof(
```

```
                short));
615
616             if (reff_result != arch_result[i][j])
617             {
618                 test_passed = false;
619
620                 printf("\n################ ANGULAR MODE 6 ERROR DETECTED : TEST %
                        llu - Step %u - INDEX [%d, %d] ################\n",
                        test_counter, angular_mode6_step_counter, i, j);
621                 printf("Expected Value   = %d\nGenerated Value = %d\n",
                        reff_result, arch_result[i][j]);
622             }
623             else if (debugg)
624             {
625                 printf("\n################ CORRECT ON ANGULAR MODE 6: TEST %llu -
                        Step %u - INDEX [%d, %d] ################\n", test_counter,
                        angular_mode6_step_counter, i, j);
626             }
627         }
628
629         angular_mode6_step_counter++;
630
631         // set_true(&angular_mode6_step_checked);
632     }
633 }
634
635 // Gets and compare the angular mode 7 predicted block
636 void get_angular_mode7_results ()
637 {
638     get_value value;
639
640     int condition = convert(acc_fetch_value(check_angular_mode7_step, "%b", value
        ), 1);
641
642     if (condition)
643     {
644         short arch_result[height][width];
645
646         for (int i = 0; i < height; ++i)
647             for (int j = 0; j < width; ++j)
648                 arch_result[i][j] = convert(acc_fetch_value(angular_mode7_p[i][j], "
                    %b", value), 8);
649
650         short reff_result = 0;
651
652         for (int i = 0; i < height; ++i)
653             for (int j = 0; j < width; ++j)
654             {
655                 read_from("../pipes/angular_mode7_results", &reff_result, sizeof(
                    short));
656
657                 if (reff_result != arch_result[i][j])
658                 {
659                     test_passed = false;
660
661                     printf("\n################ ANGULAR MODE 7 ERROR DETECTED : TEST %
                        llu - Step %u - INDEX [%d, %d] ################\n",
```

```
                    test_counter, angular_mode7_step_counter, i, j);
662                 printf("Expected Value   = %d\nGenerated Value = %d\n",
                        reff_result, arch_result[i][j]);
663             }
664             else if (debugg)
665             {
666                 printf("\n################ CORRECT ON ANGULAR MODE 7: TEST %llu -
                        Step %u - INDEX [%d, %d] ################\n", test_counter,
                        angular_mode7_step_counter, i, j);
667             }
668         }
669
670         angular_mode7_step_counter++;
671
672         // set_true(&angular_mode7_step_checked);
673     }
674 }
675
676 // Gets and compare the angular mode 8 predicted block
677 void get_angular_mode8_results ()
678 {
679     get_value value;
680
681     int condition = convert(acc_fetch_value(check_angular_mode8_step, "%b", value
            ), 1);
682
683     if (condition)
684     {
685         short arch_result[height][width];
686
687         for (int i = 0; i < height; ++i)
688             for (int j = 0; j < width; ++j)
689                 arch_result[i][j] = convert(acc_fetch_value(angular_mode8_p[i][j], "
                        %b", value), 8);
690
691         short reff_result = 0;
692
693         for (int i = 0; i < height; ++i)
694             for (int j = 0; j < width; ++j)
695             {
696                 read_from("../pipes/angular_mode8_results", &reff_result, sizeof(
                        short));
697
698                 if (reff_result != arch_result[i][j])
699                 {
700                     test_passed = false;
701
702                     printf("\n################ ANGULAR MODE 8 ERROR DETECTED : TEST %
                            llu - Step %u - INDEX [%d, %d] ################\n",
                            test_counter, angular_mode8_step_counter, i, j);
703                     printf("Expected Value   = %d\nGenerated Value = %d\n",
                            reff_result, arch_result[i][j]);
704                 }
705                 else if (debugg)
706                 {
707                     printf("\n################ CORRECT ON ANGULAR MODE 8: TEST %llu -
                            Step %u - INDEX [%d, %d] ################\n", test_counter,
```

```
                    angular_mode8_step_counter, i, j);
708             }
709          }
710
711       angular_mode8_step_counter++;
712
713       // set_true(&angular_mode8_step_checked);
714    }
715 }
716
717 // Gets and compare the angular mode 9 predicted block
718 void get_angular_mode9_results ()
719 {
720    get_value value;
721
722    int condition = convert(acc_fetch_value(check_angular_mode9_step, "%b", value
       ), 1);
723
724    if (condition)
725    {
726       short arch_result[height][width];
727
728       for (int i = 0; i < height; ++i)
729          for (int j = 0; j < width; ++j)
730             arch_result[i][j] = convert(acc_fetch_value(angular_mode9_p[i][j], "
                %b", value), 8);
731
732       short reff_result = 0;
733
734       for (int i = 0; i < height; ++i)
735          for (int j = 0; j < width; ++j)
736          {
737             read_from("../pipes/angular_mode9_results", &reff_result, sizeof(
                short));
738
739             if (reff_result != arch_result[i][j])
740             {
741                test_passed = false;
742
743                printf("\n################ ANGULAR MODE 9 ERROR DETECTED: TEST %
                   llu - Step %u - INDEX [%d, %d] ################\n",
                   test_counter, angular_mode9_step_counter, i, j);
744                printf("Expected Value  = %d\nGenerated Value = %d\n",
                   reff_result, arch_result[i][j]);
745             }
746             else if (debugg)
747             {
748                printf("\n################ CORRECT ON ANGULAR MODE 9: TEST %llu -
                   Step %u - INDEX [%d, %d] ################\n", test_counter,
                   angular_mode9_step_counter, i, j);
749             }
750          }
751
752       angular_mode9_step_counter++;
753
754       // set_true(&angular_mode9_step_checked);
755    }
```

```
}

// Gets and compare the angular mode 10 predicted block
void get_angular_mode10_results ()
{
   get_value value;

   int condition = convert(acc_fetch_value(check_angular_mode10_step, "%b",
       value), 1);

   if (condition)
   {
      short arch_result[height][width];

      for (int i = 0; i < height; ++i)
         for (int j = 0; j < width; ++j)
            arch_result[i][j] = convert(acc_fetch_value(angular_mode10_p[i][j],
                "%b", value), 8);

      short reff_result = 0;

      for (int i = 0; i < height; ++i)
         for (int j = 0; j < width; ++j)
         {
            read_from("../pipes/angular_mode10_results", &reff_result, sizeof(
                short));

            if (reff_result != arch_result[i][j])
            {
               test_passed = false;

               printf("\n##################␣ANGULAR␣MODE␣10␣ERROR␣DETECTED:␣TEST␣%
                   llu␣-␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣################\n",
                   test_counter, angular_mode10_step_counter, i, j);
               printf("Expected␣Value␣␣=␣%d\nGenerated␣Value␣=␣%d\n",
                   reff_result, arch_result[i][j]);
            }
            else if (debugg)
            {
               printf("\n##################␣CORRECT␣ON␣ANGULAR␣MODE␣10:␣TEST␣%llu␣
                   -␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣################\n", test_counter,
                    angular_mode10_step_counter, i, j);
            }
         }

      angular_mode10_step_counter++;

      // set_true(&angular_mode10_step_checked);
   }
}

// Gets and compare the angular mode 11 predicted block
void get_angular_mode11_results ()
{
   get_value value;

   int condition = convert(acc_fetch_value(check_angular_mode11_step, "%b",
```

```
        value), 1);
805
806     if (condition)
807     {
808         short arch_result[height][width];
809
810         for (int i = 0; i < height; ++i)
811             for (int j = 0; j < width; ++j)
812                 arch_result[i][j] = convert(acc_fetch_value(angular_mode11_p[i][j],
                        "%b", value), 8);
813
814         short reff_result = 0;
815
816         for (int i = 0; i < height; ++i)
817             for (int j = 0; j < width; ++j)
818             {
819                 read_from("../pipes/angular_mode11_results", &reff_result, sizeof(
                        short));
820
821                 if (reff_result != arch_result[i][j])
822                 {
823                     test_passed = false;
824
825                     printf("\n#################␣ANGULAR␣MODE␣11␣ERROR␣DETECTED:␣TEST␣%
                            llu␣-␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣################\n",
                            test_counter, angular_mode11_step_counter, i, j);
826                     printf("Expected␣Value␣␣=␣%d\nGenerated␣Value␣=␣%d\n",
                            reff_result, arch_result[i][j]);
827                 }
828                 else if (debugg)
829                 {
830                     printf("\n#################␣CORRECT␣ON␣ANGULAR␣MODE␣11:␣TEST␣%llu␣
                            -␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣################\n", test_counter,
                             angular_mode11_step_counter, i, j);
831                 }
832             }
833
834         angular_mode11_step_counter++;
835
836         // set_true(&angular_mode11_step_checked);
837     }
838 }
839
840 // Gets and compare the angular mode 12 predicted block
841 void get_angular_mode12_results ()
842 {
843     get_value value;
844
845     int condition = convert(acc_fetch_value(check_angular_mode12_step, "%b",
            value), 1);
846
847     if (condition)
848     {
849         short arch_result[height][width];
850
851         for (int i = 0; i < height; ++i)
852             for (int j = 0; j < width; ++j)
```

```
853             arch_result[i][j] = convert(acc_fetch_value(angular_mode12_p[i][j],
                    "%b", value), 8);
854
855         short reff_result = 0;
856
857         for (int i = 0; i < height; ++i)
858             for (int j = 0; j < width; ++j)
859             {
860                 read_from("../pipes/angular_mode12_results", &reff_result, sizeof(
                        short));
861
862                 if (reff_result != arch_result[i][j])
863                 {
864                     test_passed = false;
865
866                     printf("\n#################␣ANGULAR␣MODE␣12␣ERROR␣DETECTED:␣TEST␣%
                            llu␣-␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣################\n",
                            test_counter, angular_mode12_step_counter, i, j);
867                     printf("Expected␣Value␣␣=␣%d\nGenerated␣Value␣=␣%d\n",
                            reff_result, arch_result[i][j]);
868                 }
869                 else if (debugg)
870                 {
871                     printf("\n#################␣CORRECT␣ON␣ANGULAR␣MODE␣12:␣TEST␣%llu␣
                            -␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣################\n", test_counter,
                             angular_mode12_step_counter, i, j);
872                 }
873             }
874
875         angular_mode12_step_counter++;
876
877         // set_true(&angular_mode12_step_checked);
878     }
879 }
880
881 // Gets and compare the angular mode 13 predicted block
882 void get_angular_mode13_results ()
883 {
884     get_value value;
885
886     int condition = convert(acc_fetch_value(check_angular_mode13_step, "%b",
            value), 1);
887
888     if (condition)
889     {
890         short arch_result[height][width];
891
892         for (int i = 0; i < height; ++i)
893             for (int j = 0; j < width; ++j)
894                 arch_result[i][j] = convert(acc_fetch_value(angular_mode13_p[i][j],
                        "%b", value), 8);
895
896         short reff_result = 0;
897
898         for (int i = 0; i < height; ++i)
899             for (int j = 0; j < width; ++j)
900             {
```

```
901                 read_from("../pipes/angular_mode13_results", &reff_result, sizeof(
                        short));
902
903                 if (reff_result != arch_result[i][j])
904                 {
905                     test_passed = false;
906
907                     printf("\n################ ANGULAR MODE 13 ERROR DETECTED: TEST %
                            llu - Step %u - INDEX [%d, %d] ################\n",
                            test_counter, angular_mode13_step_counter, i, j);
908                     printf("Expected Value  = %d\nGenerated Value = %d\n",
                            reff_result, arch_result[i][j]);
909                 }
910                 else if (debugg)
911                 {
912                     printf("\n################ CORRECT ON ANGULAR MODE 13: TEST %llu
                            - Step %u - INDEX [%d, %d] ################\n", test_counter,
                             angular_mode13_step_counter, i, j);
913                 }
914             }
915
916         angular_mode13_step_counter++;
917
918         // set_true(&angular_mode13_step_checked);
919     }
920 }
921
922 // Gets and compare the angular mode 14 predicted block
923 void get_angular_mode14_results ()
924 {
925     get_value value;
926
927     int condition = convert(acc_fetch_value(check_angular_mode14_step, "%b",
            value), 1);
928
929     if (condition)
930     {
931         short arch_result[height][width];
932
933         for (int i = 0; i < height; ++i)
934             for (int j = 0; j < width; ++j)
935                 arch_result[i][j] = convert(acc_fetch_value(angular_mode14_p[i][j],
                        "%b", value), 8);
936
937         short reff_result = 0;
938
939         for (int i = 0; i < height; ++i)
940             for (int j = 0; j < width; ++j)
941             {
942                 read_from("../pipes/angular_mode14_results", &reff_result, sizeof(
                        short));
943
944                 if (reff_result != arch_result[i][j])
945                 {
946                     test_passed = false;
947
948                     printf("\n################ ANGULAR MODE 14 ERROR DETECTED: TEST %
```

```
                    llu␣-␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣################\n",
                    test_counter, angular_mode14_step_counter, i, j);
949                 printf("Expected␣Value␣␣=␣%d\nGenerated␣Value␣=␣%d\n",
                    reff_result, arch_result[i][j]);
950             }
951             else if (debugg)
952             {
953                 printf("\n################␣CORRECT␣ON␣ANGULAR␣MODE␣14:␣TEST␣%llu␣
                    -␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣################\n", test_counter,
                     angular_mode14_step_counter, i, j);
954             }
955         }
956
957         angular_mode14_step_counter++;
958
959         // set_true(&angular_mode14_step_checked);
960     }
961 }
962
963 // Gets and compare the angular mode 15 predicted block
964 void get_angular_mode15_results ()
965 {
966     get_value value;
967
968     int condition = convert(acc_fetch_value(check_angular_mode15_step, "%b",
        value), 1);
969
970     if (condition)
971     {
972         short arch_result[height][width];
973
974         for (int i = 0; i < height; ++i)
975             for (int j = 0; j < width; ++j)
976                 arch_result[i][j] = convert(acc_fetch_value(angular_mode15_p[i][j],
                    "%b", value), 8);
977
978         short reff_result = 0;
979
980         for (int i = 0; i < height; ++i)
981             for (int j = 0; j < width; ++j)
982             {
983                 read_from("../pipes/angular_mode15_results", &reff_result, sizeof(
                    short));
984
985                 if (reff_result != arch_result[i][j])
986                 {
987                     test_passed = false;
988
989                     printf("\n################␣ANGULAR␣MODE␣15␣ERROR␣DETECTED:␣TEST␣%
                        llu␣-␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣################\n",
                        test_counter, angular_mode15_step_counter, i, j);
990                     printf("Expected␣Value␣␣=␣%d\nGenerated␣Value␣=␣%d\n",
                        reff_result, arch_result[i][j]);
991                 }
992                 else if (debugg)
993                 {
994                     printf("\n################␣CORRECT␣ON␣ANGULAR␣MODE␣15:␣TEST␣%llu␣
```

```
                    -␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣################\n", test_counter,
                     angular_mode15_step_counter, i, j);
995             }
996          }
997
998       angular_mode15_step_counter++;
999
1000      // set_true(&angular_mode15_step_checked);
1001   }
1002}
1003
1004// Gets and compare the angular mode 16 predicted block
1005void get_angular_mode16_results ()
1006{
1007   get_value value;
1008
1009   int condition = convert(acc_fetch_value(check_angular_mode16_step, "%b",
          value), 1);
1010
1011   if (condition)
1012   {
1013      short arch_result[height][width];
1014
1015      for (int i = 0; i < height; ++i)
1016         for (int j = 0; j < width; ++j)
1017            arch_result[i][j] = convert(acc_fetch_value(angular_mode16_p[i][j],
                  "%b", value), 8);
1018
1019      short reff_result = 0;
1020
1021      for (int i = 0; i < height; ++i)
1022         for (int j = 0; j < width; ++j)
1023         {
1024            read_from("../pipes/angular_mode16_results", &reff_result, sizeof(
                  short));
1025
1026            if (reff_result != arch_result[i][j])
1027            {
1028               test_passed = false;
1029
1030               printf("\n################␣ANGULAR␣MODE␣16␣ERROR␣DETECTED:␣TEST␣%
                     llu␣-␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣################\n",
                     test_counter, angular_mode16_step_counter, i, j);
1031               printf("Expected␣Value␣␣=␣%d\nGenerated␣Value␣=␣%d\n",
                     reff_result, arch_result[i][j]);
1032            }
1033            else if (debugg)
1034            {
1035               printf("\n################␣CORRECT␣ON␣ANGULAR␣MODE␣16:␣TEST␣%llu␣
                     -␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣################\n", test_counter,
                      angular_mode16_step_counter, i, j);
1036            }
1037         }
1038
1039      angular_mode16_step_counter++;
1040
1041      // set_true(&angular_mode16_step_checked);
```

```
1042     }
1043 }
1044
1045 // Gets and compare the angular mode 17 predicted block
1046 void get_angular_mode17_results ()
1047 {
1048     get_value value;
1049
1050     int condition = convert(acc_fetch_value(check_angular_mode17_step, "%b",
         value), 1);
1051
1052     if (condition)
1053     {
1054         short arch_result[height][width];
1055
1056         for (int i = 0; i < height; ++i)
1057             for (int j = 0; j < width; ++j)
1058                 arch_result[i][j] = convert(acc_fetch_value(angular_mode17_p[i][j],
                     "%b", value), 8);
1059
1060         short reff_result = 0;
1061
1062         for (int i = 0; i < height; ++i)
1063             for (int j = 0; j < width; ++j)
1064             {
1065                 read_from("../pipes/angular_mode17_results", &reff_result, sizeof(
                     short));
1066
1067                 if (reff_result != arch_result[i][j])
1068                 {
1069                     test_passed = false;
1070
1071                     printf("\n#################␣ANGULAR␣MODE␣17␣ERROR␣DETECTED:␣TEST␣%
                         llu␣-␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣################\n",
                         test_counter, angular_mode17_step_counter, i, j);
1072                     printf("Expected␣Value␣␣=␣%d\nGenerated␣Value␣=␣%d\n",
                         reff_result, arch_result[i][j]);
1073                 }
1074                 else if (debugg)
1075                 {
1076                     printf("\n#################␣CORRECT␣ON␣ANGULAR␣MODE␣17:␣TEST␣%llu␣
                         -␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣################\n", test_counter,
                          angular_mode17_step_counter, i, j);
1077                 }
1078             }
1079
1080         angular_mode17_step_counter++;
1081
1082         // set_true(&angular_mode17_step_checked);
1083     }
1084 }
1085
1086 // Gets and compare the angular mode 18 predicted block
1087 void get_angular_mode18_results ()
1088 {
1089     get_value value;
1090
```

```
1091     int condition = convert(acc_fetch_value(check_angular_mode18_step, "%b",
             value), 1);
1092
1093     if (condition)
1094     {
1095        short arch_result[height][width];
1096
1097        for (int i = 0; i < height; ++i)
1098           for (int j = 0; j < width; ++j)
1099              arch_result[i][j] = convert(acc_fetch_value(angular_mode18_p[i][j],
                     "%b", value), 8);
1100
1101        short reff_result = 0;
1102
1103        for (int i = 0; i < height; ++i)
1104           for (int j = 0; j < width; ++j)
1105           {
1106              read_from("../pipes/angular_mode18_results", &reff_result, sizeof(
                     short));
1107
1108              if (reff_result != arch_result[i][j])
1109              {
1110                 test_passed = false;
1111
1112                 printf("\n################ ANGULAR MODE 18 ERROR DETECTED: TEST %
                        llu - Step %u - INDEX [%d, %d] ################\n",
                        test_counter, angular_mode18_step_counter, i, j);
1113                 printf("Expected Value  = %d\nGenerated Value = %d\n",
                        reff_result, arch_result[i][j]);
1114              }
1115              else if (debugg)
1116              {
1117                 printf("\n################ CORRECT ON ANGULAR MODE 18: TEST %llu
                        - Step %u - INDEX [%d, %d] ################\n", test_counter,
                         angular_mode18_step_counter, i, j);
1118              }
1119           }
1120
1121        angular_mode18_step_counter++;
1122
1123        // set_true(&angular_mode18_step_checked);
1124     }
1125  }
1126
1127  // Gets and compare the angular mode 19 predicted block
1128  void get_angular_mode19_results ()
1129  {
1130     get_value value;
1131
1132     int condition = convert(acc_fetch_value(check_angular_mode19_step, "%b",
             value), 1);
1133
1134     if (condition)
1135     {
1136        short arch_result[height][width];
1137
1138        for (int i = 0; i < height; ++i)
```

```
1139            for (int j = 0; j < width; ++j)
1140               arch_result[i][j] = convert(acc_fetch_value(angular_mode19_p[i][j],
                      "%b", value), 8);
1141
1142         short reff_result = 0;
1143
1144         for (int i = 0; i < height; ++i)
1145            for (int j = 0; j < width; ++j)
1146            {
1147               read_from("../pipes/angular_mode19_results", &reff_result, sizeof(
                      short));
1148
1149               if (reff_result != arch_result[i][j])
1150               {
1151                  test_passed = false;
1152
1153                  printf("\n#################␣ANGULAR␣MODE␣19␣ERROR␣DETECTED:␣TEST␣%
                         llu␣-␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣#################\n",
                         test_counter, angular_mode19_step_counter, i, j);
1154                  printf("Expected␣Value␣␣=␣%d\nGenerated␣Value␣=␣%d\n",
                         reff_result, arch_result[i][j]);
1155               }
1156               else if (debugg)
1157               {
1158                  printf("\n#################␣CORRECT␣ON␣ANGULAR␣MODE␣19:␣TEST␣%llu␣
                         -␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣#################\n", test_counter,
                          angular_mode19_step_counter, i, j);
1159               }
1160            }
1161
1162         angular_mode19_step_counter++;
1163
1164         // set_true(&angular_mode19_step_checked);
1165      }
1166   }
1167
1168   // Gets and compare the angular mode 20 predicted block
1169   void get_angular_mode20_results ()
1170   {
1171      get_value value;
1172
1173      int condition = convert(acc_fetch_value(check_angular_mode20_step, "%b",
          value), 1);
1174
1175      if (condition)
1176      {
1177         short arch_result[height][width];
1178
1179         for (int i = 0; i < height; ++i)
1180            for (int j = 0; j < width; ++j)
1181               arch_result[i][j] = convert(acc_fetch_value(angular_mode20_p[i][j],
                      "%b", value), 8);
1182
1183         short reff_result = 0;
1184
1185         for (int i = 0; i < height; ++i)
1186            for (int j = 0; j < width; ++j)
```

```
1187            {
1188                read_from("../pipes/angular_mode20_results", &reff_result, sizeof(
                        short));
1189
1190                if (reff_result != arch_result[i][j])
1191                {
1192                    test_passed = false;
1193
1194                    printf("\n################ ANGULAR MODE 20 ERROR DETECTED: TEST %
                            llu - Step %u - INDEX [%d, %d] ################\n",
                            test_counter, angular_mode20_step_counter, i, j);
1195                    printf("Expected Value  = %d\nGenerated Value = %d\n",
                            reff_result, arch_result[i][j]);
1196                }
1197                else if (debugg)
1198                {
1199                    printf("\n################ CORRECT ON ANGULAR MODE 20: TEST %llu
                            - Step %u - INDEX [%d, %d] ################\n", test_counter,
                             angular_mode20_step_counter, i, j);
1200                }
1201            }
1202
1203        angular_mode20_step_counter++;
1204
1205        // set_true(&angular_mode20_step_checked);
1206    }
1207 }
1208
1209 // Gets and compare the angular mode 21 predicted block
1210 void get_angular_mode21_results ()
1211 {
1212    get_value value;
1213
1214    int condition = convert(acc_fetch_value(check_angular_mode21_step, "%b",
            value), 1);
1215
1216    if (condition)
1217    {
1218        short arch_result[height][width];
1219
1220        for (int i = 0; i < height; ++i)
1221            for (int j = 0; j < width; ++j)
1222                arch_result[i][j] = convert(acc_fetch_value(angular_mode21_p[i][j],
                        "%b", value), 8);
1223
1224        short reff_result = 0;
1225
1226        for (int i = 0; i < height; ++i)
1227            for (int j = 0; j < width; ++j)
1228            {
1229                read_from("../pipes/angular_mode21_results", &reff_result, sizeof(
                        short));
1230
1231                if (reff_result != arch_result[i][j])
1232                {
1233                    test_passed = false;
1234
```

```
1235                printf("\n#################␣ANGULAR␣MODE␣21␣ERROR␣DETECTED:␣TEST␣%
                        llu␣-␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣################\n",
                        test_counter, angular_mode21_step_counter, i, j);
1236                printf("Expected␣Value␣␣=␣%d\nGenerated␣Value␣=␣%d\n",
                        reff_result, arch_result[i][j]);
1237             }
1238             else if (debugg)
1239             {
1240                printf("\n#################␣CORRECT␣ON␣ANGULAR␣MODE␣21:␣TEST␣%llu␣
                        -␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣################\n", test_counter,
                         angular_mode21_step_counter, i, j);
1241             }
1242          }
1243
1244       angular_mode21_step_counter++;
1245
1246       // set_true(&angular_mode21_step_checked);
1247    }
1248 }
1249
1250 // Gets and compare the angular mode 22 predicted block
1251 void get_angular_mode22_results ()
1252 {
1253    get_value value;
1254
1255    int condition = convert(acc_fetch_value(check_angular_mode22_step, "%b",
          value), 1);
1256
1257    if (condition)
1258    {
1259       short arch_result[height][width];
1260
1261       for (int i = 0; i < height; ++i)
1262          for (int j = 0; j < width; ++j)
1263             arch_result[i][j] = convert(acc_fetch_value(angular_mode22_p[i][j],
                   "%b", value), 8);
1264
1265       short reff_result = 0;
1266
1267       for (int i = 0; i < height; ++i)
1268          for (int j = 0; j < width; ++j)
1269          {
1270             read_from("../pipes/angular_mode22_results", &reff_result, sizeof(
                   short));
1271
1272             if (reff_result != arch_result[i][j])
1273             {
1274                test_passed = false;
1275
1276                printf("\n#################␣ANGULAR␣MODE␣22␣ERROR␣DETECTED:␣TEST␣%
                        llu␣-␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣################\n",
                        test_counter, angular_mode22_step_counter, i, j);
1277                printf("Expected␣Value␣␣=␣%d\nGenerated␣Value␣=␣%d\n",
                        reff_result, arch_result[i][j]);
1278             }
1279             else if (debugg)
1280             {
```

```
1281                printf("\n################ CORRECT ON ANGULAR MODE 22: TEST %llu
                        - Step %u - INDEX [%d, %d] ################\n", test_counter,
                         angular_mode22_step_counter, i, j);
1282             }
1283          }
1284
1285       angular_mode22_step_counter++;
1286
1287       // set_true(&angular_mode22_step_checked);
1288    }
1289 }
1290
1291 // Gets and compare the angular mode 23 predicted block
1292 void get_angular_mode23_results ()
1293 {
1294    get_value value;
1295
1296    int condition = convert(acc_fetch_value(check_angular_mode23_step, "%b",
          value), 1);
1297
1298    if (condition)
1299    {
1300       short arch_result[height][width];
1301
1302       for (int i = 0; i < height; ++i)
1303          for (int j = 0; j < width; ++j)
1304             arch_result[i][j] = convert(acc_fetch_value(angular_mode23_p[i][j],
                   "%b", value), 8);
1305
1306       short reff_result = 0;
1307
1308       for (int i = 0; i < height; ++i)
1309          for (int j = 0; j < width; ++j)
1310          {
1311             read_from("../pipes/angular_mode23_results", &reff_result, sizeof(
                   short));
1312
1313             if (reff_result != arch_result[i][j])
1314             {
1315                test_passed = false;
1316
1317                printf("\n################ ANGULAR MODE 23 ERROR DETECTED: TEST %
                      llu - Step %u - INDEX [%d, %d] ################\n",
                      test_counter, angular_mode23_step_counter, i, j);
1318                printf("Expected Value  = %d\nGenerated Value = %d\n",
                      reff_result, arch_result[i][j]);
1319             }
1320             else if (debugg)
1321             {
1322                printf("\n################ CORRECT ON ANGULAR MODE 23: TEST %llu
                      - Step %u - INDEX [%d, %d] ################\n", test_counter,
                       angular_mode23_step_counter, i, j);
1323             }
1324          }
1325
1326       angular_mode23_step_counter++;
1327
```

```
1328        // set_true(&angular_mode23_step_checked);
1329     }
1330  }
1331
1332  // Gets and compare the angular mode 24 predicted block
1333  void get_angular_mode24_results ()
1334  {
1335     get_value value;
1336
1337     int condition = convert(acc_fetch_value(check_angular_mode24_step, "%b",
           value), 1);
1338
1339     if (condition)
1340     {
1341        short arch_result[height][width];
1342
1343        for (int i = 0; i < height; ++i)
1344           for (int j = 0; j < width; ++j)
1345              arch_result[i][j] = convert(acc_fetch_value(angular_mode24_p[i][j],
                  "%b", value), 8);
1346
1347        short reff_result = 0;
1348
1349        for (int i = 0; i < height; ++i)
1350           for (int j = 0; j < width; ++j)
1351           {
1352              read_from("../pipes/angular_mode24_results", &reff_result, sizeof(
                     short));
1353
1354              if (reff_result != arch_result[i][j])
1355              {
1356                 test_passed = false;
1357
1358                 printf("\n#################␣ANGULAR␣MODE␣24␣ERROR␣DETECTED:␣TEST␣%
                        llu␣-␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣#################\n",
                        test_counter, angular_mode24_step_counter, i, j);
1359                 printf("Expected␣Value␣␣=␣%d\nGenerated␣Value␣=␣%d\n",
                        reff_result, arch_result[i][j]);
1360              }
1361              else if (debugg)
1362              {
1363                 printf("\n#################␣CORRECT␣ON␣ANGULAR␣MODE␣24:␣TEST␣%llu␣
                        -␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣#################\n", test_counter,
                         angular_mode24_step_counter, i, j);
1364              }
1365           }
1366
1367        angular_mode24_step_counter++;
1368
1369        // set_true(&angular_mode24_step_checked);
1370     }
1371  }
1372
1373  // Gets and compare the angular mode 25 predicted block
1374  void get_angular_mode25_results ()
1375  {
1376     get_value value;
```

```
1377
1378     int condition = convert(acc_fetch_value(check_angular_mode25_step, "%b",
             value), 1);
1379
1380     if (condition)
1381     {
1382         short arch_result[height][width];
1383
1384         for (int i = 0; i < height; ++i)
1385             for (int j = 0; j < width; ++j)
1386                 arch_result[i][j] = convert(acc_fetch_value(angular_mode25_p[i][j],
                        "%b", value), 8);
1387
1388         short reff_result = 0;
1389
1390         for (int i = 0; i < height; ++i)
1391             for (int j = 0; j < width; ++j)
1392             {
1393                 read_from("../pipes/angular_mode25_results", &reff_result, sizeof(
                        short));
1394
1395                 if (reff_result != arch_result[i][j])
1396                 {
1397                     test_passed = false;
1398
1399                     printf("\n#################␣ANGULAR␣MODE␣25␣ERROR␣DETECTED:␣TEST␣%
                            llu␣-␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣################\n",
                            test_counter, angular_mode25_step_counter, i, j);
1400                     printf("Expected␣Value␣␣=␣%d\nGenerated␣Value␣=␣%d\n",
                            reff_result, arch_result[i][j]);
1401                 }
1402                 else if (debugg)
1403                 {
1404                     printf("\n#################␣CORRECT␣ON␣ANGULAR␣MODE␣25:␣TEST␣%llu␣
                            -␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣################\n", test_counter,
                             angular_mode25_step_counter, i, j);
1405                 }
1406             }
1407
1408         angular_mode25_step_counter++;
1409
1410         // set_true(&angular_mode25_step_checked);
1411     }
1412 }
1413
1414 // Gets and compare the angular mode 26 predicted block
1415 void get_angular_mode26_results ()
1416 {
1417     get_value value;
1418
1419     int condition = convert(acc_fetch_value(check_angular_mode26_step, "%b",
             value), 1);
1420
1421     if (condition)
1422     {
1423         short arch_result[height][width];
1424
```

```
1425        for (int i = 0; i < height; ++i)
1426           for (int j = 0; j < width; ++j)
1427              arch_result[i][j] = convert(acc_fetch_value(angular_mode26_p[i][j],
                      "%b", value), 8);
1428
1429        short reff_result = 0;
1430
1431        for (int i = 0; i < height; ++i)
1432           for (int j = 0; j < width; ++j)
1433           {
1434              read_from("../pipes/angular_mode26_results", &reff_result, sizeof(
                      short));
1435
1436              if (reff_result != arch_result[i][j])
1437              {
1438                 test_passed = false;
1439
1440                 printf("\n##################␣ANGULAR␣MODE␣26␣ERROR␣DETECTED:␣TEST␣%
                         llu␣-␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣#################\n",
                         test_counter, angular_mode26_step_counter, i, j);
1441                 printf("Expected␣Value␣␣=␣%d\nGenerated␣Value␣=␣%d\n",
                         reff_result, arch_result[i][j]);
1442              }
1443              else if (debugg)
1444              {
1445                 printf("\n##################␣CORRECT␣ON␣ANGULAR␣MODE␣26:␣TEST␣%llu␣
                         -␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣#################\n", test_counter,
                          angular_mode26_step_counter, i, j);
1446              }
1447           }
1448
1449        angular_mode26_step_counter++;
1450
1451        // set_true(&angular_mode26_step_checked);
1452     }
1453  }
1454
1455  // Gets and compare the angular mode 27 predicted block
1456  void get_angular_mode27_results ()
1457  {
1458     get_value value;
1459
1460     int condition = convert(acc_fetch_value(check_angular_mode27_step, "%b",
          value), 1);
1461
1462     if (condition)
1463     {
1464        short arch_result[height][width];
1465
1466        for (int i = 0; i < height; ++i)
1467           for (int j = 0; j < width; ++j)
1468              arch_result[i][j] = convert(acc_fetch_value(angular_mode27_p[i][j],
                      "%b", value), 8);
1469
1470        short reff_result = 0;
1471
1472        for (int i = 0; i < height; ++i)
```

```
1473            for (int j = 0; j < width; ++j)
1474            {
1475               read_from("../pipes/angular_mode27_results", &reff_result, sizeof(
                      short));
1476
1477               if (reff_result != arch_result[i][j])
1478               {
1479                  test_passed = false;
1480
1481                  printf("\n################ ANGULAR MODE 27 ERROR DETECTED: TEST %
                         llu - Step %u - INDEX [%d, %d] ################\n",
                         test_counter, angular_mode27_step_counter, i, j);
1482                  printf("Expected Value   = %d\nGenerated Value = %d\n",
                         reff_result, arch_result[i][j]);
1483               }
1484               else if (debugg)
1485               {
1486                  printf("\n################ CORRECT ON ANGULAR MODE 27: TEST %llu
                         - Step %u - INDEX [%d, %d] ################\n", test_counter,
                          angular_mode27_step_counter, i, j);
1487               }
1488            }
1489
1490         angular_mode27_step_counter++;
1491
1492         // set_true(&angular_mode27_step_checked);
1493      }
1494   }
1495
1496   // Gets and compare the angular mode 28 predicted block
1497   void get_angular_mode28_results ()
1498   {
1499      get_value value;
1500
1501      int condition = convert(acc_fetch_value(check_angular_mode28_step, "%b",
          value), 1);
1502
1503      if (condition)
1504      {
1505         short arch_result[height][width];
1506
1507         for (int i = 0; i < height; ++i)
1508            for (int j = 0; j < width; ++j)
1509               arch_result[i][j] = convert(acc_fetch_value(angular_mode28_p[i][j],
                      "%b", value), 8);
1510
1511         short reff_result = 0;
1512
1513         for (int i = 0; i < height; ++i)
1514            for (int j = 0; j < width; ++j)
1515            {
1516               read_from("../pipes/angular_mode28_results", &reff_result, sizeof(
                      short));
1517
1518               if (reff_result != arch_result[i][j])
1519               {
1520                  test_passed = false;
```

```
1521
1522                printf("\n#################␣ANGULAR␣MODE␣28␣ERROR␣DETECTED:␣TEST␣%
                        llu␣-␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣################\n",
                        test_counter, angular_mode28_step_counter, i, j);
1523                printf("Expected␣Value␣␣=␣%d\nGenerated␣Value␣=␣%d\n",
                        reff_result, arch_result[i][j]);
1524             }
1525             else if (debugg)
1526             {
1527                printf("\n#################␣CORRECT␣ON␣ANGULAR␣MODE␣28:␣TEST␣%llu␣
                        -␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣################\n", test_counter,
                         angular_mode28_step_counter, i, j);
1528             }
1529          }
1530
1531       angular_mode28_step_counter++;
1532
1533       // set_true(&angular_mode28_step_checked);
1534    }
1535 }
1536
1537 // Gets and compare the angular mode 29 predicted block
1538 void get_angular_mode29_results ()
1539 {
1540    get_value value;
1541
1542    int condition = convert(acc_fetch_value(check_angular_mode29_step, "%b",
          value), 1);
1543
1544    if (condition)
1545    {
1546       short arch_result[height][width];
1547
1548       for (int i = 0; i < height; ++i)
1549          for (int j = 0; j < width; ++j)
1550             arch_result[i][j] = convert(acc_fetch_value(angular_mode29_p[i][j],
                    "%b", value), 8);
1551
1552       short reff_result = 0;
1553
1554       for (int i = 0; i < height; ++i)
1555          for (int j = 0; j < width; ++j)
1556          {
1557             read_from("../pipes/angular_mode29_results", &reff_result, sizeof(
                    short));
1558
1559             if (reff_result != arch_result[i][j])
1560             {
1561                test_passed = false;
1562
1563                printf("\n#################␣ANGULAR␣MODE␣29␣ERROR␣DETECTED:␣TEST␣%
                        llu␣-␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣################\n",
                        test_counter, angular_mode29_step_counter, i, j);
1564                printf("Expected␣Value␣␣=␣%d\nGenerated␣Value␣=␣%d\n",
                        reff_result, arch_result[i][j]);
1565             }
1566             else if (debugg)
```

```
1567                {
1568                    printf("\n################ CORRECT ON ANGULAR MODE 29: TEST %llu
                            - Step %u - INDEX [%d, %d] ################\n", test_counter,
                                angular_mode29_step_counter, i, j);
1569                }
1570            }
1571
1572        angular_mode29_step_counter++;
1573
1574        // set_true(&angular_mode29_step_checked);
1575    }
1576 }
1577
1578 // Gets and compare the angular mode 30 predicted block
1579 void get_angular_mode30_results ()
1580 {
1581    get_value value;
1582
1583    int condition = convert(acc_fetch_value(check_angular_mode30_step, "%b",
            value), 1);
1584
1585    if (condition)
1586    {
1587        short arch_result[height][width];
1588
1589        for (int i = 0; i < height; ++i)
1590            for (int j = 0; j < width; ++j)
1591                arch_result[i][j] = convert(acc_fetch_value(angular_mode30_p[i][j],
                        "%b", value), 8);
1592
1593        short reff_result = 0;
1594
1595        for (int i = 0; i < height; ++i)
1596            for (int j = 0; j < width; ++j)
1597            {
1598                read_from("../pipes/angular_mode30_results", &reff_result, sizeof(
                        short));
1599
1600                if (reff_result != arch_result[i][j])
1601                {
1602                    test_passed = false;
1603
1604                    printf("\n################ ANGULAR MODE 30 ERROR DETECTED: TEST %
                            llu - Step %u - INDEX [%d, %d] ################\n",
                            test_counter, angular_mode30_step_counter, i, j);
1605                    printf("Expected Value  = %d\nGenerated Value = %d\n",
                            reff_result, arch_result[i][j]);
1606                }
1607                else if (debugg)
1608                {
1609                    printf("\n################ CORRECT ON ANGULAR MODE 30: TEST %llu
                            - Step %u - INDEX [%d, %d] ################\n", test_counter,
                                angular_mode30_step_counter, i, j);
1610                }
1611            }
1612
1613        angular_mode30_step_counter++;
```

```
1614
1615        // set_true(&angular_mode30_step_checked);
1616    }
1617 }
1618
1619 // Gets and compare the angular mode 31 predicted block
1620 void get_angular_mode31_results ()
1621 {
1622    get_value value;
1623
1624    int condition = convert(acc_fetch_value(check_angular_mode31_step, "%b",
           value), 1);
1625
1626    if (condition)
1627    {
1628       short arch_result[height][width];
1629
1630       for (int i = 0; i < height; ++i)
1631          for (int j = 0; j < width; ++j)
1632             arch_result[i][j] = convert(acc_fetch_value(angular_mode31_p[i][j],
                  "%b", value), 8);
1633
1634       short reff_result = 0;
1635
1636       for (int i = 0; i < height; ++i)
1637          for (int j = 0; j < width; ++j)
1638          {
1639             read_from("../pipes/angular_mode31_results", &reff_result, sizeof(
                   short));
1640
1641             if (reff_result != arch_result[i][j])
1642             {
1643                test_passed = false;
1644
1645                printf("\n##################␣ANGULAR␣MODE␣31␣ERROR␣DETECTED:␣TEST␣%
                      llu␣-␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣################\n",
                      test_counter, angular_mode31_step_counter, i, j);
1646                printf("Expected␣Value␣␣=␣%d\nGenerated␣Value␣=␣%d\n",
                      reff_result, arch_result[i][j]);
1647             }
1648             else if (debugg)
1649             {
1650                printf("\n##################␣CORRECT␣ON␣ANGULAR␣MODE␣31:␣TEST␣%llu␣
                      -␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣################\n", test_counter,
                       angular_mode31_step_counter, i, j);
1651             }
1652          }
1653
1654       angular_mode31_step_counter++;
1655
1656       // set_true(&angular_mode31_step_checked);
1657    }
1658 }
1659
1660 // Gets and compare the angular mode 32 predicted block
1661 void get_angular_mode32_results ()
1662 {
```

```
1663     get_value value;
1664
1665     int condition = convert(acc_fetch_value(check_angular_mode32_step, "%b",
             value), 1);
1666
1667     if (condition)
1668     {
1669         short arch_result[height][width];
1670
1671         for (int i = 0; i < height; ++i)
1672             for (int j = 0; j < width; ++j)
1673                 arch_result[i][j] = convert(acc_fetch_value(angular_mode32_p[i][j],
                        "%b", value), 8);
1674
1675         short reff_result = 0;
1676
1677         for (int i = 0; i < height; ++i)
1678             for (int j = 0; j < width; ++j)
1679             {
1680                 read_from("../pipes/angular_mode32_results", &reff_result, sizeof(
                        short));
1681
1682                 if (reff_result != arch_result[i][j])
1683                 {
1684                     test_passed = false;
1685
1686                     printf("\n#################␣ANGULAR␣MODE␣32␣ERROR␣DETECTED:␣TEST␣%
                            llu␣-␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣################\n",
                            test_counter, angular_mode32_step_counter, i, j);
1687                     printf("Expected␣Value␣␣=␣%d\nGenerated␣Value␣=␣%d\n",
                            reff_result, arch_result[i][j]);
1688                 }
1689                 else if (debugg)
1690                 {
1691                     printf("\n#################␣CORRECT␣ON␣ANGULAR␣MODE␣32:␣TEST␣%llu␣
                            -␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣################\n", test_counter,
                             angular_mode32_step_counter, i, j);
1692                 }
1693             }
1694
1695         angular_mode32_step_counter++;
1696
1697         // set_true(&angular_mode32_step_checked);
1698     }
1699 }
1700
1701 // Gets and compare the angular mode 33 predicted block
1702 void get_angular_mode33_results ()
1703 {
1704     get_value value;
1705
1706     int condition = convert(acc_fetch_value(check_angular_mode33_step, "%b",
             value), 1);
1707
1708     if (condition)
1709     {
1710         short arch_result[height][width];
```

```
1711
1712        for (int i = 0; i < height; ++i)
1713           for (int j = 0; j < width; ++j)
1714              arch_result[i][j] = convert(acc_fetch_value(angular_mode33_p[i][j],
                      "%b", value), 8);
1715
1716        short reff_result = 0;
1717
1718        for (int i = 0; i < height; ++i)
1719           for (int j = 0; j < width; ++j)
1720           {
1721              read_from("../pipes/angular_mode33_results", &reff_result, sizeof(
                      short));
1722
1723              if (reff_result != arch_result[i][j])
1724              {
1725                 test_passed = false;
1726
1727                 printf("\n##################␣ANGULAR␣MODE␣33␣ERROR␣DETECTED:␣TEST␣%
                         llu␣-␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣#################\n",
                         test_counter, angular_mode33_step_counter, i, j);
1728                 printf("Expected␣Value␣␣=␣%d\nGenerated␣Value␣=␣%d\n",
                         reff_result, arch_result[i][j]);
1729              }
1730              else if (debugg)
1731              {
1732                 printf("\n##################␣CORRECT␣ON␣ANGULAR␣MODE␣33:␣TEST␣%llu␣
                         -␣Step␣%u␣-␣INDEX␣[%d,␣%d]␣#################\n", test_counter,
                          angular_mode33_step_counter, i, j);
1733              }
1734           }
1735
1736        angular_mode33_step_counter++;
1737
1738        // set_true(&angular_mode33_step_checked);
1739     }
1740  }
1741
1742  // Gets and compare the angular mode 34 predicted block
1743  void get_angular_mode34_results ()
1744  {
1745     get_value value;
1746
1747     int condition = convert(acc_fetch_value(check_angular_mode34_step, "%b",
          value), 1);
1748
1749     if (condition)
1750     {
1751        short arch_result[height][width];
1752
1753        for (int i = 0; i < height; ++i)
1754           for (int j = 0; j < width; ++j)
1755              arch_result[i][j] = convert(acc_fetch_value(angular_mode34_p[i][j],
                      "%b", value), 8);
1756
1757        short reff_result = 0;
1758
```

```
1759        for (int i = 0; i < height; ++i)
1760           for (int j = 0; j < width; ++j)
1761           {
1762              read_from("../pipes/angular_mode34_results", &reff_result, sizeof(
                      short));
1763
1764              if (reff_result != arch_result[i][j])
1765              {
1766                 test_passed = false;
1767
1768                 printf("\n################ ANGULAR MODE 34 ERROR DETECTED: TEST %
                         llu - Step %u - INDEX [%d, %d] ################\n",
                         test_counter, angular_mode34_step_counter, i, j);
1769                 printf("Expected Value   = %d\nGenerated Value = %d\n",
                         reff_result, arch_result[i][j]);
1770              }
1771              else if (debugg)
1772              {
1773                 printf("\n################ CORRECT ON ANGULAR MODE 34: TEST %llu
                         - Step %u - INDEX [%d, %d] ################\n", test_counter,
                          angular_mode34_step_counter, i, j);
1774              }
1775           }
1776
1777        angular_mode34_step_counter++;
1778
1779        // set_true(&angular_mode34_step_checked);
1780     }
1781  }
1782
1783  // Gets and compare the dc predicted block
1784  void get_dc_results ()
1785  {
1786     get_value value;
1787
1788     int condition = convert(acc_fetch_value(check_dc_step, "%b", value), 1);
1789
1790     if (condition)
1791     {
1792        short arch_result[height][width];
1793
1794        for (int i = 0; i < height; ++i)
1795           for (int j = 0; j < width; ++j)
1796              arch_result[i][j] = convert(acc_fetch_value(dc_p[i][j], "%b", value)
                      , 8);
1797
1798        short reff_result = 0;
1799
1800        for (int j = 0; j < width; ++j)
1801           for (int i = 0; i < height; ++i)
1802           {
1803              read_from("../pipes/dc_results", &reff_result, sizeof(short));
1804
1805              if (reff_result != arch_result[i][j])
1806              {
1807                 test_passed = false;
1808
```

```
1809                printf("\n#################␣DC␣ERROR␣DETECTED:␣TEST␣%llu␣-␣Step␣%u
                        ␣-␣INDEX␣[%d,␣%d]␣################\n", test_counter,
                        dc_step_counter, i, j);
1810                printf("Expected␣Value␣␣=␣%d\nGenerated␣Value␣=␣%d\n",
                        reff_result, arch_result[i][j]);
1811             }
1812             else if (debugg)
1813             {
1814                printf("\n#################␣CORRECT␣ON␣DC:␣TEST␣%llu␣-␣Step␣%u␣-␣
                        INDEX␣[%d,␣%d]␣################\n", test_counter,
                        dc_step_counter, i, j);
1815             }
1816          }
1817
1818       dc_step_counter++;
1819
1820       // set_true(&dc_step_checked);
1821    }
1822 }
1823
1824 // Gets and compare the planar predicted block
1825 void get_planar_results ()
1826 {
1827    get_value value;
1828
1829    int condition = convert(acc_fetch_value(check_planar_step, "%b", value), 1);
1830
1831    if (condition)
1832    {
1833       short arch_result[height][width];
1834
1835       for (int i = 0; i < height; ++i)
1836          for (int j = 0; j < width; ++j)
1837             arch_result[i][j] = convert(acc_fetch_value(planar_p[i][j], "%b",
                       value), 8);
1838
1839       short reff_result = 0;
1840
1841       for (int j = 0; j < width; ++j)
1842          for (int i = 0; i < height; ++i)
1843          {
1844             read_from("../pipes/planar_results", &reff_result, sizeof(short));
1845
1846             if (reff_result != arch_result[i][j])
1847             {
1848                test_passed = false;
1849
1850                printf("\n#################␣PLANAR␣ERROR␣DETECTED:␣TEST␣%llu␣-␣
                        Step␣%u␣-␣INDEX␣[%d,␣%d]␣################\n", test_counter,
                        planar_step_counter, i, j);
1851                printf("Expected␣Value␣␣=␣%d\nGenerated␣Value␣=␣%d\n",
                        reff_result, arch_result[i][j]);
1852             }
1853             else if (debugg)
1854             {
1855                printf("\n#################␣CORRECT␣ON␣PLANAR:␣TEST␣%llu␣-␣Step␣%u
                        ␣-␣INDEX␣[%d,␣%d]␣################\n", test_counter,
```

```
                        planar_step_counter, i, j);
1856                }
1857             }
1858
1859          planar_step_counter++;
1860
1861          // set_true(&planar_step_checked);
1862       }
1863    }
1864
1865    // Gets the samples used for predictors
1866    void get_samples ()
1867    {
1868       get_value value;
1869
1870       int condition = convert(acc_fetch_value(read_samples, "%b", value), 1);
1871
1872       if (condition)
1873       {
1874          short pred_sample;
1875
1876          set_value reg_sample;
1877
1878          reg_sample.format = accIntVal;
1879
1880          for (int i = 0; i < width; ++i)
1881          {
1882             read_from("../pipes/samples", &pred_sample, sizeof(short));
1883
1884             reg_sample.value.integer = (int) pred_sample;
1885
1886             acc_set_value(samples_kn[i], &reg_sample, &delay);
1887          }
1888
1889          for (int i = 0; i < width; ++i)
1890          {
1891             read_from("../pipes/samples", &pred_sample, sizeof(short));
1892
1893             reg_sample.value.integer = (int) pred_sample;
1894
1895             acc_set_value(samples_nk[i], &reg_sample, &delay);
1896          }
1897
1898          printf("#######␣reading␣samples␣#######\n");
1899          // set_true(&samples_readed);
1900       }
1901    }
1902
1903    // Gets the default informations for predictors
1904    void get_defs ()
1905    {
1906       get_value value;
1907
1908       int condition = convert(acc_fetch_value(read_def, "%b", value), 1);
1909
1910       if (condition)
1911       {
```

```
1912        short size = 0;
1913        short Nn = 0;
1914        short nN = 0;
1915        short NN = 0;
1916
1917        read_from("../pipes/default", &size, sizeof(short));
1918        read_from("../pipes/default", &Nn, sizeof(short));
1919        read_from("../pipes/default", &nN, sizeof(short));
1920        read_from("../pipes/default", &NN, sizeof(short));
1921
1922        set_value reg_Nn;
1923        set_value reg_nN;
1924        set_value reg_NN;
1925        set_value reg_size;
1926
1927        reg_Nn.format = accIntVal;
1928        reg_nN.format = accIntVal;
1929        reg_NN.format = accIntVal;
1930        reg_size.format = accIntVal;
1931
1932        reg_Nn.value.integer = (int) Nn;
1933        reg_nN.value.integer = (int) nN;
1934        reg_NN.value.integer = (int) NN;
1935        reg_size.value.integer = (int) size;
1936
1937        acc_set_value(sample_Nn, &reg_Nn, &delay);
1938        acc_set_value(sample_nN, &reg_nN, &delay);
1939        acc_set_value(sample_NN, &reg_NN, &delay);
1940        acc_set_value(N, &reg_size, &delay);
1941
1942        printf("#######␣reading␣defaults␣#######\n");
1943        // set_true(&def_readed);
1944    }
1945 }
1946
1947 // Like main function
1948 void intra_monitor ()
1949 {
1950    acc_initialize();
1951    default_initialize();
1952
1953    size_t index = 1;
1954
1955    read_def              = acc_handle_tfarg(index++);
1956    def_readed               = acc_handle_tfarg(index++);
1957    read_samples             = acc_handle_tfarg(index++);
1958    samples_readed           = acc_handle_tfarg(index++);
1959    check_planar_step        = acc_handle_tfarg(index++);
1960    planar_step_checked         = acc_handle_tfarg(index++);
1961    check_dc_step            = acc_handle_tfarg(index++);
1962    dc_step_checked             = acc_handle_tfarg(index++);
1963    check_angular_mode2_step    = acc_handle_tfarg(index++);
1964    angular_mode2_step_checked = acc_handle_tfarg(index++);
1965    check_angular_mode3_step    = acc_handle_tfarg(index++);
1966    angular_mode3_step_checked = acc_handle_tfarg(index++);
1967    check_angular_mode4_step    = acc_handle_tfarg(index++);
1968    angular_mode4_step_checked = acc_handle_tfarg(index++);
```

```
1969     check_angular_mode5_step   = acc_handle_tfarg(index++);
1970     angular_mode5_step_checked = acc_handle_tfarg(index++);
1971     check_angular_mode6_step   = acc_handle_tfarg(index++);
1972     angular_mode6_step_checked = acc_handle_tfarg(index++);
1973     check_angular_mode7_step   = acc_handle_tfarg(index++);
1974     angular_mode7_step_checked = acc_handle_tfarg(index++);
1975     check_angular_mode8_step   = acc_handle_tfarg(index++);
1976     angular_mode8_step_checked = acc_handle_tfarg(index++);
1977     check_angular_mode9_step   = acc_handle_tfarg(index++);
1978     angular_mode9_step_checked = acc_handle_tfarg(index++);
1979     check_angular_mode10_step  = acc_handle_tfarg(index++);
1980     angular_mode10_step_checked   = acc_handle_tfarg(index++);
1981     check_angular_mode11_step  = acc_handle_tfarg(index++);
1982     angular_mode11_step_checked   = acc_handle_tfarg(index++);
1983     check_angular_mode12_step  = acc_handle_tfarg(index++);
1984     angular_mode12_step_checked   = acc_handle_tfarg(index++);
1985     check_angular_mode13_step  = acc_handle_tfarg(index++);
1986     angular_mode13_step_checked   = acc_handle_tfarg(index++);
1987     check_angular_mode14_step  = acc_handle_tfarg(index++);
1988     angular_mode14_step_checked   = acc_handle_tfarg(index++);
1989     check_angular_mode15_step  = acc_handle_tfarg(index++);
1990     angular_mode15_step_checked   = acc_handle_tfarg(index++);
1991     check_angular_mode16_step  = acc_handle_tfarg(index++);
1992     angular_mode16_step_checked   = acc_handle_tfarg(index++);
1993     check_angular_mode17_step  = acc_handle_tfarg(index++);
1994     angular_mode17_step_checked   = acc_handle_tfarg(index++);
1995     check_angular_mode18_step  = acc_handle_tfarg(index++);
1996     angular_mode18_step_checked   = acc_handle_tfarg(index++);
1997     check_angular_mode19_step  = acc_handle_tfarg(index++);
1998     angular_mode19_step_checked   = acc_handle_tfarg(index++);
1999     check_angular_mode20_step  = acc_handle_tfarg(index++);
2000     angular_mode20_step_checked   = acc_handle_tfarg(index++);
2001     check_angular_mode21_step  = acc_handle_tfarg(index++);
2002     angular_mode21_step_checked   = acc_handle_tfarg(index++);
2003     check_angular_mode22_step  = acc_handle_tfarg(index++);
2004     angular_mode22_step_checked   = acc_handle_tfarg(index++);
2005     check_angular_mode23_step  = acc_handle_tfarg(index++);
2006     angular_mode23_step_checked   = acc_handle_tfarg(index++);
2007     check_angular_mode24_step  = acc_handle_tfarg(index++);
2008     angular_mode24_step_checked   = acc_handle_tfarg(index++);
2009     check_angular_mode25_step  = acc_handle_tfarg(index++);
2010     angular_mode25_step_checked   = acc_handle_tfarg(index++);
2011     check_angular_mode26_step  = acc_handle_tfarg(index++);
2012     angular_mode26_step_checked   = acc_handle_tfarg(index++);
2013     check_angular_mode27_step  = acc_handle_tfarg(index++);
2014     angular_mode27_step_checked   = acc_handle_tfarg(index++);
2015     check_angular_mode28_step  = acc_handle_tfarg(index++);
2016     angular_mode28_step_checked   = acc_handle_tfarg(index++);
2017     check_angular_mode29_step  = acc_handle_tfarg(index++);
2018     angular_mode29_step_checked   = acc_handle_tfarg(index++);
2019     check_angular_mode30_step  = acc_handle_tfarg(index++);
2020     angular_mode30_step_checked   = acc_handle_tfarg(index++);
2021     check_angular_mode31_step  = acc_handle_tfarg(index++);
2022     angular_mode31_step_checked   = acc_handle_tfarg(index++);
2023     check_angular_mode32_step  = acc_handle_tfarg(index++);
2024     angular_mode32_step_checked   = acc_handle_tfarg(index++);
2025     check_angular_mode33_step  = acc_handle_tfarg(index++);
```

```
    angular_mode33_step_checked   = acc_handle_tfarg(index++);
    check_angular_mode34_step  = acc_handle_tfarg(index++);
    angular_mode34_step_checked   = acc_handle_tfarg(index++);
    done                  = acc_handle_tfarg(index++);

    for (int i = 0; i < width; ++i)
       samples_kn[i] = acc_handle_tfarg(index++);

    for (int i = 0; i < width; ++i)
       samples_nk[i] = acc_handle_tfarg(index++);

    sample_Nn   = acc_handle_tfarg(index++);
    sample_nN   = acc_handle_tfarg(index++);
    sample_NN   = acc_handle_tfarg(index++);
    N        = acc_handle_tfarg(index++);

    for (int i = 0; i < height; ++i)
       for (int j = 0; j < width; ++j)
          planar_p[i][j] = acc_handle_tfarg(index++);

    for (int i = 0; i < height; ++i)
       for (int j = 0; j < width; ++j)
          dc_p[i][j] = acc_handle_tfarg(index++);

    for (int i = 0; i < height; ++i)
       for (int j = 0; j < width; ++j)
          angular_mode2_p[i][j] = acc_handle_tfarg(index++);

    for (int i = 0; i < height; ++i)
       for (int j = 0; j < width; ++j)
          angular_mode3_p[i][j] = acc_handle_tfarg(index++);

    for (int i = 0; i < height; ++i)
       for (int j = 0; j < width; ++j)
          angular_mode4_p[i][j] = acc_handle_tfarg(index++);

    for (int i = 0; i < height; ++i)
       for (int j = 0; j < width; ++j)
          angular_mode5_p[i][j] = acc_handle_tfarg(index++);

    for (int i = 0; i < height; ++i)
       for (int j = 0; j < width; ++j)
          angular_mode6_p[i][j] = acc_handle_tfarg(index++);

    for (int i = 0; i < height; ++i)
       for (int j = 0; j < width; ++j)
          angular_mode7_p[i][j] = acc_handle_tfarg(index++);

    for (int i = 0; i < height; ++i)
       for (int j = 0; j < width; ++j)
          angular_mode8_p[i][j] = acc_handle_tfarg(index++);

    for (int i = 0; i < height; ++i)
       for (int j = 0; j < width; ++j)
          angular_mode9_p[i][j] = acc_handle_tfarg(index++);

    for (int i = 0; i < height; ++i)
```

```
2083        for (int j = 0; j < width; ++j)
2084            angular_mode10_p[i][j] = acc_handle_tfarg(index++);
2085
2086    for (int i = 0; i < height; ++i)
2087        for (int j = 0; j < width; ++j)
2088            angular_mode11_p[i][j] = acc_handle_tfarg(index++);
2089
2090    for (int i = 0; i < height; ++i)
2091        for (int j = 0; j < width; ++j)
2092            angular_mode12_p[i][j] = acc_handle_tfarg(index++);
2093
2094    for (int i = 0; i < height; ++i)
2095        for (int j = 0; j < width; ++j)
2096            angular_mode13_p[i][j] = acc_handle_tfarg(index++);
2097
2098    for (int i = 0; i < height; ++i)
2099        for (int j = 0; j < width; ++j)
2100            angular_mode14_p[i][j] = acc_handle_tfarg(index++);
2101
2102    for (int i = 0; i < height; ++i)
2103        for (int j = 0; j < width; ++j)
2104            angular_mode15_p[i][j] = acc_handle_tfarg(index++);
2105
2106    for (int i = 0; i < height; ++i)
2107        for (int j = 0; j < width; ++j)
2108            angular_mode16_p[i][j] = acc_handle_tfarg(index++);
2109
2110    for (int i = 0; i < height; ++i)
2111        for (int j = 0; j < width; ++j)
2112            angular_mode17_p[i][j] = acc_handle_tfarg(index++);
2113
2114    for (int i = 0; i < height; ++i)
2115        for (int j = 0; j < width; ++j)
2116            angular_mode18_p[i][j] = acc_handle_tfarg(index++);
2117
2118    for (int i = 0; i < height; ++i)
2119        for (int j = 0; j < width; ++j)
2120            angular_mode19_p[i][j] = acc_handle_tfarg(index++);
2121
2122    for (int i = 0; i < height; ++i)
2123        for (int j = 0; j < width; ++j)
2124            angular_mode20_p[i][j] = acc_handle_tfarg(index++);
2125
2126    for (int i = 0; i < height; ++i)
2127        for (int j = 0; j < width; ++j)
2128            angular_mode21_p[i][j] = acc_handle_tfarg(index++);
2129
2130    for (int i = 0; i < height; ++i)
2131        for (int j = 0; j < width; ++j)
2132            angular_mode22_p[i][j] = acc_handle_tfarg(index++);
2133
2134    for (int i = 0; i < height; ++i)
2135        for (int j = 0; j < width; ++j)
2136            angular_mode23_p[i][j] = acc_handle_tfarg(index++);
2137
2138    for (int i = 0; i < height; ++i)
2139        for (int j = 0; j < width; ++j)
```

```
2140            angular_mode24_p[i][j] = acc_handle_tfarg(index++);
2141
2142        for (int i = 0; i < height; ++i)
2143           for (int j = 0; j < width; ++j)
2144              angular_mode25_p[i][j] = acc_handle_tfarg(index++);
2145
2146        for (int i = 0; i < height; ++i)
2147           for (int j = 0; j < width; ++j)
2148              angular_mode26_p[i][j] = acc_handle_tfarg(index++);
2149
2150        for (int i = 0; i < height; ++i)
2151           for (int j = 0; j < width; ++j)
2152              angular_mode27_p[i][j] = acc_handle_tfarg(index++);
2153
2154        for (int i = 0; i < height; ++i)
2155           for (int j = 0; j < width; ++j)
2156              angular_mode28_p[i][j] = acc_handle_tfarg(index++);
2157
2158        for (int i = 0; i < height; ++i)
2159           for (int j = 0; j < width; ++j)
2160              angular_mode29_p[i][j] = acc_handle_tfarg(index++);
2161
2162        for (int i = 0; i < height; ++i)
2163           for (int j = 0; j < width; ++j)
2164              angular_mode30_p[i][j] = acc_handle_tfarg(index++);
2165
2166        for (int i = 0; i < height; ++i)
2167           for (int j = 0; j < width; ++j)
2168              angular_mode31_p[i][j] = acc_handle_tfarg(index++);
2169
2170        for (int i = 0; i < height; ++i)
2171           for (int j = 0; j < width; ++j)
2172              angular_mode32_p[i][j] = acc_handle_tfarg(index++);
2173
2174        for (int i = 0; i < height; ++i)
2175           for (int j = 0; j < width; ++j)
2176              angular_mode33_p[i][j] = acc_handle_tfarg(index++);
2177
2178        for (int i = 0; i < height; ++i)
2179           for (int j = 0; j < width; ++j)
2180              angular_mode34_p[i][j] = acc_handle_tfarg(index++);
2181
2182        end_simulation = acc_handle_tfarg(index++);
2183
2184        acc_vcl_add(read_def, get_defs, null, vcl_verilog_logic);
2185        acc_vcl_add(read_samples, get_samples, null, vcl_verilog_logic);
2186        acc_vcl_add(check_planar_step, get_planar_results, null, vcl_verilog_logic);
2187        acc_vcl_add(check_dc_step, get_dc_results, null, vcl_verilog_logic);
2188        acc_vcl_add(check_angular_mode2_step, get_angular_mode2_results, null,
               vcl_verilog_logic);
2189        acc_vcl_add(check_angular_mode3_step, get_angular_mode3_results, null,
               vcl_verilog_logic);
2190        acc_vcl_add(check_angular_mode4_step, get_angular_mode4_results, null,
               vcl_verilog_logic);
2191        acc_vcl_add(check_angular_mode5_step, get_angular_mode5_results, null,
               vcl_verilog_logic);
2192        acc_vcl_add(check_angular_mode6_step, get_angular_mode6_results, null,
```

```
        vcl_verilog_logic);
2193    acc_vcl_add(check_angular_mode7_step, get_angular_mode7_results, null,
        vcl_verilog_logic);
2194    acc_vcl_add(check_angular_mode8_step, get_angular_mode8_results, null,
        vcl_verilog_logic);
2195    acc_vcl_add(check_angular_mode9_step, get_angular_mode9_results, null,
        vcl_verilog_logic);
2196    acc_vcl_add(check_angular_mode10_step, get_angular_mode10_results, null,
        vcl_verilog_logic);
2197    acc_vcl_add(check_angular_mode11_step, get_angular_mode11_results, null,
        vcl_verilog_logic);
2198    acc_vcl_add(check_angular_mode12_step, get_angular_mode12_results, null,
        vcl_verilog_logic);
2199    acc_vcl_add(check_angular_mode13_step, get_angular_mode13_results, null,
        vcl_verilog_logic);
2200    acc_vcl_add(check_angular_mode14_step, get_angular_mode14_results, null,
        vcl_verilog_logic);
2201    acc_vcl_add(check_angular_mode15_step, get_angular_mode15_results, null,
        vcl_verilog_logic);
2202    acc_vcl_add(check_angular_mode16_step, get_angular_mode16_results, null,
        vcl_verilog_logic);
2203    acc_vcl_add(check_angular_mode17_step, get_angular_mode17_results, null,
        vcl_verilog_logic);
2204    acc_vcl_add(check_angular_mode18_step, get_angular_mode18_results, null,
        vcl_verilog_logic);
2205    acc_vcl_add(check_angular_mode19_step, get_angular_mode19_results, null,
        vcl_verilog_logic);
2206    acc_vcl_add(check_angular_mode20_step, get_angular_mode20_results, null,
        vcl_verilog_logic);
2207    acc_vcl_add(check_angular_mode21_step, get_angular_mode21_results, null,
        vcl_verilog_logic);
2208    acc_vcl_add(check_angular_mode22_step, get_angular_mode22_results, null,
        vcl_verilog_logic);
2209    acc_vcl_add(check_angular_mode23_step, get_angular_mode23_results, null,
        vcl_verilog_logic);
2210    acc_vcl_add(check_angular_mode24_step, get_angular_mode24_results, null,
        vcl_verilog_logic);
2211    acc_vcl_add(check_angular_mode25_step, get_angular_mode25_results, null,
        vcl_verilog_logic);
2212    acc_vcl_add(check_angular_mode26_step, get_angular_mode26_results, null,
        vcl_verilog_logic);
2213    acc_vcl_add(check_angular_mode27_step, get_angular_mode27_results, null,
        vcl_verilog_logic);
2214    acc_vcl_add(check_angular_mode28_step, get_angular_mode28_results, null,
        vcl_verilog_logic);
2215    acc_vcl_add(check_angular_mode29_step, get_angular_mode29_results, null,
        vcl_verilog_logic);
2216    acc_vcl_add(check_angular_mode30_step, get_angular_mode30_results, null,
        vcl_verilog_logic);
2217    acc_vcl_add(check_angular_mode31_step, get_angular_mode31_results, null,
        vcl_verilog_logic);
2218    acc_vcl_add(check_angular_mode32_step, get_angular_mode32_results, null,
        vcl_verilog_logic);
2219    acc_vcl_add(check_angular_mode33_step, get_angular_mode33_results, null,
        vcl_verilog_logic);
2220    acc_vcl_add(check_angular_mode34_step, get_angular_mode34_results, null,
        vcl_verilog_logic);
```

```
2221
2222     acc_vcl_add(done, loop_finalization, null, vcl_verilog_logic);
2223
2224     acc_close();
2225 }
```

B.3 *TESTBENCH* EM HDL

Listing B.4 – Código fonte do *testbench* em HDL

```
1
2  module testbench;
3
4     parameter H_CLOCK_PERIOD = To_place;
5     parameter CLOCK_PERIOD = H_CLOCK_PERIOD*2;
6     parameter WIDTH = 8;
7
8     reg   ack,
9        clock,
10       reset,
11       start,
12       def_readed,
13       samples_readed,
14       planar_step_checked,
15       dc_step_checked,
16       angular_mode2_step_checked,
17       angular_mode3_step_checked,
18       angular_mode4_step_checked,
19       angular_mode5_step_checked,
20       angular_mode6_step_checked,
21       angular_mode7_step_checked,
22       angular_mode8_step_checked,
23       angular_mode9_step_checked,
24       angular_mode10_step_checked,
25       angular_mode11_step_checked,
26       angular_mode12_step_checked,
27       angular_mode13_step_checked,
28       angular_mode14_step_checked,
29       angular_mode15_step_checked,
30       angular_mode16_step_checked,
31       angular_mode17_step_checked,
32       angular_mode18_step_checked,
33       angular_mode19_step_checked,
34       angular_mode20_step_checked,
35       angular_mode21_step_checked,
36       angular_mode22_step_checked,
37       angular_mode23_step_checked,
38       angular_mode24_step_checked,
39       angular_mode25_step_checked,
40       angular_mode26_step_checked,
41       angular_mode27_step_checked,
42       angular_mode28_step_checked,
43       angular_mode29_step_checked,
44       angular_mode30_step_checked,
45       angular_mode31_step_checked,
46       angular_mode32_step_checked,
```

```
47          angular_mode33_step_checked,
48          angular_mode34_step_checked,
49          end_simulation;
50
51      reg [WIDTH-1:0]    sample_0n,
52                         sample_1n,
53                         sample_2n,
54                         sample_3n,
55                         sample_Nn,
56                         sample_n0,
57                         sample_n1,
58                         sample_n2,
59                         sample_n3,
60                         sample_nN,
61                         sample_NN;
62
63      reg[WIDTH-3:0] N;
64
65      wire   check_planar_step,
66             check_dc_step,
67             check_angular_mode2_step,
68             check_angular_mode3_step,
69             check_angular_mode4_step,
70             check_angular_mode5_step,
71             check_angular_mode6_step,
72             check_angular_mode7_step,
73             check_angular_mode8_step,
74             check_angular_mode9_step,
75             check_angular_mode10_step,
76             check_angular_mode11_step,
77             check_angular_mode12_step,
78             check_angular_mode13_step,
79             check_angular_mode14_step,
80             check_angular_mode15_step,
81             check_angular_mode16_step,
82             check_angular_mode17_step,
83             check_angular_mode18_step,
84             check_angular_mode19_step,
85             check_angular_mode20_step,
86             check_angular_mode21_step,
87             check_angular_mode22_step,
88             check_angular_mode23_step,
89             check_angular_mode24_step,
90             check_angular_mode25_step,
91             check_angular_mode26_step,
92             check_angular_mode27_step,
93             check_angular_mode28_step,
94             check_angular_mode29_step,
95             check_angular_mode30_step,
96             check_angular_mode31_step,
97             check_angular_mode32_step,
98             check_angular_mode33_step,
99             check_angular_mode34_step,
100            read_samples,
101            read_samples_clock_trick,
102            read_def,
103            done;
```

```
104
105     wire [WIDTH-1:0]  planar_p00,
106                       planar_p01,
107                       planar_p02,
108                       planar_p03,
109                       planar_p10,
110                       planar_p11,
111                       planar_p12,
112                       planar_p13,
113                       planar_p20,
114                       planar_p21,
115                       planar_p22,
116                       planar_p23,
117                       planar_p30,
118                       planar_p31,
119                       planar_p32,
120                       planar_p33,
121
122                       dc_p00,
123                       dc_p01,
124                       dc_p02,
125                       dc_p03,
126                       dc_p10,
127                       dc_p11,
128                       dc_p12,
129                       dc_p13,
130                       dc_p20,
131                       dc_p21,
132                       dc_p22,
133                       dc_p23,
134                       dc_p30,
135                       dc_p31,
136                       dc_p32,
137                       dc_p33,
138
139                       angular_mode2_p00,
140                       angular_mode2_p01,
141                       angular_mode2_p02,
142                       angular_mode2_p03,
143                       angular_mode2_p10,
144                       angular_mode2_p11,
145                       angular_mode2_p12,
146                       angular_mode2_p13,
147                       angular_mode2_p20,
148                       angular_mode2_p21,
149                       angular_mode2_p22,
150                       angular_mode2_p23,
151                       angular_mode2_p30,
152                       angular_mode2_p31,
153                       angular_mode2_p32,
154                       angular_mode2_p33,
155
156                       angular_mode3_p00,
157                       angular_mode3_p01,
158                       angular_mode3_p02,
159                       angular_mode3_p03,
160                       angular_mode3_p10,
```

```
161                 angular_mode3_p11,
162                 angular_mode3_p12,
163                 angular_mode3_p13,
164                 angular_mode3_p20,
165                 angular_mode3_p21,
166                 angular_mode3_p22,
167                 angular_mode3_p23,
168                 angular_mode3_p30,
169                 angular_mode3_p31,
170                 angular_mode3_p32,
171                 angular_mode3_p33,
172
173                 angular_mode4_p00,
174                 angular_mode4_p01,
175                 angular_mode4_p02,
176                 angular_mode4_p03,
177                 angular_mode4_p10,
178                 angular_mode4_p11,
179                 angular_mode4_p12,
180                 angular_mode4_p13,
181                 angular_mode4_p20,
182                 angular_mode4_p21,
183                 angular_mode4_p22,
184                 angular_mode4_p23,
185                 angular_mode4_p30,
186                 angular_mode4_p31,
187                 angular_mode4_p32,
188                 angular_mode4_p33,
189
190                 angular_mode5_p00,
191                 angular_mode5_p01,
192                 angular_mode5_p02,
193                 angular_mode5_p03,
194                 angular_mode5_p10,
195                 angular_mode5_p11,
196                 angular_mode5_p12,
197                 angular_mode5_p13,
198                 angular_mode5_p20,
199                 angular_mode5_p21,
200                 angular_mode5_p22,
201                 angular_mode5_p23,
202                 angular_mode5_p30,
203                 angular_mode5_p31,
204                 angular_mode5_p32,
205                 angular_mode5_p33,
206
207                 angular_mode6_p00,
208                 angular_mode6_p01,
209                 angular_mode6_p02,
210                 angular_mode6_p03,
211                 angular_mode6_p10,
212                 angular_mode6_p11,
213                 angular_mode6_p12,
214                 angular_mode6_p13,
215                 angular_mode6_p20,
216                 angular_mode6_p21,
217                 angular_mode6_p22,
```

```
                    angular_mode6_p23,
                    angular_mode6_p30,
                    angular_mode6_p31,
                    angular_mode6_p32,
                    angular_mode6_p33,

                    angular_mode7_p00,
                    angular_mode7_p01,
                    angular_mode7_p02,
                    angular_mode7_p03,
                    angular_mode7_p10,
                    angular_mode7_p11,
                    angular_mode7_p12,
                    angular_mode7_p13,
                    angular_mode7_p20,
                    angular_mode7_p21,
                    angular_mode7_p22,
                    angular_mode7_p23,
                    angular_mode7_p30,
                    angular_mode7_p31,
                    angular_mode7_p32,
                    angular_mode7_p33,

                    angular_mode8_p00,
                    angular_mode8_p01,
                    angular_mode8_p02,
                    angular_mode8_p03,
                    angular_mode8_p10,
                    angular_mode8_p11,
                    angular_mode8_p12,
                    angular_mode8_p13,
                    angular_mode8_p20,
                    angular_mode8_p21,
                    angular_mode8_p22,
                    angular_mode8_p23,
                    angular_mode8_p30,
                    angular_mode8_p31,
                    angular_mode8_p32,
                    angular_mode8_p33,

                    angular_mode9_p00,
                    angular_mode9_p01,
                    angular_mode9_p02,
                    angular_mode9_p03,
                    angular_mode9_p10,
                    angular_mode9_p11,
                    angular_mode9_p12,
                    angular_mode9_p13,
                    angular_mode9_p20,
                    angular_mode9_p21,
                    angular_mode9_p22,
                    angular_mode9_p23,
                    angular_mode9_p30,
                    angular_mode9_p31,
                    angular_mode9_p32,
                    angular_mode9_p33,

```

```
275                 angular_mode10_p00,
276                 angular_mode10_p01,
277                 angular_mode10_p02,
278                 angular_mode10_p03,
279                 angular_mode10_p10,
280                 angular_mode10_p11,
281                 angular_mode10_p12,
282                 angular_mode10_p13,
283                 angular_mode10_p20,
284                 angular_mode10_p21,
285                 angular_mode10_p22,
286                 angular_mode10_p23,
287                 angular_mode10_p30,
288                 angular_mode10_p31,
289                 angular_mode10_p32,
290                 angular_mode10_p33,
291
292                 angular_mode11_p00,
293                 angular_mode11_p01,
294                 angular_mode11_p02,
295                 angular_mode11_p03,
296                 angular_mode11_p10,
297                 angular_mode11_p11,
298                 angular_mode11_p12,
299                 angular_mode11_p13,
300                 angular_mode11_p20,
301                 angular_mode11_p21,
302                 angular_mode11_p22,
303                 angular_mode11_p23,
304                 angular_mode11_p30,
305                 angular_mode11_p31,
306                 angular_mode11_p32,
307                 angular_mode11_p33,
308
309                 angular_mode12_p00,
310                 angular_mode12_p01,
311                 angular_mode12_p02,
312                 angular_mode12_p03,
313                 angular_mode12_p10,
314                 angular_mode12_p11,
315                 angular_mode12_p12,
316                 angular_mode12_p13,
317                 angular_mode12_p20,
318                 angular_mode12_p21,
319                 angular_mode12_p22,
320                 angular_mode12_p23,
321                 angular_mode12_p30,
322                 angular_mode12_p31,
323                 angular_mode12_p32,
324                 angular_mode12_p33,
325
326                 angular_mode13_p00,
327                 angular_mode13_p01,
328                 angular_mode13_p02,
329                 angular_mode13_p03,
330                 angular_mode13_p10,
331                 angular_mode13_p11,
```

```
332                 angular_mode13_p12,
333                 angular_mode13_p13,
334                 angular_mode13_p20,
335                 angular_mode13_p21,
336                 angular_mode13_p22,
337                 angular_mode13_p23,
338                 angular_mode13_p30,
339                 angular_mode13_p31,
340                 angular_mode13_p32,
341                 angular_mode13_p33,
342
343                 angular_mode14_p00,
344                 angular_mode14_p01,
345                 angular_mode14_p02,
346                 angular_mode14_p03,
347                 angular_mode14_p10,
348                 angular_mode14_p11,
349                 angular_mode14_p12,
350                 angular_mode14_p13,
351                 angular_mode14_p20,
352                 angular_mode14_p21,
353                 angular_mode14_p22,
354                 angular_mode14_p23,
355                 angular_mode14_p30,
356                 angular_mode14_p31,
357                 angular_mode14_p32,
358                 angular_mode14_p33,
359
360                 angular_mode15_p00,
361                 angular_mode15_p01,
362                 angular_mode15_p02,
363                 angular_mode15_p03,
364                 angular_mode15_p10,
365                 angular_mode15_p11,
366                 angular_mode15_p12,
367                 angular_mode15_p13,
368                 angular_mode15_p20,
369                 angular_mode15_p21,
370                 angular_mode15_p22,
371                 angular_mode15_p23,
372                 angular_mode15_p30,
373                 angular_mode15_p31,
374                 angular_mode15_p32,
375                 angular_mode15_p33,
376
377                 angular_mode16_p00,
378                 angular_mode16_p01,
379                 angular_mode16_p02,
380                 angular_mode16_p03,
381                 angular_mode16_p10,
382                 angular_mode16_p11,
383                 angular_mode16_p12,
384                 angular_mode16_p13,
385                 angular_mode16_p20,
386                 angular_mode16_p21,
387                 angular_mode16_p22,
388                 angular_mode16_p23,
```

```
389                 angular_mode16_p30,
390                 angular_mode16_p31,
391                 angular_mode16_p32,
392                 angular_mode16_p33,
393
394                 angular_mode17_p00,
395                 angular_mode17_p01,
396                 angular_mode17_p02,
397                 angular_mode17_p03,
398                 angular_mode17_p10,
399                 angular_mode17_p11,
400                 angular_mode17_p12,
401                 angular_mode17_p13,
402                 angular_mode17_p20,
403                 angular_mode17_p21,
404                 angular_mode17_p22,
405                 angular_mode17_p23,
406                 angular_mode17_p30,
407                 angular_mode17_p31,
408                 angular_mode17_p32,
409                 angular_mode17_p33,
410
411                 angular_mode18_p00,
412                 angular_mode18_p01,
413                 angular_mode18_p02,
414                 angular_mode18_p03,
415                 angular_mode18_p10,
416                 angular_mode18_p11,
417                 angular_mode18_p12,
418                 angular_mode18_p13,
419                 angular_mode18_p20,
420                 angular_mode18_p21,
421                 angular_mode18_p22,
422                 angular_mode18_p23,
423                 angular_mode18_p30,
424                 angular_mode18_p31,
425                 angular_mode18_p32,
426                 angular_mode18_p33,
427
428                 angular_mode19_p00,
429                 angular_mode19_p01,
430                 angular_mode19_p02,
431                 angular_mode19_p03,
432                 angular_mode19_p10,
433                 angular_mode19_p11,
434                 angular_mode19_p12,
435                 angular_mode19_p13,
436                 angular_mode19_p20,
437                 angular_mode19_p21,
438                 angular_mode19_p22,
439                 angular_mode19_p23,
440                 angular_mode19_p30,
441                 angular_mode19_p31,
442                 angular_mode19_p32,
443                 angular_mode19_p33,
444
445                 angular_mode20_p00,
```

```
                angular_mode20_p01,
                angular_mode20_p02,
                angular_mode20_p03,
                angular_mode20_p10,
                angular_mode20_p11,
                angular_mode20_p12,
                angular_mode20_p13,
                angular_mode20_p20,
                angular_mode20_p21,
                angular_mode20_p22,
                angular_mode20_p23,
                angular_mode20_p30,
                angular_mode20_p31,
                angular_mode20_p32,
                angular_mode20_p33,

                angular_mode21_p00,
                angular_mode21_p01,
                angular_mode21_p02,
                angular_mode21_p03,
                angular_mode21_p10,
                angular_mode21_p11,
                angular_mode21_p12,
                angular_mode21_p13,
                angular_mode21_p20,
                angular_mode21_p21,
                angular_mode21_p22,
                angular_mode21_p23,
                angular_mode21_p30,
                angular_mode21_p31,
                angular_mode21_p32,
                angular_mode21_p33,

                angular_mode22_p00,
                angular_mode22_p01,
                angular_mode22_p02,
                angular_mode22_p03,
                angular_mode22_p10,
                angular_mode22_p11,
                angular_mode22_p12,
                angular_mode22_p13,
                angular_mode22_p20,
                angular_mode22_p21,
                angular_mode22_p22,
                angular_mode22_p23,
                angular_mode22_p30,
                angular_mode22_p31,
                angular_mode22_p32,
                angular_mode22_p33,

                angular_mode23_p00,
                angular_mode23_p01,
                angular_mode23_p02,
                angular_mode23_p03,
                angular_mode23_p10,
                angular_mode23_p11,
                angular_mode23_p12,
```

```
503                     angular_mode23_p13,
504                     angular_mode23_p20,
505                     angular_mode23_p21,
506                     angular_mode23_p22,
507                     angular_mode23_p23,
508                     angular_mode23_p30,
509                     angular_mode23_p31,
510                     angular_mode23_p32,
511                     angular_mode23_p33,
512
513                     angular_mode24_p00,
514                     angular_mode24_p01,
515                     angular_mode24_p02,
516                     angular_mode24_p03,
517                     angular_mode24_p10,
518                     angular_mode24_p11,
519                     angular_mode24_p12,
520                     angular_mode24_p13,
521                     angular_mode24_p20,
522                     angular_mode24_p21,
523                     angular_mode24_p22,
524                     angular_mode24_p23,
525                     angular_mode24_p30,
526                     angular_mode24_p31,
527                     angular_mode24_p32,
528                     angular_mode24_p33,
529
530                     angular_mode25_p00,
531                     angular_mode25_p01,
532                     angular_mode25_p02,
533                     angular_mode25_p03,
534                     angular_mode25_p10,
535                     angular_mode25_p11,
536                     angular_mode25_p12,
537                     angular_mode25_p13,
538                     angular_mode25_p20,
539                     angular_mode25_p21,
540                     angular_mode25_p22,
541                     angular_mode25_p23,
542                     angular_mode25_p30,
543                     angular_mode25_p31,
544                     angular_mode25_p32,
545                     angular_mode25_p33,
546
547                     angular_mode26_p00,
548                     angular_mode26_p01,
549                     angular_mode26_p02,
550                     angular_mode26_p03,
551                     angular_mode26_p10,
552                     angular_mode26_p11,
553                     angular_mode26_p12,
554                     angular_mode26_p13,
555                     angular_mode26_p20,
556                     angular_mode26_p21,
557                     angular_mode26_p22,
558                     angular_mode26_p23,
559                     angular_mode26_p30,
```

```
560                     angular_mode26_p31,
561                     angular_mode26_p32,
562                     angular_mode26_p33,
563
564                     angular_mode27_p00,
565                     angular_mode27_p01,
566                     angular_mode27_p02,
567                     angular_mode27_p03,
568                     angular_mode27_p10,
569                     angular_mode27_p11,
570                     angular_mode27_p12,
571                     angular_mode27_p13,
572                     angular_mode27_p20,
573                     angular_mode27_p21,
574                     angular_mode27_p22,
575                     angular_mode27_p23,
576                     angular_mode27_p30,
577                     angular_mode27_p31,
578                     angular_mode27_p32,
579                     angular_mode27_p33,
580
581                     angular_mode28_p00,
582                     angular_mode28_p01,
583                     angular_mode28_p02,
584                     angular_mode28_p03,
585                     angular_mode28_p10,
586                     angular_mode28_p11,
587                     angular_mode28_p12,
588                     angular_mode28_p13,
589                     angular_mode28_p20,
590                     angular_mode28_p21,
591                     angular_mode28_p22,
592                     angular_mode28_p23,
593                     angular_mode28_p30,
594                     angular_mode28_p31,
595                     angular_mode28_p32,
596                     angular_mode28_p33,
597
598                     angular_mode29_p00,
599                     angular_mode29_p01,
600                     angular_mode29_p02,
601                     angular_mode29_p03,
602                     angular_mode29_p10,
603                     angular_mode29_p11,
604                     angular_mode29_p12,
605                     angular_mode29_p13,
606                     angular_mode29_p20,
607                     angular_mode29_p21,
608                     angular_mode29_p22,
609                     angular_mode29_p23,
610                     angular_mode29_p30,
611                     angular_mode29_p31,
612                     angular_mode29_p32,
613                     angular_mode29_p33,
614
615                     angular_mode30_p00,
616                     angular_mode30_p01,
```

```
617                 angular_mode30_p02,
618                 angular_mode30_p03,
619                 angular_mode30_p10,
620                 angular_mode30_p11,
621                 angular_mode30_p12,
622                 angular_mode30_p13,
623                 angular_mode30_p20,
624                 angular_mode30_p21,
625                 angular_mode30_p22,
626                 angular_mode30_p23,
627                 angular_mode30_p30,
628                 angular_mode30_p31,
629                 angular_mode30_p32,
630                 angular_mode30_p33,
631
632                 angular_mode31_p00,
633                 angular_mode31_p01,
634                 angular_mode31_p02,
635                 angular_mode31_p03,
636                 angular_mode31_p10,
637                 angular_mode31_p11,
638                 angular_mode31_p12,
639                 angular_mode31_p13,
640                 angular_mode31_p20,
641                 angular_mode31_p21,
642                 angular_mode31_p22,
643                 angular_mode31_p23,
644                 angular_mode31_p30,
645                 angular_mode31_p31,
646                 angular_mode31_p32,
647                 angular_mode31_p33,
648
649                 angular_mode32_p00,
650                 angular_mode32_p01,
651                 angular_mode32_p02,
652                 angular_mode32_p03,
653                 angular_mode32_p10,
654                 angular_mode32_p11,
655                 angular_mode32_p12,
656                 angular_mode32_p13,
657                 angular_mode32_p20,
658                 angular_mode32_p21,
659                 angular_mode32_p22,
660                 angular_mode32_p23,
661                 angular_mode32_p30,
662                 angular_mode32_p31,
663                 angular_mode32_p32,
664                 angular_mode32_p33,
665
666                 angular_mode33_p00,
667                 angular_mode33_p01,
668                 angular_mode33_p02,
669                 angular_mode33_p03,
670                 angular_mode33_p10,
671                 angular_mode33_p11,
672                 angular_mode33_p12,
673                 angular_mode33_p13,
```

```
674                     angular_mode33_p20,
675                     angular_mode33_p21,
676                     angular_mode33_p22,
677                     angular_mode33_p23,
678                     angular_mode33_p30,
679                     angular_mode33_p31,
680                     angular_mode33_p32,
681                     angular_mode33_p33,
682
683                     angular_mode34_p00,
684                     angular_mode34_p01,
685                     angular_mode34_p02,
686                     angular_mode34_p03,
687                     angular_mode34_p10,
688                     angular_mode34_p11,
689                     angular_mode34_p12,
690                     angular_mode34_p13,
691                     angular_mode34_p20,
692                     angular_mode34_p21,
693                     angular_mode34_p22,
694                     angular_mode34_p23,
695                     angular_mode34_p30,
696                     angular_mode34_p31,
697                     angular_mode34_p32,
698                     angular_mode34_p33;
699
700     main_top_level #(WIDTH) intra_predictor(
701        .ack(ack),
702        .clock(clock),
703        .reset(reset),
704        .start(start),
705
706        .sample_0n(sample_0n),
707        .sample_1n(sample_1n),
708        .sample_2n(sample_2n),
709        .sample_3n(sample_3n),
710        .sample_n0(sample_n0),
711        .sample_n1(sample_n1),
712        .sample_n2(sample_n2),
713        .sample_n3(sample_n3),
714
715        .sample_Nn(sample_Nn),
716        .sample_nN(sample_nN),
717        .sample_NN(sample_NN),
718
719        .N(N),
720
721        .read_samples(read_samples),
722        .read_def(read_def),
723        .done(done),
724        .check_planar_step(check_planar_step),
725        .check_dc_step(check_dc_step),
726        .check_angular_mode2_step(check_angular_mode2_step),
727        .check_angular_mode3_step(check_angular_mode3_step),
728        .check_angular_mode4_step(check_angular_mode4_step),
729        .check_angular_mode5_step(check_angular_mode5_step),
730        .check_angular_mode6_step(check_angular_mode6_step),
```

```
731         .check_angular_mode7_step(check_angular_mode7_step),
732         .check_angular_mode8_step(check_angular_mode8_step),
733         .check_angular_mode9_step(check_angular_mode9_step),
734         .check_angular_mode10_step(check_angular_mode10_step),
735         .check_angular_mode11_step(check_angular_mode11_step),
736         .check_angular_mode12_step(check_angular_mode12_step),
737         .check_angular_mode13_step(check_angular_mode13_step),
738         .check_angular_mode14_step(check_angular_mode14_step),
739         .check_angular_mode15_step(check_angular_mode15_step),
740         .check_angular_mode16_step(check_angular_mode16_step),
741         .check_angular_mode17_step(check_angular_mode17_step),
742         .check_angular_mode18_step(check_angular_mode18_step),
743         .check_angular_mode19_step(check_angular_mode19_step),
744         .check_angular_mode20_step(check_angular_mode20_step),
745         .check_angular_mode21_step(check_angular_mode21_step),
746         .check_angular_mode22_step(check_angular_mode22_step),
747         .check_angular_mode23_step(check_angular_mode23_step),
748         .check_angular_mode24_step(check_angular_mode24_step),
749         .check_angular_mode25_step(check_angular_mode25_step),
750         .check_angular_mode26_step(check_angular_mode26_step),
751         .check_angular_mode27_step(check_angular_mode27_step),
752         .check_angular_mode28_step(check_angular_mode28_step),
753         .check_angular_mode29_step(check_angular_mode29_step),
754         .check_angular_mode30_step(check_angular_mode30_step),
755         .check_angular_mode31_step(check_angular_mode31_step),
756         .check_angular_mode32_step(check_angular_mode32_step),
757         .check_angular_mode33_step(check_angular_mode33_step),
758         .check_angular_mode34_step(check_angular_mode34_step),
759
760         .planar_p00(planar_p00),
761         .planar_p01(planar_p01),
762         .planar_p02(planar_p02),
763         .planar_p03(planar_p03),
764         .planar_p10(planar_p10),
765         .planar_p11(planar_p11),
766         .planar_p12(planar_p12),
767         .planar_p13(planar_p13),
768         .planar_p20(planar_p20),
769         .planar_p21(planar_p21),
770         .planar_p22(planar_p22),
771         .planar_p23(planar_p23),
772         .planar_p30(planar_p30),
773         .planar_p31(planar_p31),
774         .planar_p32(planar_p32),
775         .planar_p33(planar_p33),
776
777         .dc_p00(dc_p00),
778         .dc_p01(dc_p01),
779         .dc_p02(dc_p02),
780         .dc_p03(dc_p03),
781         .dc_p10(dc_p10),
782         .dc_p11(dc_p11),
783         .dc_p12(dc_p12),
784         .dc_p13(dc_p13),
785         .dc_p20(dc_p20),
786         .dc_p21(dc_p21),
787         .dc_p22(dc_p22),
```

```
788         .dc_p23(dc_p23),
789         .dc_p30(dc_p30),
790         .dc_p31(dc_p31),
791         .dc_p32(dc_p32),
792         .dc_p33(dc_p33),
793 
794         .angular_mode2_p00(angular_mode2_p00),
795         .angular_mode2_p01(angular_mode2_p01),
796         .angular_mode2_p02(angular_mode2_p02),
797         .angular_mode2_p03(angular_mode2_p03),
798         .angular_mode2_p10(angular_mode2_p10),
799         .angular_mode2_p11(angular_mode2_p11),
800         .angular_mode2_p12(angular_mode2_p12),
801         .angular_mode2_p13(angular_mode2_p13),
802         .angular_mode2_p20(angular_mode2_p20),
803         .angular_mode2_p21(angular_mode2_p21),
804         .angular_mode2_p22(angular_mode2_p22),
805         .angular_mode2_p23(angular_mode2_p23),
806         .angular_mode2_p30(angular_mode2_p30),
807         .angular_mode2_p31(angular_mode2_p31),
808         .angular_mode2_p32(angular_mode2_p32),
809         .angular_mode2_p33(angular_mode2_p33),
810 
811         .angular_mode3_p00(angular_mode3_p00),
812         .angular_mode3_p01(angular_mode3_p01),
813         .angular_mode3_p02(angular_mode3_p02),
814         .angular_mode3_p03(angular_mode3_p03),
815         .angular_mode3_p10(angular_mode3_p10),
816         .angular_mode3_p11(angular_mode3_p11),
817         .angular_mode3_p12(angular_mode3_p12),
818         .angular_mode3_p13(angular_mode3_p13),
819         .angular_mode3_p20(angular_mode3_p20),
820         .angular_mode3_p21(angular_mode3_p21),
821         .angular_mode3_p22(angular_mode3_p22),
822         .angular_mode3_p23(angular_mode3_p23),
823         .angular_mode3_p30(angular_mode3_p30),
824         .angular_mode3_p31(angular_mode3_p31),
825         .angular_mode3_p32(angular_mode3_p32),
826         .angular_mode3_p33(angular_mode3_p33),
827 
828         .angular_mode4_p00(angular_mode4_p00),
829         .angular_mode4_p01(angular_mode4_p01),
830         .angular_mode4_p02(angular_mode4_p02),
831         .angular_mode4_p03(angular_mode4_p03),
832         .angular_mode4_p10(angular_mode4_p10),
833         .angular_mode4_p11(angular_mode4_p11),
834         .angular_mode4_p12(angular_mode4_p12),
835         .angular_mode4_p13(angular_mode4_p13),
836         .angular_mode4_p20(angular_mode4_p20),
837         .angular_mode4_p21(angular_mode4_p21),
838         .angular_mode4_p22(angular_mode4_p22),
839         .angular_mode4_p23(angular_mode4_p23),
840         .angular_mode4_p30(angular_mode4_p30),
841         .angular_mode4_p31(angular_mode4_p31),
842         .angular_mode4_p32(angular_mode4_p32),
843         .angular_mode4_p33(angular_mode4_p33),
844 
```

```
845         .angular_mode5_p00(angular_mode5_p00),
846         .angular_mode5_p01(angular_mode5_p01),
847         .angular_mode5_p02(angular_mode5_p02),
848         .angular_mode5_p03(angular_mode5_p03),
849         .angular_mode5_p10(angular_mode5_p10),
850         .angular_mode5_p11(angular_mode5_p11),
851         .angular_mode5_p12(angular_mode5_p12),
852         .angular_mode5_p13(angular_mode5_p13),
853         .angular_mode5_p20(angular_mode5_p20),
854         .angular_mode5_p21(angular_mode5_p21),
855         .angular_mode5_p22(angular_mode5_p22),
856         .angular_mode5_p23(angular_mode5_p23),
857         .angular_mode5_p30(angular_mode5_p30),
858         .angular_mode5_p31(angular_mode5_p31),
859         .angular_mode5_p32(angular_mode5_p32),
860         .angular_mode5_p33(angular_mode5_p33),
861
862         .angular_mode6_p00(angular_mode6_p00),
863         .angular_mode6_p01(angular_mode6_p01),
864         .angular_mode6_p02(angular_mode6_p02),
865         .angular_mode6_p03(angular_mode6_p03),
866         .angular_mode6_p10(angular_mode6_p10),
867         .angular_mode6_p11(angular_mode6_p11),
868         .angular_mode6_p12(angular_mode6_p12),
869         .angular_mode6_p13(angular_mode6_p13),
870         .angular_mode6_p20(angular_mode6_p20),
871         .angular_mode6_p21(angular_mode6_p21),
872         .angular_mode6_p22(angular_mode6_p22),
873         .angular_mode6_p23(angular_mode6_p23),
874         .angular_mode6_p30(angular_mode6_p30),
875         .angular_mode6_p31(angular_mode6_p31),
876         .angular_mode6_p32(angular_mode6_p32),
877         .angular_mode6_p33(angular_mode6_p33),
878
879         .angular_mode7_p00(angular_mode7_p00),
880         .angular_mode7_p01(angular_mode7_p01),
881         .angular_mode7_p02(angular_mode7_p02),
882         .angular_mode7_p03(angular_mode7_p03),
883         .angular_mode7_p10(angular_mode7_p10),
884         .angular_mode7_p11(angular_mode7_p11),
885         .angular_mode7_p12(angular_mode7_p12),
886         .angular_mode7_p13(angular_mode7_p13),
887         .angular_mode7_p20(angular_mode7_p20),
888         .angular_mode7_p21(angular_mode7_p21),
889         .angular_mode7_p22(angular_mode7_p22),
890         .angular_mode7_p23(angular_mode7_p23),
891         .angular_mode7_p30(angular_mode7_p30),
892         .angular_mode7_p31(angular_mode7_p31),
893         .angular_mode7_p32(angular_mode7_p32),
894         .angular_mode7_p33(angular_mode7_p33),
895
896         .angular_mode8_p00(angular_mode8_p00),
897         .angular_mode8_p01(angular_mode8_p01),
898         .angular_mode8_p02(angular_mode8_p02),
899         .angular_mode8_p03(angular_mode8_p03),
900         .angular_mode8_p10(angular_mode8_p10),
901         .angular_mode8_p11(angular_mode8_p11),
```

```
902         .angular_mode8_p12(angular_mode8_p12),
903         .angular_mode8_p13(angular_mode8_p13),
904         .angular_mode8_p20(angular_mode8_p20),
905         .angular_mode8_p21(angular_mode8_p21),
906         .angular_mode8_p22(angular_mode8_p22),
907         .angular_mode8_p23(angular_mode8_p23),
908         .angular_mode8_p30(angular_mode8_p30),
909         .angular_mode8_p31(angular_mode8_p31),
910         .angular_mode8_p32(angular_mode8_p32),
911         .angular_mode8_p33(angular_mode8_p33),
912
913         .angular_mode9_p00(angular_mode9_p00),
914         .angular_mode9_p01(angular_mode9_p01),
915         .angular_mode9_p02(angular_mode9_p02),
916         .angular_mode9_p03(angular_mode9_p03),
917         .angular_mode9_p10(angular_mode9_p10),
918         .angular_mode9_p11(angular_mode9_p11),
919         .angular_mode9_p12(angular_mode9_p12),
920         .angular_mode9_p13(angular_mode9_p13),
921         .angular_mode9_p20(angular_mode9_p20),
922         .angular_mode9_p21(angular_mode9_p21),
923         .angular_mode9_p22(angular_mode9_p22),
924         .angular_mode9_p23(angular_mode9_p23),
925         .angular_mode9_p30(angular_mode9_p30),
926         .angular_mode9_p31(angular_mode9_p31),
927         .angular_mode9_p32(angular_mode9_p32),
928         .angular_mode9_p33(angular_mode9_p33),
929
930         .angular_mode10_p00(angular_mode10_p00),
931         .angular_mode10_p01(angular_mode10_p01),
932         .angular_mode10_p02(angular_mode10_p02),
933         .angular_mode10_p03(angular_mode10_p03),
934         .angular_mode10_p10(angular_mode10_p10),
935         .angular_mode10_p11(angular_mode10_p11),
936         .angular_mode10_p12(angular_mode10_p12),
937         .angular_mode10_p13(angular_mode10_p13),
938         .angular_mode10_p20(angular_mode10_p20),
939         .angular_mode10_p21(angular_mode10_p21),
940         .angular_mode10_p22(angular_mode10_p22),
941         .angular_mode10_p23(angular_mode10_p23),
942         .angular_mode10_p30(angular_mode10_p30),
943         .angular_mode10_p31(angular_mode10_p31),
944         .angular_mode10_p32(angular_mode10_p32),
945         .angular_mode10_p33(angular_mode10_p33),
946
947         .angular_mode11_p00(angular_mode11_p00),
948         .angular_mode11_p01(angular_mode11_p01),
949         .angular_mode11_p02(angular_mode11_p02),
950         .angular_mode11_p03(angular_mode11_p03),
951         .angular_mode11_p10(angular_mode11_p10),
952         .angular_mode11_p11(angular_mode11_p11),
953         .angular_mode11_p12(angular_mode11_p12),
954         .angular_mode11_p13(angular_mode11_p13),
955         .angular_mode11_p20(angular_mode11_p20),
956         .angular_mode11_p21(angular_mode11_p21),
957         .angular_mode11_p22(angular_mode11_p22),
958         .angular_mode11_p23(angular_mode11_p23),
```

```
959         .angular_mode11_p30(angular_mode11_p30),
960         .angular_mode11_p31(angular_mode11_p31),
961         .angular_mode11_p32(angular_mode11_p32),
962         .angular_mode11_p33(angular_mode11_p33),
963
964         .angular_mode12_p00(angular_mode12_p00),
965         .angular_mode12_p01(angular_mode12_p01),
966         .angular_mode12_p02(angular_mode12_p02),
967         .angular_mode12_p03(angular_mode12_p03),
968         .angular_mode12_p10(angular_mode12_p10),
969         .angular_mode12_p11(angular_mode12_p11),
970         .angular_mode12_p12(angular_mode12_p12),
971         .angular_mode12_p13(angular_mode12_p13),
972         .angular_mode12_p20(angular_mode12_p20),
973         .angular_mode12_p21(angular_mode12_p21),
974         .angular_mode12_p22(angular_mode12_p22),
975         .angular_mode12_p23(angular_mode12_p23),
976         .angular_mode12_p30(angular_mode12_p30),
977         .angular_mode12_p31(angular_mode12_p31),
978         .angular_mode12_p32(angular_mode12_p32),
979         .angular_mode12_p33(angular_mode12_p33),
980
981         .angular_mode13_p00(angular_mode13_p00),
982         .angular_mode13_p01(angular_mode13_p01),
983         .angular_mode13_p02(angular_mode13_p02),
984         .angular_mode13_p03(angular_mode13_p03),
985         .angular_mode13_p10(angular_mode13_p10),
986         .angular_mode13_p11(angular_mode13_p11),
987         .angular_mode13_p12(angular_mode13_p12),
988         .angular_mode13_p13(angular_mode13_p13),
989         .angular_mode13_p20(angular_mode13_p20),
990         .angular_mode13_p21(angular_mode13_p21),
991         .angular_mode13_p22(angular_mode13_p22),
992         .angular_mode13_p23(angular_mode13_p23),
993         .angular_mode13_p30(angular_mode13_p30),
994         .angular_mode13_p31(angular_mode13_p31),
995         .angular_mode13_p32(angular_mode13_p32),
996         .angular_mode13_p33(angular_mode13_p33),
997
998         .angular_mode14_p00(angular_mode14_p00),
999         .angular_mode14_p01(angular_mode14_p01),
1000        .angular_mode14_p02(angular_mode14_p02),
1001        .angular_mode14_p03(angular_mode14_p03),
1002        .angular_mode14_p10(angular_mode14_p10),
1003        .angular_mode14_p11(angular_mode14_p11),
1004        .angular_mode14_p12(angular_mode14_p12),
1005        .angular_mode14_p13(angular_mode14_p13),
1006        .angular_mode14_p20(angular_mode14_p20),
1007        .angular_mode14_p21(angular_mode14_p21),
1008        .angular_mode14_p22(angular_mode14_p22),
1009        .angular_mode14_p23(angular_mode14_p23),
1010        .angular_mode14_p30(angular_mode14_p30),
1011        .angular_mode14_p31(angular_mode14_p31),
1012        .angular_mode14_p32(angular_mode14_p32),
1013        .angular_mode14_p33(angular_mode14_p33),
1014
1015        .angular_mode15_p00(angular_mode15_p00),
```

```
1016        .angular_mode15_p01(angular_mode15_p01),
1017        .angular_mode15_p02(angular_mode15_p02),
1018        .angular_mode15_p03(angular_mode15_p03),
1019        .angular_mode15_p10(angular_mode15_p10),
1020        .angular_mode15_p11(angular_mode15_p11),
1021        .angular_mode15_p12(angular_mode15_p12),
1022        .angular_mode15_p13(angular_mode15_p13),
1023        .angular_mode15_p20(angular_mode15_p20),
1024        .angular_mode15_p21(angular_mode15_p21),
1025        .angular_mode15_p22(angular_mode15_p22),
1026        .angular_mode15_p23(angular_mode15_p23),
1027        .angular_mode15_p30(angular_mode15_p30),
1028        .angular_mode15_p31(angular_mode15_p31),
1029        .angular_mode15_p32(angular_mode15_p32),
1030        .angular_mode15_p33(angular_mode15_p33),
1031
1032        .angular_mode16_p00(angular_mode16_p00),
1033        .angular_mode16_p01(angular_mode16_p01),
1034        .angular_mode16_p02(angular_mode16_p02),
1035        .angular_mode16_p03(angular_mode16_p03),
1036        .angular_mode16_p10(angular_mode16_p10),
1037        .angular_mode16_p11(angular_mode16_p11),
1038        .angular_mode16_p12(angular_mode16_p12),
1039        .angular_mode16_p13(angular_mode16_p13),
1040        .angular_mode16_p20(angular_mode16_p20),
1041        .angular_mode16_p21(angular_mode16_p21),
1042        .angular_mode16_p22(angular_mode16_p22),
1043        .angular_mode16_p23(angular_mode16_p23),
1044        .angular_mode16_p30(angular_mode16_p30),
1045        .angular_mode16_p31(angular_mode16_p31),
1046        .angular_mode16_p32(angular_mode16_p32),
1047        .angular_mode16_p33(angular_mode16_p33),
1048
1049        .angular_mode17_p00(angular_mode17_p00),
1050        .angular_mode17_p01(angular_mode17_p01),
1051        .angular_mode17_p02(angular_mode17_p02),
1052        .angular_mode17_p03(angular_mode17_p03),
1053        .angular_mode17_p10(angular_mode17_p10),
1054        .angular_mode17_p11(angular_mode17_p11),
1055        .angular_mode17_p12(angular_mode17_p12),
1056        .angular_mode17_p13(angular_mode17_p13),
1057        .angular_mode17_p20(angular_mode17_p20),
1058        .angular_mode17_p21(angular_mode17_p21),
1059        .angular_mode17_p22(angular_mode17_p22),
1060        .angular_mode17_p23(angular_mode17_p23),
1061        .angular_mode17_p30(angular_mode17_p30),
1062        .angular_mode17_p31(angular_mode17_p31),
1063        .angular_mode17_p32(angular_mode17_p32),
1064        .angular_mode17_p33(angular_mode17_p33),
1065
1066        .angular_mode18_p00(angular_mode18_p00),
1067        .angular_mode18_p01(angular_mode18_p01),
1068        .angular_mode18_p02(angular_mode18_p02),
1069        .angular_mode18_p03(angular_mode18_p03),
1070        .angular_mode18_p10(angular_mode18_p10),
1071        .angular_mode18_p11(angular_mode18_p11),
1072        .angular_mode18_p12(angular_mode18_p12),
```

```
1073        .angular_mode18_p13(angular_mode18_p13),
1074        .angular_mode18_p20(angular_mode18_p20),
1075        .angular_mode18_p21(angular_mode18_p21),
1076        .angular_mode18_p22(angular_mode18_p22),
1077        .angular_mode18_p23(angular_mode18_p23),
1078        .angular_mode18_p30(angular_mode18_p30),
1079        .angular_mode18_p31(angular_mode18_p31),
1080        .angular_mode18_p32(angular_mode18_p32),
1081        .angular_mode18_p33(angular_mode18_p33),
1082
1083        .angular_mode19_p00(angular_mode19_p00),
1084        .angular_mode19_p01(angular_mode19_p01),
1085        .angular_mode19_p02(angular_mode19_p02),
1086        .angular_mode19_p03(angular_mode19_p03),
1087        .angular_mode19_p10(angular_mode19_p10),
1088        .angular_mode19_p11(angular_mode19_p11),
1089        .angular_mode19_p12(angular_mode19_p12),
1090        .angular_mode19_p13(angular_mode19_p13),
1091        .angular_mode19_p20(angular_mode19_p20),
1092        .angular_mode19_p21(angular_mode19_p21),
1093        .angular_mode19_p22(angular_mode19_p22),
1094        .angular_mode19_p23(angular_mode19_p23),
1095        .angular_mode19_p30(angular_mode19_p30),
1096        .angular_mode19_p31(angular_mode19_p31),
1097        .angular_mode19_p32(angular_mode19_p32),
1098        .angular_mode19_p33(angular_mode19_p33),
1099
1100        .angular_mode20_p00(angular_mode20_p00),
1101        .angular_mode20_p01(angular_mode20_p01),
1102        .angular_mode20_p02(angular_mode20_p02),
1103        .angular_mode20_p03(angular_mode20_p03),
1104        .angular_mode20_p10(angular_mode20_p10),
1105        .angular_mode20_p11(angular_mode20_p11),
1106        .angular_mode20_p12(angular_mode20_p12),
1107        .angular_mode20_p13(angular_mode20_p13),
1108        .angular_mode20_p20(angular_mode20_p20),
1109        .angular_mode20_p21(angular_mode20_p21),
1110        .angular_mode20_p22(angular_mode20_p22),
1111        .angular_mode20_p23(angular_mode20_p23),
1112        .angular_mode20_p30(angular_mode20_p30),
1113        .angular_mode20_p31(angular_mode20_p31),
1114        .angular_mode20_p32(angular_mode20_p32),
1115        .angular_mode20_p33(angular_mode20_p33),
1116
1117        .angular_mode21_p00(angular_mode21_p00),
1118        .angular_mode21_p01(angular_mode21_p01),
1119        .angular_mode21_p02(angular_mode21_p02),
1120        .angular_mode21_p03(angular_mode21_p03),
1121        .angular_mode21_p10(angular_mode21_p10),
1122        .angular_mode21_p11(angular_mode21_p11),
1123        .angular_mode21_p12(angular_mode21_p12),
1124        .angular_mode21_p13(angular_mode21_p13),
1125        .angular_mode21_p20(angular_mode21_p20),
1126        .angular_mode21_p21(angular_mode21_p21),
1127        .angular_mode21_p22(angular_mode21_p22),
1128        .angular_mode21_p23(angular_mode21_p23),
1129        .angular_mode21_p30(angular_mode21_p30),
```

```
1130        .angular_mode21_p31(angular_mode21_p31),
1131        .angular_mode21_p32(angular_mode21_p32),
1132        .angular_mode21_p33(angular_mode21_p33),
1133
1134        .angular_mode22_p00(angular_mode22_p00),
1135        .angular_mode22_p01(angular_mode22_p01),
1136        .angular_mode22_p02(angular_mode22_p02),
1137        .angular_mode22_p03(angular_mode22_p03),
1138        .angular_mode22_p10(angular_mode22_p10),
1139        .angular_mode22_p11(angular_mode22_p11),
1140        .angular_mode22_p12(angular_mode22_p12),
1141        .angular_mode22_p13(angular_mode22_p13),
1142        .angular_mode22_p20(angular_mode22_p20),
1143        .angular_mode22_p21(angular_mode22_p21),
1144        .angular_mode22_p22(angular_mode22_p22),
1145        .angular_mode22_p23(angular_mode22_p23),
1146        .angular_mode22_p30(angular_mode22_p30),
1147        .angular_mode22_p31(angular_mode22_p31),
1148        .angular_mode22_p32(angular_mode22_p32),
1149        .angular_mode22_p33(angular_mode22_p33),
1150
1151        .angular_mode23_p00(angular_mode23_p00),
1152        .angular_mode23_p01(angular_mode23_p01),
1153        .angular_mode23_p02(angular_mode23_p02),
1154        .angular_mode23_p03(angular_mode23_p03),
1155        .angular_mode23_p10(angular_mode23_p10),
1156        .angular_mode23_p11(angular_mode23_p11),
1157        .angular_mode23_p12(angular_mode23_p12),
1158        .angular_mode23_p13(angular_mode23_p13),
1159        .angular_mode23_p20(angular_mode23_p20),
1160        .angular_mode23_p21(angular_mode23_p21),
1161        .angular_mode23_p22(angular_mode23_p22),
1162        .angular_mode23_p23(angular_mode23_p23),
1163        .angular_mode23_p30(angular_mode23_p30),
1164        .angular_mode23_p31(angular_mode23_p31),
1165        .angular_mode23_p32(angular_mode23_p32),
1166        .angular_mode23_p33(angular_mode23_p33),
1167
1168        .angular_mode24_p00(angular_mode24_p00),
1169        .angular_mode24_p01(angular_mode24_p01),
1170        .angular_mode24_p02(angular_mode24_p02),
1171        .angular_mode24_p03(angular_mode24_p03),
1172        .angular_mode24_p10(angular_mode24_p10),
1173        .angular_mode24_p11(angular_mode24_p11),
1174        .angular_mode24_p12(angular_mode24_p12),
1175        .angular_mode24_p13(angular_mode24_p13),
1176        .angular_mode24_p20(angular_mode24_p20),
1177        .angular_mode24_p21(angular_mode24_p21),
1178        .angular_mode24_p22(angular_mode24_p22),
1179        .angular_mode24_p23(angular_mode24_p23),
1180        .angular_mode24_p30(angular_mode24_p30),
1181        .angular_mode24_p31(angular_mode24_p31),
1182        .angular_mode24_p32(angular_mode24_p32),
1183        .angular_mode24_p33(angular_mode24_p33),
1184
1185        .angular_mode25_p00(angular_mode25_p00),
1186        .angular_mode25_p01(angular_mode25_p01),
```

```
1187        .angular_mode25_p02(angular_mode25_p02),
1188        .angular_mode25_p03(angular_mode25_p03),
1189        .angular_mode25_p10(angular_mode25_p10),
1190        .angular_mode25_p11(angular_mode25_p11),
1191        .angular_mode25_p12(angular_mode25_p12),
1192        .angular_mode25_p13(angular_mode25_p13),
1193        .angular_mode25_p20(angular_mode25_p20),
1194        .angular_mode25_p21(angular_mode25_p21),
1195        .angular_mode25_p22(angular_mode25_p22),
1196        .angular_mode25_p23(angular_mode25_p23),
1197        .angular_mode25_p30(angular_mode25_p30),
1198        .angular_mode25_p31(angular_mode25_p31),
1199        .angular_mode25_p32(angular_mode25_p32),
1200        .angular_mode25_p33(angular_mode25_p33),
1201
1202        .angular_mode26_p00(angular_mode26_p00),
1203        .angular_mode26_p01(angular_mode26_p01),
1204        .angular_mode26_p02(angular_mode26_p02),
1205        .angular_mode26_p03(angular_mode26_p03),
1206        .angular_mode26_p10(angular_mode26_p10),
1207        .angular_mode26_p11(angular_mode26_p11),
1208        .angular_mode26_p12(angular_mode26_p12),
1209        .angular_mode26_p13(angular_mode26_p13),
1210        .angular_mode26_p20(angular_mode26_p20),
1211        .angular_mode26_p21(angular_mode26_p21),
1212        .angular_mode26_p22(angular_mode26_p22),
1213        .angular_mode26_p23(angular_mode26_p23),
1214        .angular_mode26_p30(angular_mode26_p30),
1215        .angular_mode26_p31(angular_mode26_p31),
1216        .angular_mode26_p32(angular_mode26_p32),
1217        .angular_mode26_p33(angular_mode26_p33),
1218
1219        .angular_mode27_p00(angular_mode27_p00),
1220        .angular_mode27_p01(angular_mode27_p01),
1221        .angular_mode27_p02(angular_mode27_p02),
1222        .angular_mode27_p03(angular_mode27_p03),
1223        .angular_mode27_p10(angular_mode27_p10),
1224        .angular_mode27_p11(angular_mode27_p11),
1225        .angular_mode27_p12(angular_mode27_p12),
1226        .angular_mode27_p13(angular_mode27_p13),
1227        .angular_mode27_p20(angular_mode27_p20),
1228        .angular_mode27_p21(angular_mode27_p21),
1229        .angular_mode27_p22(angular_mode27_p22),
1230        .angular_mode27_p23(angular_mode27_p23),
1231        .angular_mode27_p30(angular_mode27_p30),
1232        .angular_mode27_p31(angular_mode27_p31),
1233        .angular_mode27_p32(angular_mode27_p32),
1234        .angular_mode27_p33(angular_mode27_p33),
1235
1236        .angular_mode28_p00(angular_mode28_p00),
1237        .angular_mode28_p01(angular_mode28_p01),
1238        .angular_mode28_p02(angular_mode28_p02),
1239        .angular_mode28_p03(angular_mode28_p03),
1240        .angular_mode28_p10(angular_mode28_p10),
1241        .angular_mode28_p11(angular_mode28_p11),
1242        .angular_mode28_p12(angular_mode28_p12),
1243        .angular_mode28_p13(angular_mode28_p13),
```

```
1244        .angular_mode28_p20(angular_mode28_p20),
1245        .angular_mode28_p21(angular_mode28_p21),
1246        .angular_mode28_p22(angular_mode28_p22),
1247        .angular_mode28_p23(angular_mode28_p23),
1248        .angular_mode28_p30(angular_mode28_p30),
1249        .angular_mode28_p31(angular_mode28_p31),
1250        .angular_mode28_p32(angular_mode28_p32),
1251        .angular_mode28_p33(angular_mode28_p33),
1252
1253        .angular_mode29_p00(angular_mode29_p00),
1254        .angular_mode29_p01(angular_mode29_p01),
1255        .angular_mode29_p02(angular_mode29_p02),
1256        .angular_mode29_p03(angular_mode29_p03),
1257        .angular_mode29_p10(angular_mode29_p10),
1258        .angular_mode29_p11(angular_mode29_p11),
1259        .angular_mode29_p12(angular_mode29_p12),
1260        .angular_mode29_p13(angular_mode29_p13),
1261        .angular_mode29_p20(angular_mode29_p20),
1262        .angular_mode29_p21(angular_mode29_p21),
1263        .angular_mode29_p22(angular_mode29_p22),
1264        .angular_mode29_p23(angular_mode29_p23),
1265        .angular_mode29_p30(angular_mode29_p30),
1266        .angular_mode29_p31(angular_mode29_p31),
1267        .angular_mode29_p32(angular_mode29_p32),
1268        .angular_mode29_p33(angular_mode29_p33),
1269
1270        .angular_mode30_p00(angular_mode30_p00),
1271        .angular_mode30_p01(angular_mode30_p01),
1272        .angular_mode30_p02(angular_mode30_p02),
1273        .angular_mode30_p03(angular_mode30_p03),
1274        .angular_mode30_p10(angular_mode30_p10),
1275        .angular_mode30_p11(angular_mode30_p11),
1276        .angular_mode30_p12(angular_mode30_p12),
1277        .angular_mode30_p13(angular_mode30_p13),
1278        .angular_mode30_p20(angular_mode30_p20),
1279        .angular_mode30_p21(angular_mode30_p21),
1280        .angular_mode30_p22(angular_mode30_p22),
1281        .angular_mode30_p23(angular_mode30_p23),
1282        .angular_mode30_p30(angular_mode30_p30),
1283        .angular_mode30_p31(angular_mode30_p31),
1284        .angular_mode30_p32(angular_mode30_p32),
1285        .angular_mode30_p33(angular_mode30_p33),
1286
1287        .angular_mode31_p00(angular_mode31_p00),
1288        .angular_mode31_p01(angular_mode31_p01),
1289        .angular_mode31_p02(angular_mode31_p02),
1290        .angular_mode31_p03(angular_mode31_p03),
1291        .angular_mode31_p10(angular_mode31_p10),
1292        .angular_mode31_p11(angular_mode31_p11),
1293        .angular_mode31_p12(angular_mode31_p12),
1294        .angular_mode31_p13(angular_mode31_p13),
1295        .angular_mode31_p20(angular_mode31_p20),
1296        .angular_mode31_p21(angular_mode31_p21),
1297        .angular_mode31_p22(angular_mode31_p22),
1298        .angular_mode31_p23(angular_mode31_p23),
1299        .angular_mode31_p30(angular_mode31_p30),
1300        .angular_mode31_p31(angular_mode31_p31),
```

```
1301        .angular_mode31_p32(angular_mode31_p32),
1302        .angular_mode31_p33(angular_mode31_p33),
1303
1304        .angular_mode32_p00(angular_mode32_p00),
1305        .angular_mode32_p01(angular_mode32_p01),
1306        .angular_mode32_p02(angular_mode32_p02),
1307        .angular_mode32_p03(angular_mode32_p03),
1308        .angular_mode32_p10(angular_mode32_p10),
1309        .angular_mode32_p11(angular_mode32_p11),
1310        .angular_mode32_p12(angular_mode32_p12),
1311        .angular_mode32_p13(angular_mode32_p13),
1312        .angular_mode32_p20(angular_mode32_p20),
1313        .angular_mode32_p21(angular_mode32_p21),
1314        .angular_mode32_p22(angular_mode32_p22),
1315        .angular_mode32_p23(angular_mode32_p23),
1316        .angular_mode32_p30(angular_mode32_p30),
1317        .angular_mode32_p31(angular_mode32_p31),
1318        .angular_mode32_p32(angular_mode32_p32),
1319        .angular_mode32_p33(angular_mode32_p33),
1320
1321        .angular_mode33_p00(angular_mode33_p00),
1322        .angular_mode33_p01(angular_mode33_p01),
1323        .angular_mode33_p02(angular_mode33_p02),
1324        .angular_mode33_p03(angular_mode33_p03),
1325        .angular_mode33_p10(angular_mode33_p10),
1326        .angular_mode33_p11(angular_mode33_p11),
1327        .angular_mode33_p12(angular_mode33_p12),
1328        .angular_mode33_p13(angular_mode33_p13),
1329        .angular_mode33_p20(angular_mode33_p20),
1330        .angular_mode33_p21(angular_mode33_p21),
1331        .angular_mode33_p22(angular_mode33_p22),
1332        .angular_mode33_p23(angular_mode33_p23),
1333        .angular_mode33_p30(angular_mode33_p30),
1334        .angular_mode33_p31(angular_mode33_p31),
1335        .angular_mode33_p32(angular_mode33_p32),
1336        .angular_mode33_p33(angular_mode33_p33),
1337
1338        .angular_mode34_p00(angular_mode34_p00),
1339        .angular_mode34_p01(angular_mode34_p01),
1340        .angular_mode34_p02(angular_mode34_p02),
1341        .angular_mode34_p03(angular_mode34_p03),
1342        .angular_mode34_p10(angular_mode34_p10),
1343        .angular_mode34_p11(angular_mode34_p11),
1344        .angular_mode34_p12(angular_mode34_p12),
1345        .angular_mode34_p13(angular_mode34_p13),
1346        .angular_mode34_p20(angular_mode34_p20),
1347        .angular_mode34_p21(angular_mode34_p21),
1348        .angular_mode34_p22(angular_mode34_p22),
1349        .angular_mode34_p23(angular_mode34_p23),
1350        .angular_mode34_p30(angular_mode34_p30),
1351        .angular_mode34_p31(angular_mode34_p31),
1352        .angular_mode34_p32(angular_mode34_p32),
1353        .angular_mode34_p33(angular_mode34_p33)
1354    );
1355
1356    always
1357        #H_CLOCK_PERIOD clock = !clock;
```

```
1358
1359    initial
1360    begin: Test_Case
1361
1362       ack = 0;
1363       clock = 0;
1364       reset = 0;
1365       start = 0;
1366
1367       def_readed              = 0;
1368       samples_readed          = 0;
1369       planar_step_checked        = 0;
1370       dc_step_checked            = 0;
1371       angular_mode2_step_checked = 0;
1372       angular_mode3_step_checked = 0;
1373       angular_mode4_step_checked = 0;
1374       angular_mode5_step_checked = 0;
1375       angular_mode6_step_checked = 0;
1376       angular_mode7_step_checked = 0;
1377       angular_mode8_step_checked = 0;
1378       angular_mode9_step_checked = 0;
1379       angular_mode10_step_checked   = 0;
1380       angular_mode11_step_checked   = 0;
1381       angular_mode12_step_checked   = 0;
1382       angular_mode13_step_checked   = 0;
1383       angular_mode14_step_checked   = 0;
1384       angular_mode15_step_checked   = 0;
1385       angular_mode16_step_checked   = 0;
1386       angular_mode17_step_checked   = 0;
1387       angular_mode18_step_checked   = 0;
1388       angular_mode19_step_checked   = 0;
1389       angular_mode20_step_checked   = 0;
1390       angular_mode21_step_checked   = 0;
1391       angular_mode22_step_checked   = 0;
1392       angular_mode23_step_checked   = 0;
1393       angular_mode24_step_checked   = 0;
1394       angular_mode25_step_checked   = 0;
1395       angular_mode26_step_checked   = 0;
1396       angular_mode27_step_checked   = 0;
1397       angular_mode28_step_checked   = 0;
1398       angular_mode29_step_checked   = 0;
1399       angular_mode30_step_checked   = 0;
1400       angular_mode31_step_checked   = 0;
1401       angular_mode32_step_checked   = 0;
1402       angular_mode33_step_checked   = 0;
1403       angular_mode34_step_checked   = 0;
1404       end_simulation          = 0;
1405
1406       sample_0n = 0;
1407       sample_1n = 0;
1408       sample_2n = 0;
1409       sample_3n = 0;
1410       sample_Nn = 0;
1411       sample_n0 = 0;
1412       sample_n1 = 0;
1413       sample_n2 = 0;
1414       sample_n3 = 0;
```

```
1415        sample_nN = 0;
1416        sample_NN = 0;
1417
1418        N = 0;
1419
1420        $intra_monitor(
1421           testbench.read_def,
1422           testbench.def_readed,
1423           testbench.read_samples_clock_trick,
1424           testbench.samples_readed,
1425           testbench.check_planar_step,
1426           testbench.planar_step_checked,
1427           testbench.check_dc_step,
1428           testbench.dc_step_checked,
1429           testbench.check_angular_mode2_step,
1430           testbench.angular_mode2_step_checked,
1431           testbench.check_angular_mode3_step,
1432           testbench.angular_mode3_step_checked,
1433           testbench.check_angular_mode4_step,
1434           testbench.angular_mode4_step_checked,
1435           testbench.check_angular_mode5_step,
1436           testbench.angular_mode5_step_checked,
1437           testbench.check_angular_mode6_step,
1438           testbench.angular_mode6_step_checked,
1439           testbench.check_angular_mode7_step,
1440           testbench.angular_mode7_step_checked,
1441           testbench.check_angular_mode8_step,
1442           testbench.angular_mode8_step_checked,
1443           testbench.check_angular_mode9_step,
1444           testbench.angular_mode9_step_checked,
1445           testbench.check_angular_mode10_step,
1446           testbench.angular_mode10_step_checked,
1447           testbench.check_angular_mode11_step,
1448           testbench.angular_mode11_step_checked,
1449           testbench.check_angular_mode12_step,
1450           testbench.angular_mode12_step_checked,
1451           testbench.check_angular_mode13_step,
1452           testbench.angular_mode13_step_checked,
1453           testbench.check_angular_mode14_step,
1454           testbench.angular_mode14_step_checked,
1455           testbench.check_angular_mode15_step,
1456           testbench.angular_mode15_step_checked,
1457           testbench.check_angular_mode16_step,
1458           testbench.angular_mode16_step_checked,
1459           testbench.check_angular_mode17_step,
1460           testbench.angular_mode17_step_checked,
1461           testbench.check_angular_mode18_step,
1462           testbench.angular_mode18_step_checked,
1463           testbench.check_angular_mode19_step,
1464           testbench.angular_mode19_step_checked,
1465           testbench.check_angular_mode20_step,
1466           testbench.angular_mode20_step_checked,
1467           testbench.check_angular_mode21_step,
1468           testbench.angular_mode21_step_checked,
1469           testbench.check_angular_mode22_step,
1470           testbench.angular_mode22_step_checked,
1471           testbench.check_angular_mode23_step,
```

```
1472            testbench.angular_mode23_step_checked,
1473            testbench.check_angular_mode24_step,
1474            testbench.angular_mode24_step_checked,
1475            testbench.check_angular_mode25_step,
1476            testbench.angular_mode25_step_checked,
1477            testbench.check_angular_mode26_step,
1478            testbench.angular_mode26_step_checked,
1479            testbench.check_angular_mode27_step,
1480            testbench.angular_mode27_step_checked,
1481            testbench.check_angular_mode28_step,
1482            testbench.angular_mode28_step_checked,
1483            testbench.check_angular_mode29_step,
1484            testbench.angular_mode29_step_checked,
1485            testbench.check_angular_mode30_step,
1486            testbench.angular_mode30_step_checked,
1487            testbench.check_angular_mode31_step,
1488            testbench.angular_mode31_step_checked,
1489            testbench.check_angular_mode32_step,
1490            testbench.angular_mode32_step_checked,
1491            testbench.check_angular_mode33_step,
1492            testbench.angular_mode33_step_checked,
1493            testbench.check_angular_mode34_step,
1494            testbench.angular_mode34_step_checked,
1495
1496            testbench.done,
1497
1498            testbench.sample_0n,
1499            testbench.sample_1n,
1500            testbench.sample_2n,
1501            testbench.sample_3n,
1502            testbench.sample_n0,
1503            testbench.sample_n1,
1504            testbench.sample_n2,
1505            testbench.sample_n3,
1506            testbench.sample_Nn,
1507            testbench.sample_nN,
1508            testbench.sample_NN,
1509            testbench.N,
1510
1511            testbench.planar_p00,
1512            testbench.planar_p01,
1513            testbench.planar_p02,
1514            testbench.planar_p03,
1515            testbench.planar_p10,
1516            testbench.planar_p11,
1517            testbench.planar_p12,
1518            testbench.planar_p13,
1519            testbench.planar_p20,
1520            testbench.planar_p21,
1521            testbench.planar_p22,
1522            testbench.planar_p23,
1523            testbench.planar_p30,
1524            testbench.planar_p31,
1525            testbench.planar_p32,
1526            testbench.planar_p33,
1527
1528            testbench.dc_p00,
```

```
1529            testbench.dc_p01,
1530            testbench.dc_p02,
1531            testbench.dc_p03,
1532            testbench.dc_p10,
1533            testbench.dc_p11,
1534            testbench.dc_p12,
1535            testbench.dc_p13,
1536            testbench.dc_p20,
1537            testbench.dc_p21,
1538            testbench.dc_p22,
1539            testbench.dc_p23,
1540            testbench.dc_p30,
1541            testbench.dc_p31,
1542            testbench.dc_p32,
1543            testbench.dc_p33,
1544
1545            testbench.angular_mode2_p00,
1546            testbench.angular_mode2_p01,
1547            testbench.angular_mode2_p02,
1548            testbench.angular_mode2_p03,
1549            testbench.angular_mode2_p10,
1550            testbench.angular_mode2_p11,
1551            testbench.angular_mode2_p12,
1552            testbench.angular_mode2_p13,
1553            testbench.angular_mode2_p20,
1554            testbench.angular_mode2_p21,
1555            testbench.angular_mode2_p22,
1556            testbench.angular_mode2_p23,
1557            testbench.angular_mode2_p30,
1558            testbench.angular_mode2_p31,
1559            testbench.angular_mode2_p32,
1560            testbench.angular_mode2_p33,
1561
1562            testbench.angular_mode3_p00,
1563            testbench.angular_mode3_p01,
1564            testbench.angular_mode3_p02,
1565            testbench.angular_mode3_p03,
1566            testbench.angular_mode3_p10,
1567            testbench.angular_mode3_p11,
1568            testbench.angular_mode3_p12,
1569            testbench.angular_mode3_p13,
1570            testbench.angular_mode3_p20,
1571            testbench.angular_mode3_p21,
1572            testbench.angular_mode3_p22,
1573            testbench.angular_mode3_p23,
1574            testbench.angular_mode3_p30,
1575            testbench.angular_mode3_p31,
1576            testbench.angular_mode3_p32,
1577            testbench.angular_mode3_p33,
1578
1579            testbench.angular_mode4_p00,
1580            testbench.angular_mode4_p01,
1581            testbench.angular_mode4_p02,
1582            testbench.angular_mode4_p03,
1583            testbench.angular_mode4_p10,
1584            testbench.angular_mode4_p11,
1585            testbench.angular_mode4_p12,
```

```
1586            testbench.angular_mode4_p13,
1587            testbench.angular_mode4_p20,
1588            testbench.angular_mode4_p21,
1589            testbench.angular_mode4_p22,
1590            testbench.angular_mode4_p23,
1591            testbench.angular_mode4_p30,
1592            testbench.angular_mode4_p31,
1593            testbench.angular_mode4_p32,
1594            testbench.angular_mode4_p33,
1595
1596            testbench.angular_mode5_p00,
1597            testbench.angular_mode5_p01,
1598            testbench.angular_mode5_p02,
1599            testbench.angular_mode5_p03,
1600            testbench.angular_mode5_p10,
1601            testbench.angular_mode5_p11,
1602            testbench.angular_mode5_p12,
1603            testbench.angular_mode5_p13,
1604            testbench.angular_mode5_p20,
1605            testbench.angular_mode5_p21,
1606            testbench.angular_mode5_p22,
1607            testbench.angular_mode5_p23,
1608            testbench.angular_mode5_p30,
1609            testbench.angular_mode5_p31,
1610            testbench.angular_mode5_p32,
1611            testbench.angular_mode5_p33,
1612
1613            testbench.angular_mode6_p00,
1614            testbench.angular_mode6_p01,
1615            testbench.angular_mode6_p02,
1616            testbench.angular_mode6_p03,
1617            testbench.angular_mode6_p10,
1618            testbench.angular_mode6_p11,
1619            testbench.angular_mode6_p12,
1620            testbench.angular_mode6_p13,
1621            testbench.angular_mode6_p20,
1622            testbench.angular_mode6_p21,
1623            testbench.angular_mode6_p22,
1624            testbench.angular_mode6_p23,
1625            testbench.angular_mode6_p30,
1626            testbench.angular_mode6_p31,
1627            testbench.angular_mode6_p32,
1628            testbench.angular_mode6_p33,
1629
1630            testbench.angular_mode7_p00,
1631            testbench.angular_mode7_p01,
1632            testbench.angular_mode7_p02,
1633            testbench.angular_mode7_p03,
1634            testbench.angular_mode7_p10,
1635            testbench.angular_mode7_p11,
1636            testbench.angular_mode7_p12,
1637            testbench.angular_mode7_p13,
1638            testbench.angular_mode7_p20,
1639            testbench.angular_mode7_p21,
1640            testbench.angular_mode7_p22,
1641            testbench.angular_mode7_p23,
1642            testbench.angular_mode7_p30,
```

```
1643            testbench.angular_mode7_p31,
1644            testbench.angular_mode7_p32,
1645            testbench.angular_mode7_p33,
1646
1647            testbench.angular_mode8_p00,
1648            testbench.angular_mode8_p01,
1649            testbench.angular_mode8_p02,
1650            testbench.angular_mode8_p03,
1651            testbench.angular_mode8_p10,
1652            testbench.angular_mode8_p11,
1653            testbench.angular_mode8_p12,
1654            testbench.angular_mode8_p13,
1655            testbench.angular_mode8_p20,
1656            testbench.angular_mode8_p21,
1657            testbench.angular_mode8_p22,
1658            testbench.angular_mode8_p23,
1659            testbench.angular_mode8_p30,
1660            testbench.angular_mode8_p31,
1661            testbench.angular_mode8_p32,
1662            testbench.angular_mode8_p33,
1663
1664            testbench.angular_mode9_p00,
1665            testbench.angular_mode9_p01,
1666            testbench.angular_mode9_p02,
1667            testbench.angular_mode9_p03,
1668            testbench.angular_mode9_p10,
1669            testbench.angular_mode9_p11,
1670            testbench.angular_mode9_p12,
1671            testbench.angular_mode9_p13,
1672            testbench.angular_mode9_p20,
1673            testbench.angular_mode9_p21,
1674            testbench.angular_mode9_p22,
1675            testbench.angular_mode9_p23,
1676            testbench.angular_mode9_p30,
1677            testbench.angular_mode9_p31,
1678            testbench.angular_mode9_p32,
1679            testbench.angular_mode9_p33,
1680
1681            testbench.angular_mode10_p00,
1682            testbench.angular_mode10_p01,
1683            testbench.angular_mode10_p02,
1684            testbench.angular_mode10_p03,
1685            testbench.angular_mode10_p10,
1686            testbench.angular_mode10_p11,
1687            testbench.angular_mode10_p12,
1688            testbench.angular_mode10_p13,
1689            testbench.angular_mode10_p20,
1690            testbench.angular_mode10_p21,
1691            testbench.angular_mode10_p22,
1692            testbench.angular_mode10_p23,
1693            testbench.angular_mode10_p30,
1694            testbench.angular_mode10_p31,
1695            testbench.angular_mode10_p32,
1696            testbench.angular_mode10_p33,
1697
1698            testbench.angular_mode11_p00,
1699            testbench.angular_mode11_p01,
```

```
        testbench.angular_mode11_p02,
        testbench.angular_mode11_p03,
        testbench.angular_mode11_p10,
        testbench.angular_mode11_p11,
        testbench.angular_mode11_p12,
        testbench.angular_mode11_p13,
        testbench.angular_mode11_p20,
        testbench.angular_mode11_p21,
        testbench.angular_mode11_p22,
        testbench.angular_mode11_p23,
        testbench.angular_mode11_p30,
        testbench.angular_mode11_p31,
        testbench.angular_mode11_p32,
        testbench.angular_mode11_p33,

        testbench.angular_mode12_p00,
        testbench.angular_mode12_p01,
        testbench.angular_mode12_p02,
        testbench.angular_mode12_p03,
        testbench.angular_mode12_p10,
        testbench.angular_mode12_p11,
        testbench.angular_mode12_p12,
        testbench.angular_mode12_p13,
        testbench.angular_mode12_p20,
        testbench.angular_mode12_p21,
        testbench.angular_mode12_p22,
        testbench.angular_mode12_p23,
        testbench.angular_mode12_p30,
        testbench.angular_mode12_p31,
        testbench.angular_mode12_p32,
        testbench.angular_mode12_p33,

        testbench.angular_mode13_p00,
        testbench.angular_mode13_p01,
        testbench.angular_mode13_p02,
        testbench.angular_mode13_p03,
        testbench.angular_mode13_p10,
        testbench.angular_mode13_p11,
        testbench.angular_mode13_p12,
        testbench.angular_mode13_p13,
        testbench.angular_mode13_p20,
        testbench.angular_mode13_p21,
        testbench.angular_mode13_p22,
        testbench.angular_mode13_p23,
        testbench.angular_mode13_p30,
        testbench.angular_mode13_p31,
        testbench.angular_mode13_p32,
        testbench.angular_mode13_p33,

        testbench.angular_mode14_p00,
        testbench.angular_mode14_p01,
        testbench.angular_mode14_p02,
        testbench.angular_mode14_p03,
        testbench.angular_mode14_p10,
        testbench.angular_mode14_p11,
        testbench.angular_mode14_p12,
        testbench.angular_mode14_p13,
```

```
1757            testbench.angular_mode14_p20,
1758            testbench.angular_mode14_p21,
1759            testbench.angular_mode14_p22,
1760            testbench.angular_mode14_p23,
1761            testbench.angular_mode14_p30,
1762            testbench.angular_mode14_p31,
1763            testbench.angular_mode14_p32,
1764            testbench.angular_mode14_p33,
1765
1766            testbench.angular_mode15_p00,
1767            testbench.angular_mode15_p01,
1768            testbench.angular_mode15_p02,
1769            testbench.angular_mode15_p03,
1770            testbench.angular_mode15_p10,
1771            testbench.angular_mode15_p11,
1772            testbench.angular_mode15_p12,
1773            testbench.angular_mode15_p13,
1774            testbench.angular_mode15_p20,
1775            testbench.angular_mode15_p21,
1776            testbench.angular_mode15_p22,
1777            testbench.angular_mode15_p23,
1778            testbench.angular_mode15_p30,
1779            testbench.angular_mode15_p31,
1780            testbench.angular_mode15_p32,
1781            testbench.angular_mode15_p33,
1782
1783            testbench.angular_mode16_p00,
1784            testbench.angular_mode16_p01,
1785            testbench.angular_mode16_p02,
1786            testbench.angular_mode16_p03,
1787            testbench.angular_mode16_p10,
1788            testbench.angular_mode16_p11,
1789            testbench.angular_mode16_p12,
1790            testbench.angular_mode16_p13,
1791            testbench.angular_mode16_p20,
1792            testbench.angular_mode16_p21,
1793            testbench.angular_mode16_p22,
1794            testbench.angular_mode16_p23,
1795            testbench.angular_mode16_p30,
1796            testbench.angular_mode16_p31,
1797            testbench.angular_mode16_p32,
1798            testbench.angular_mode16_p33,
1799
1800            testbench.angular_mode17_p00,
1801            testbench.angular_mode17_p01,
1802            testbench.angular_mode17_p02,
1803            testbench.angular_mode17_p03,
1804            testbench.angular_mode17_p10,
1805            testbench.angular_mode17_p11,
1806            testbench.angular_mode17_p12,
1807            testbench.angular_mode17_p13,
1808            testbench.angular_mode17_p20,
1809            testbench.angular_mode17_p21,
1810            testbench.angular_mode17_p22,
1811            testbench.angular_mode17_p23,
1812            testbench.angular_mode17_p30,
1813            testbench.angular_mode17_p31,
```

```
1814            testbench.angular_mode17_p32,
1815            testbench.angular_mode17_p33,
1816
1817            testbench.angular_mode18_p00,
1818            testbench.angular_mode18_p01,
1819            testbench.angular_mode18_p02,
1820            testbench.angular_mode18_p03,
1821            testbench.angular_mode18_p10,
1822            testbench.angular_mode18_p11,
1823            testbench.angular_mode18_p12,
1824            testbench.angular_mode18_p13,
1825            testbench.angular_mode18_p20,
1826            testbench.angular_mode18_p21,
1827            testbench.angular_mode18_p22,
1828            testbench.angular_mode18_p23,
1829            testbench.angular_mode18_p30,
1830            testbench.angular_mode18_p31,
1831            testbench.angular_mode18_p32,
1832            testbench.angular_mode18_p33,
1833
1834            testbench.angular_mode19_p00,
1835            testbench.angular_mode19_p01,
1836            testbench.angular_mode19_p02,
1837            testbench.angular_mode19_p03,
1838            testbench.angular_mode19_p10,
1839            testbench.angular_mode19_p11,
1840            testbench.angular_mode19_p12,
1841            testbench.angular_mode19_p13,
1842            testbench.angular_mode19_p20,
1843            testbench.angular_mode19_p21,
1844            testbench.angular_mode19_p22,
1845            testbench.angular_mode19_p23,
1846            testbench.angular_mode19_p30,
1847            testbench.angular_mode19_p31,
1848            testbench.angular_mode19_p32,
1849            testbench.angular_mode19_p33,
1850
1851            testbench.angular_mode20_p00,
1852            testbench.angular_mode20_p01,
1853            testbench.angular_mode20_p02,
1854            testbench.angular_mode20_p03,
1855            testbench.angular_mode20_p10,
1856            testbench.angular_mode20_p11,
1857            testbench.angular_mode20_p12,
1858            testbench.angular_mode20_p13,
1859            testbench.angular_mode20_p20,
1860            testbench.angular_mode20_p21,
1861            testbench.angular_mode20_p22,
1862            testbench.angular_mode20_p23,
1863            testbench.angular_mode20_p30,
1864            testbench.angular_mode20_p31,
1865            testbench.angular_mode20_p32,
1866            testbench.angular_mode20_p33,
1867
1868            testbench.angular_mode21_p00,
1869            testbench.angular_mode21_p01,
1870            testbench.angular_mode21_p02,
```

```
1871            testbench.angular_mode21_p03,
1872            testbench.angular_mode21_p10,
1873            testbench.angular_mode21_p11,
1874            testbench.angular_mode21_p12,
1875            testbench.angular_mode21_p13,
1876            testbench.angular_mode21_p20,
1877            testbench.angular_mode21_p21,
1878            testbench.angular_mode21_p22,
1879            testbench.angular_mode21_p23,
1880            testbench.angular_mode21_p30,
1881            testbench.angular_mode21_p31,
1882            testbench.angular_mode21_p32,
1883            testbench.angular_mode21_p33,
1884
1885            testbench.angular_mode22_p00,
1886            testbench.angular_mode22_p01,
1887            testbench.angular_mode22_p02,
1888            testbench.angular_mode22_p03,
1889            testbench.angular_mode22_p10,
1890            testbench.angular_mode22_p11,
1891            testbench.angular_mode22_p12,
1892            testbench.angular_mode22_p13,
1893            testbench.angular_mode22_p20,
1894            testbench.angular_mode22_p21,
1895            testbench.angular_mode22_p22,
1896            testbench.angular_mode22_p23,
1897            testbench.angular_mode22_p30,
1898            testbench.angular_mode22_p31,
1899            testbench.angular_mode22_p32,
1900            testbench.angular_mode22_p33,
1901
1902            testbench.angular_mode23_p00,
1903            testbench.angular_mode23_p01,
1904            testbench.angular_mode23_p02,
1905            testbench.angular_mode23_p03,
1906            testbench.angular_mode23_p10,
1907            testbench.angular_mode23_p11,
1908            testbench.angular_mode23_p12,
1909            testbench.angular_mode23_p13,
1910            testbench.angular_mode23_p20,
1911            testbench.angular_mode23_p21,
1912            testbench.angular_mode23_p22,
1913            testbench.angular_mode23_p23,
1914            testbench.angular_mode23_p30,
1915            testbench.angular_mode23_p31,
1916            testbench.angular_mode23_p32,
1917            testbench.angular_mode23_p33,
1918
1919            testbench.angular_mode24_p00,
1920            testbench.angular_mode24_p01,
1921            testbench.angular_mode24_p02,
1922            testbench.angular_mode24_p03,
1923            testbench.angular_mode24_p10,
1924            testbench.angular_mode24_p11,
1925            testbench.angular_mode24_p12,
1926            testbench.angular_mode24_p13,
1927            testbench.angular_mode24_p20,
```

```
1928            testbench.angular_mode24_p21,
1929            testbench.angular_mode24_p22,
1930            testbench.angular_mode24_p23,
1931            testbench.angular_mode24_p30,
1932            testbench.angular_mode24_p31,
1933            testbench.angular_mode24_p32,
1934            testbench.angular_mode24_p33,
1935
1936            testbench.angular_mode25_p00,
1937            testbench.angular_mode25_p01,
1938            testbench.angular_mode25_p02,
1939            testbench.angular_mode25_p03,
1940            testbench.angular_mode25_p10,
1941            testbench.angular_mode25_p11,
1942            testbench.angular_mode25_p12,
1943            testbench.angular_mode25_p13,
1944            testbench.angular_mode25_p20,
1945            testbench.angular_mode25_p21,
1946            testbench.angular_mode25_p22,
1947            testbench.angular_mode25_p23,
1948            testbench.angular_mode25_p30,
1949            testbench.angular_mode25_p31,
1950            testbench.angular_mode25_p32,
1951            testbench.angular_mode25_p33,
1952
1953            testbench.angular_mode26_p00,
1954            testbench.angular_mode26_p01,
1955            testbench.angular_mode26_p02,
1956            testbench.angular_mode26_p03,
1957            testbench.angular_mode26_p10,
1958            testbench.angular_mode26_p11,
1959            testbench.angular_mode26_p12,
1960            testbench.angular_mode26_p13,
1961            testbench.angular_mode26_p20,
1962            testbench.angular_mode26_p21,
1963            testbench.angular_mode26_p22,
1964            testbench.angular_mode26_p23,
1965            testbench.angular_mode26_p30,
1966            testbench.angular_mode26_p31,
1967            testbench.angular_mode26_p32,
1968            testbench.angular_mode26_p33,
1969
1970            testbench.angular_mode27_p00,
1971            testbench.angular_mode27_p01,
1972            testbench.angular_mode27_p02,
1973            testbench.angular_mode27_p03,
1974            testbench.angular_mode27_p10,
1975            testbench.angular_mode27_p11,
1976            testbench.angular_mode27_p12,
1977            testbench.angular_mode27_p13,
1978            testbench.angular_mode27_p20,
1979            testbench.angular_mode27_p21,
1980            testbench.angular_mode27_p22,
1981            testbench.angular_mode27_p23,
1982            testbench.angular_mode27_p30,
1983            testbench.angular_mode27_p31,
1984            testbench.angular_mode27_p32,
```

```
1985            testbench.angular_mode27_p33,
1986
1987            testbench.angular_mode28_p00,
1988            testbench.angular_mode28_p01,
1989            testbench.angular_mode28_p02,
1990            testbench.angular_mode28_p03,
1991            testbench.angular_mode28_p10,
1992            testbench.angular_mode28_p11,
1993            testbench.angular_mode28_p12,
1994            testbench.angular_mode28_p13,
1995            testbench.angular_mode28_p20,
1996            testbench.angular_mode28_p21,
1997            testbench.angular_mode28_p22,
1998            testbench.angular_mode28_p23,
1999            testbench.angular_mode28_p30,
2000            testbench.angular_mode28_p31,
2001            testbench.angular_mode28_p32,
2002            testbench.angular_mode28_p33,
2003
2004            testbench.angular_mode29_p00,
2005            testbench.angular_mode29_p01,
2006            testbench.angular_mode29_p02,
2007            testbench.angular_mode29_p03,
2008            testbench.angular_mode29_p10,
2009            testbench.angular_mode29_p11,
2010            testbench.angular_mode29_p12,
2011            testbench.angular_mode29_p13,
2012            testbench.angular_mode29_p20,
2013            testbench.angular_mode29_p21,
2014            testbench.angular_mode29_p22,
2015            testbench.angular_mode29_p23,
2016            testbench.angular_mode29_p30,
2017            testbench.angular_mode29_p31,
2018            testbench.angular_mode29_p32,
2019            testbench.angular_mode29_p33,
2020
2021            testbench.angular_mode30_p00,
2022            testbench.angular_mode30_p01,
2023            testbench.angular_mode30_p02,
2024            testbench.angular_mode30_p03,
2025            testbench.angular_mode30_p10,
2026            testbench.angular_mode30_p11,
2027            testbench.angular_mode30_p12,
2028            testbench.angular_mode30_p13,
2029            testbench.angular_mode30_p20,
2030            testbench.angular_mode30_p21,
2031            testbench.angular_mode30_p22,
2032            testbench.angular_mode30_p23,
2033            testbench.angular_mode30_p30,
2034            testbench.angular_mode30_p31,
2035            testbench.angular_mode30_p32,
2036            testbench.angular_mode30_p33,
2037
2038            testbench.angular_mode31_p00,
2039            testbench.angular_mode31_p01,
2040            testbench.angular_mode31_p02,
2041            testbench.angular_mode31_p03,
```

```
        testbench.angular_mode31_p10,
        testbench.angular_mode31_p11,
        testbench.angular_mode31_p12,
        testbench.angular_mode31_p13,
        testbench.angular_mode31_p20,
        testbench.angular_mode31_p21,
        testbench.angular_mode31_p22,
        testbench.angular_mode31_p23,
        testbench.angular_mode31_p30,
        testbench.angular_mode31_p31,
        testbench.angular_mode31_p32,
        testbench.angular_mode31_p33,

        testbench.angular_mode32_p00,
        testbench.angular_mode32_p01,
        testbench.angular_mode32_p02,
        testbench.angular_mode32_p03,
        testbench.angular_mode32_p10,
        testbench.angular_mode32_p11,
        testbench.angular_mode32_p12,
        testbench.angular_mode32_p13,
        testbench.angular_mode32_p20,
        testbench.angular_mode32_p21,
        testbench.angular_mode32_p22,
        testbench.angular_mode32_p23,
        testbench.angular_mode32_p30,
        testbench.angular_mode32_p31,
        testbench.angular_mode32_p32,
        testbench.angular_mode32_p33,

        testbench.angular_mode33_p00,
        testbench.angular_mode33_p01,
        testbench.angular_mode33_p02,
        testbench.angular_mode33_p03,
        testbench.angular_mode33_p10,
        testbench.angular_mode33_p11,
        testbench.angular_mode33_p12,
        testbench.angular_mode33_p13,
        testbench.angular_mode33_p20,
        testbench.angular_mode33_p21,
        testbench.angular_mode33_p22,
        testbench.angular_mode33_p23,
        testbench.angular_mode33_p30,
        testbench.angular_mode33_p31,
        testbench.angular_mode33_p32,
        testbench.angular_mode33_p33,

        testbench.angular_mode34_p00,
        testbench.angular_mode34_p01,
        testbench.angular_mode34_p02,
        testbench.angular_mode34_p03,
        testbench.angular_mode34_p10,
        testbench.angular_mode34_p11,
        testbench.angular_mode34_p12,
        testbench.angular_mode34_p13,
        testbench.angular_mode34_p20,
        testbench.angular_mode34_p21,
```

```
2099            testbench.angular_mode34_p22,
2100            testbench.angular_mode34_p23,
2101            testbench.angular_mode34_p30,
2102            testbench.angular_mode34_p31,
2103            testbench.angular_mode34_p32,
2104            testbench.angular_mode34_p33,
2105
2106            testbench.end_simulation
2107        );
2108
2109        $dumpfile("../vcds/top_level.vcd");
2110        $dumpvars;
2111
2112        $set_gate_level_monitoring("on");
2113          $set_toggle_region(intra_predictor);
2114          $toggle_start;
2115
2116        // STATE: XXXX
2117        #CLOCK_PERIOD;
2118
2119            @(negedge clock);
2120               reset = 1;
2121            @(negedge clock);
2122               reset = 0;
2123
2124        // STATE: INIT
2125        #CLOCK_PERIOD;
2126
2127            start = 1;
2128
2129            while (1)
2130            begin
2131               // STATE: ALL
2132               #CLOCK_PERIOD;
2133            end
2134    end
2135
2136    always @(posedge end_simulation)
2137    begin
2138
2139        $toggle_stop;
2140        $toggle_report("../saifs/output.saif", 0.0000000001, intra_predictor);
2141
2142        $finish;
2143
2144    end
2145
2146    always @(posedge read_def)
2147    begin
2148
2149        // while (!def_readed);
2150        $display("## def_readed ##");
2151
2152        def_readed = 0;
2153        start = 0;
2154        ack = 0;
2155
```

```
2156     end
2157
2158     always @(posedge done)
2159     begin
2160
2161        start = 1;
2162        ack = 1;
2163
2164     end
2165
2166     // always @(posedge check_planar_step)
2167     // begin
2168
2169     //    $toggle_stop;
2170     //    $toggle_report("../saifs/output.saif", 0.0000000001, intra_predictor);
2171
2172     //    $finish;
2173
2174     // end
2175
2176     // always @(posedge read_samples)
2177     // begin
2178
2179     //    // while (!samples_readed);
2180     //    $display("## samples_readed ##");
2181
2182     //    samples_readed = 0;
2183
2184     // end
2185
2186     // always @(posedge check_planar_step)
2187     // begin
2188
2189     //    while (!planar_step_checked);
2190
2191     //    planar_step_checked = 0;
2192
2193     // end
2194
2195     // always @(posedge check_dc_step)
2196     // begin
2197
2198     //    while (!dc_step_checked);
2199
2200     //    dc_step_checked = 0;
2201
2202     // end
2203
2204     // always @(posedge check_angular_mode2_step)
2205     // begin
2206
2207     //    while (!angular_mode2_step_checked);
2208
2209     //    angular_mode2_step_checked = 0;
2210
2211     // end
2212
```

```
2213    // always @(posedge check_angular_mode3_step)
2214    // begin
2215
2216    //    while (!angular_mode3_step_checked);
2217
2218    //    angular_mode3_step_checked = 0;
2219
2220    // end
2221
2222    // always @(posedge check_angular_mode4_step)
2223    // begin
2224
2225    //    while (!angular_mode4_step_checked);
2226
2227    //    angular_mode4_step_checked = 0;
2228
2229    // end
2230
2231    // always @(posedge check_angular_mode5_step)
2232    // begin
2233
2234    //    while (!angular_mode5_step_checked);
2235
2236    //    angular_mode5_step_checked = 0;
2237
2238    // end
2239
2240    // always @(posedge check_angular_mode6_step)
2241    // begin
2242
2243    //    while (!angular_mode6_step_checked);
2244
2245    //    angular_mode6_step_checked = 0;
2246
2247    // end
2248
2249    // always @(posedge check_angular_mode7_step)
2250    // begin
2251
2252    //    while (!angular_mode7_step_checked);
2253
2254    //    angular_mode7_step_checked = 0;
2255
2256    // end
2257
2258    // always @(posedge check_angular_mode8_step)
2259    // begin
2260
2261    //    while (!angular_mode8_step_checked);
2262
2263    //    angular_mode8_step_checked = 0;
2264
2265    // end
2266
2267    // always @(posedge check_angular_mode9_step)
2268    // begin
2269
```

```
2270    //    while (!angular_mode9_step_checked);
2271
2272    //    angular_mode9_step_checked = 0;
2273
2274    // end
2275
2276    // always @(posedge check_angular_mode10_step)
2277    // begin
2278
2279    //    while (!angular_mode10_step_checked);
2280
2281    //    angular_mode10_step_checked = 0;
2282
2283    // end
2284
2285    // always @(posedge check_angular_mode11_step)
2286    // begin
2287
2288    //    while (!angular_mode11_step_checked);
2289
2290    //    angular_mode11_step_checked = 0;
2291
2292    // end
2293
2294    // always @(posedge check_angular_mode12_step)
2295    // begin
2296
2297    //    while (!angular_mode12_step_checked);
2298
2299    //    angular_mode12_step_checked = 0;
2300
2301    // end
2302
2303    // always @(posedge check_angular_mode13_step)
2304    // begin
2305
2306    //    while (!angular_mode13_step_checked);
2307
2308    //    angular_mode13_step_checked = 0;
2309
2310    // end
2311
2312    // always @(posedge check_angular_mode14_step)
2313    // begin
2314
2315    //    while (!angular_mode14_step_checked);
2316
2317    //    angular_mode14_step_checked = 0;
2318
2319    // end
2320
2321    // always @(posedge check_angular_mode15_step)
2322    // begin
2323
2324    //    while (!angular_mode15_step_checked);
2325
2326    //    angular_mode15_step_checked = 0;
```

```
2327
2328    // end
2329
2330    // always @(posedge check_angular_mode16_step)
2331    // begin
2332
2333    //    while (!angular_mode16_step_checked);
2334
2335    //    angular_mode16_step_checked = 0;
2336
2337    // end
2338
2339    // always @(posedge check_angular_mode17_step)
2340    // begin
2341
2342    //    while (!angular_mode17_step_checked);
2343
2344    //    angular_mode17_step_checked = 0;
2345
2346    // end
2347
2348    // always @(posedge check_angular_mode18_step)
2349    // begin
2350
2351    //    while (!angular_mode18_step_checked);
2352
2353    //    angular_mode18_step_checked = 0;
2354
2355    // end
2356
2357    // always @(posedge check_angular_mode19_step)
2358    // begin
2359
2360    //    while (!angular_mode19_step_checked);
2361
2362    //    angular_mode19_step_checked = 0;
2363
2364    // end
2365
2366    // always @(posedge check_angular_mode20_step)
2367    // begin
2368
2369    //    while (!angular_mode20_step_checked);
2370
2371    //    angular_mode20_step_checked = 0;
2372
2373    // end
2374
2375    // always @(posedge check_angular_mode21_step)
2376    // begin
2377
2378    //    while (!angular_mode21_step_checked);
2379
2380    //    angular_mode21_step_checked = 0;
2381
2382    // end
2383
```

```
2384    // always @(posedge check_angular_mode22_step)
2385    // begin
2386
2387    //    while (!angular_mode22_step_checked);
2388
2389    //    angular_mode22_step_checked = 0;
2390
2391    // end
2392
2393    // always @(posedge check_angular_mode23_step)
2394    // begin
2395
2396    //    while (!angular_mode23_step_checked);
2397
2398    //    angular_mode23_step_checked = 0;
2399
2400    // end
2401
2402    // always @(posedge check_angular_mode24_step)
2403    // begin
2404
2405    //    while (!angular_mode24_step_checked);
2406
2407    //    angular_mode24_step_checked = 0;
2408
2409    // end
2410
2411    // always @(posedge check_angular_mode25_step)
2412    // begin
2413
2414    //    while (!angular_mode25_step_checked);
2415
2416    //    angular_mode25_step_checked = 0;
2417
2418    // end
2419
2420    // always @(posedge check_angular_mode26_step)
2421    // begin
2422
2423    //    while (!angular_mode26_step_checked);
2424
2425    //    angular_mode26_step_checked = 0;
2426
2427    // end
2428
2429    // always @(posedge check_angular_mode27_step)
2430    // begin
2431
2432    //    while (!angular_mode27_step_checked);
2433
2434    //    angular_mode27_step_checked = 0;
2435
2436    // end
2437
2438    // always @(posedge check_angular_mode28_step)
2439    // begin
2440
```

```
2441     //    while (!angular_mode28_step_checked);
2442
2443     //    angular_mode28_step_checked = 0;
2444
2445     // end
2446
2447     // always @(posedge check_angular_mode29_step)
2448     // begin
2449
2450     //    while (!angular_mode29_step_checked);
2451
2452     //    angular_mode29_step_checked = 0;
2453
2454     // end
2455
2456     // always @(posedge check_angular_mode30_step)
2457     // begin
2458
2459     //    while (!angular_mode30_step_checked);
2460
2461     //    angular_mode30_step_checked = 0;
2462
2463     // end
2464
2465     // always @(posedge check_angular_mode31_step)
2466     // begin
2467
2468     //    while (!angular_mode31_step_checked);
2469
2470     //    angular_mode31_step_checked = 0;
2471
2472     // end
2473
2474     // always @(posedge check_angular_mode32_step)
2475     // begin
2476
2477     //    while (!angular_mode32_step_checked);
2478
2479     //    angular_mode32_step_checked = 0;
2480
2481     // end
2482
2483     // always @(posedge check_angular_mode33_step)
2484     // begin
2485
2486     //    while (!angular_mode33_step_checked);
2487
2488     //    angular_mode33_step_checked = 0;
2489
2490     // end
2491
2492     // always @(posedge check_angular_mode34_step)
2493     // begin
2494
2495     //    while (!angular_mode34_step_checked);
2496
2497     //    angular_mode34_step_checked = 0;
```

```
2498
2499     // end
2500
2501     assign read_samples_clock_trick = read_samples && !clock;
2502
2503 endmodule
```

B.4 MÓDULOS HDL DA ARQUITETURA PROPOSTA

Listing B.5 – Código fonte do bloco de controle do preditor diretamente angular

```
1
2  module angular_directly_control
3
4      (
5         input     clock,
6               reset,
7               start,
8               m_condition,
9               i_condition,
10
11        output    reset_registers,
12              m_reset,
13              def_enable,
14              refs_enable,
15              shift_enable,
16              i_index_enable,
17              m_index_enable,
18              indexes_enable,
19              part_done,
20              done
21     );
22
23     reg [3:0] state;
24
25     reg reg_reset_registers,
26        reg_m_reset,
27        reg_def_enable,
28        reg_refs_enable,
29        reg_shift_enable,
30        reg_i_index_enable,
31        reg_m_index_enable,
32        reg_indexes_enable,
33        reg_part_done,
34        reg_done;
35
36     parameter    INIT = 0,
37              PRE_LOAD = 1,
38              TIME1 = 2,
39              LOAD = 3,
40              TIME2 = 4,
41              CALC = 5,
42              ITEST = 6,
43              UPDATE = 7,
44              DONE = 8;
45
```

```
46      always @(state)
47      begin
48         case (state)
49
50            INIT:
51               begin
52                  reg_reset_registers     = 1'b1;
53                  reg_m_reset          = 1'b0;
54                  reg_def_enable       = 1'b0;
55                  reg_refs_enable         = 1'b0;
56                  reg_shift_enable     = 1'b0;
57                  reg_i_index_enable      = 1'b0;
58                  reg_m_index_enable      = 1'b0;
59                  reg_indexes_enable      = 1'b0;
60
61                  reg_part_done        = 1'b0;
62                  reg_done          = 1'b0;
63               end
64
65            PRE_LOAD:
66               begin
67                  reg_reset_registers     = 1'b0;
68                  reg_m_reset          = 1'b0;
69                  reg_def_enable       = 1'b1;
70                  reg_refs_enable         = 1'b0;
71                  reg_shift_enable     = 1'b0;
72                  reg_i_index_enable      = 1'b0;
73                  reg_m_index_enable      = 1'b0;
74                  reg_indexes_enable      = 1'b0;
75
76                  reg_part_done        = 1'b0;
77                  reg_done          = 1'b0;
78               end
79
80            TIME1:
81               begin
82                  reg_reset_registers     = 1'b0;
83                  reg_m_reset          = 1'b0;
84                  reg_def_enable       = 1'b0;
85                  reg_refs_enable         = 1'b0;
86                  reg_shift_enable     = 1'b0;
87                  reg_i_index_enable      = 1'b0;
88                  reg_m_index_enable      = 1'b1;
89                  reg_indexes_enable      = 1'b0;
90
91                  reg_part_done        = 1'b0;
92                  reg_done          = 1'b0;
93               end
94
95            LOAD:
96               begin
97                  reg_reset_registers     = 1'b0;
98                  reg_m_reset          = 1'b0;
99                  reg_def_enable       = 1'b0;
100                 reg_refs_enable         = 1'b1;
101                 reg_shift_enable     = 1'b0;
102                 reg_i_index_enable      = 1'b0;
```

```
103                reg_m_index_enable      = 1'b0;
104                reg_indexes_enable      = 1'b1;
105
106                reg_part_done         = 1'b0;
107                reg_done           = 1'b0;
108             end
109
110          TIME2:
111             begin
112                reg_reset_registers     = 1'b0;
113                reg_m_reset           = 1'b0;
114                reg_def_enable        = 1'b0;
115                reg_refs_enable          = 1'b0;
116                reg_shift_enable      = 1'b0;
117                reg_i_index_enable      = 1'b0;
118                reg_m_index_enable      = 1'b0;
119                reg_indexes_enable      = 1'b0;
120
121                reg_part_done         = 1'b0;
122                reg_done           = 1'b0;
123             end
124
125          CALC:
126             begin
127                reg_reset_registers     = 1'b0;
128                reg_m_reset           = 1'b0;
129                reg_def_enable        = 1'b0;
130                reg_refs_enable          = 1'b0;
131                reg_shift_enable      = 1'b1;
132                reg_i_index_enable      = 1'b0;
133                reg_m_index_enable      = 1'b0;
134                reg_indexes_enable      = 1'b0;
135
136                reg_part_done         = 1'b0;
137                reg_done           = 1'b0;
138             end
139
140          ITEST:
141             begin
142                reg_reset_registers     = 1'b0;
143                reg_m_reset           = 1'b1;
144                reg_def_enable        = 1'b0;
145                reg_refs_enable          = 1'b0;
146                reg_shift_enable      = 1'b0;
147                reg_i_index_enable      = 1'b1;
148                reg_m_index_enable      = 1'b0;
149                reg_indexes_enable      = 1'b0;
150
151                reg_part_done         = 1'b0;
152                reg_done           = 1'b0;
153             end
154
155          UPDATE:
156             begin
157                reg_reset_registers     = 1'b0;
158                reg_m_reset           = 1'b0;
159                reg_def_enable        = 1'b0;
```

```
160                reg_refs_enable          = 1'b0;
161                reg_shift_enable     = 1'b0;
162                reg_i_index_enable       = 1'b0;
163                reg_m_index_enable       = 1'b0;
164                reg_indexes_enable       = 1'b1;
165
166                reg_part_done        = 1'b1;
167                reg_done          = 1'b0;
168             end
169
170          DONE:
171             begin
172                reg_reset_registers      = 1'b0;
173                reg_m_reset          = 1'b0;
174                reg_def_enable       = 1'b0;
175                reg_refs_enable          = 1'b0;
176                reg_shift_enable     = 1'b0;
177                reg_i_index_enable       = 1'b0;
178                reg_m_index_enable       = 1'b0;
179                reg_indexes_enable       = 1'b0;
180
181                reg_part_done        = 1'b1;
182                reg_done          = 1'b1;
183             end
184
185          default:
186             begin
187                reg_reset_registers      = 1'b0;
188                reg_m_reset          = 1'b0;
189                reg_def_enable       = 1'b0;
190                reg_refs_enable          = 1'b0;
191                reg_shift_enable     = 1'b0;
192                reg_i_index_enable       = 1'b0;
193                reg_m_index_enable       = 1'b0;
194                reg_indexes_enable       = 1'b0;
195
196                reg_part_done        = 1'b0;
197                reg_done          = 1'b0;
198             end
199       endcase
200    end
201
202    always @(posedge clock)
203    begin
204       if (reset)
205          state <= INIT;
206       else
207       begin
208          case (state)
209
210             INIT:
211                begin
212                   if (start)
213                      state <= PRE_LOAD;
214                   else
215                      state <= INIT;
216                end
```

```
217
218             PRE_LOAD:
219                begin
220                   state <= TIME1;
221                end
222
223             TIME1:
224                begin
225                   state <= LOAD;
226                end
227
228             LOAD:
229                begin
230                   state <= TIME2;
231                end
232
233             TIME2:
234                begin
235                   state <= CALC;
236                end
237
238             CALC:
239                begin
240                   if (m_condition)
241                   begin
242                      state <= TIME1;
243                   end
244                   else
245                   begin
246                      state <= ITEST;
247                   end
248                end
249
250             ITEST:
251                begin
252                   if (i_condition)
253                      state <= UPDATE;
254                   else
255                      state <= DONE;
256                end
257
258             UPDATE:
259                begin
260                   state <= PRE_LOAD;
261                end
262
263             DONE:
264                begin
265                   state <= DONE;
266                end
267
268             default:
269                begin
270                   state <= INIT;
271                end
272          endcase
273       end
```

```
274     end
275
276     assign reset_registers     = reg_reset_registers;
277     assign m_reset          = reg_m_reset;
278     assign def_enable       = reg_def_enable;
279     assign refs_enable         = reg_refs_enable;
280     assign shift_enable        = reg_shift_enable;
281     assign i_index_enable      = reg_i_index_enable;
282     assign m_index_enable      = reg_m_index_enable;
283     assign indexes_enable      = reg_indexes_enable;
284     assign part_done        = reg_part_done;
285     assign done             = reg_done;
286
287 endmodule
```

Listing B.6 – Código fonte do bloco operativo do preditor diretamente angular

```
1
2  module angular_directly_operative
3
4     #(parameter WIDTH = 8, parameter MODE = 10)
5
6     (
7        input          clock,
8                       reset,
9                       m_reset,
10                      def_enable,
11                      refs_enable,
12                      shift_enable,
13                      i_index_enable,
14                      m_index_enable,
15                      indexes_enable,
16
17       input[WIDTH-1:0]  ref0,
18                      ref1,
19                      ref2,
20                      ref3,
21                      ref4,
22                      ref5,
23
24       input[WIDTH-3:0]  N,
25
26       output             m_condition,
27                      i_condition,
28
29       output[WIDTH-2:0] filter,
30
31       output signed [WIDTH-1:0]  samples,
32
33       output[WIDTH-1:0] p00,
34                      p01,
35                      p02,
36                      p03,
37                      p10,
38                      p11,
39                      p12,
40                      p13,
```

```
41                      p20,
42                      p21,
43                      p22,
44                      p23,
45                      p30,
46                      p31,
47                      p32,
48                      p33
49      );
50
51      reg[WIDTH-6:0] reg_m;
52
53      reg[WIDTH-5:0] reg_filter;
54
55      reg[WIDTH-3:0] reg_x_mask,
56                     reg_y_mask,
57                     reg_N;
58
59      reg[WIDTH-2:0] reg_i,
60                     reg_k,
61                     reg_l,
62                     reg_iterations;
63
64      reg[WIDTH-1:0] reg_ref0,
65                     reg_ref1,
66                     reg_ref2,
67                     reg_ref3,
68                     reg_ref4,
69                     reg_ref5;
70
71      wire[WIDTH-6:0] m;
72
73      wire[WIDTH-2:0] i,
74                     k,
75                     l;
76
77      wire[WIDTH-1:0]   pn0,
78                     pn1,
79                     pn2,
80                     pn3;
81
82      angular_shift_buffer #(WIDTH) angular_shift_buffer_block(
83         .clock(clock),
84         .reset(reset),
85         .shift_enable(shift_enable),
86         .pn0(pn0),
87         .pn1(pn1),
88         .pn2(pn2),
89         .pn3(pn3),
90
91         .p00(p00),
92         .p01(p01),
93         .p02(p02),
94         .p03(p03),
95         .p10(p10),
96         .p11(p11),
97         .p12(p12),
```

```
98          .p13(p13),
99          .p20(p20),
100         .p21(p21),
101         .p22(p22),
102         .p23(p23),
103         .p30(p30),
104         .p31(p31),
105         .p32(p32),
106         .p33(p33)
107     );
108
109     always @(posedge clock)
110     begin
111
112        if (reset)
113        begin
114           reg_N          <= 6'b000000;
115           reg_i          <= 7'b0000000;
116           reg_k          <= 7'b0000000;
117           reg_l          <= 7'b0000000;
118           reg_m          <= 3'b000;
119           reg_x_mask        <= 6'b000000;
120           reg_y_mask        <= 6'b000000;
121           reg_iterations    <= 7'b0000000;
122           reg_ref0       <= 8'b00000000;
123           reg_ref1       <= 8'b00000000;
124           reg_ref2       <= 8'b00000000;
125           reg_ref3       <= 8'b00000000;
126           reg_ref4       <= 8'b00000000;
127           reg_ref5       <= 8'b00000000;
128           reg_filter        <= 4'b0000;
129        end
130        else
131        begin
132
133           if (def_enable)
134           begin
135              reg_N    <= N;
136              reg_filter  <= N >> 2;
137
138              case (N)
139                 4: begin
140                       reg_iterations <= 7'b0000001;
141                       reg_x_mask     <= 6'b000000;
142                       reg_y_mask     <= 6'b000000;
143                    end
144
145                 8: begin
146                       reg_iterations <= 7'b0000100;
147                       reg_x_mask     <= 6'b000111;
148                       reg_y_mask     <= 6'b000001;
149                    end
150
151                 16:   begin
152                       reg_iterations <= 7'b0010000;
153                       reg_x_mask     <= 6'b001111;
154                       reg_y_mask     <= 6'b000010;
```

```
155                  end
156
157               32:   begin
158                        reg_iterations <= 7'b1000000;
159                        reg_x_mask     <= 6'b011111;
160                        reg_y_mask     <= 6'b000011;
161                  end
162
163               default: begin
164                        reg_iterations <= 7'b0000001;
165                        reg_x_mask     <= 6'b000000;
166                        reg_y_mask     <= 6'b000000;
167                  end
168            endcase
169         end
170
171         if (refs_enable)
172         begin
173            reg_ref0 <= ref0;
174            reg_ref1 <= ref1;
175            reg_ref2 <= ref2;
176            reg_ref3 <= ref3;
177            reg_ref4 <= ref4;
178            reg_ref5 <= ref5;
179         end
180
181         if (i_index_enable)
182         begin
183            reg_i <= i;
184         end
185
186         if (m_reset)
187         begin
188            reg_m <= 3'b000;
189         end
190         else
191         begin
192            if (m_index_enable)
193            begin
194               reg_m <= m;
195            end
196         end
197
198         if (indexes_enable)
199         begin
200            reg_k <= k;
201            reg_l <= l;
202         end
203      end
204   end
205
206   assign m          = reg_m + 1;
207   assign i          = reg_i + 1;
208
209   assign k          = reg_m + ((reg_i << 2) & reg_x_mask);
210   assign l          = (reg_i >> reg_y_mask) << 2;
211
```

```
212     assign pn0           = reg_i < reg_filter ? ((32 * reg_ref1 + 16) >> 5) + ((
            reg_ref5 - reg_ref0) >> 1) : (32 * reg_ref1 + 16) >> 5;
213     assign pn1           = (32 * reg_ref2 + 16) >> 5;
214     assign pn2           = (32 * reg_ref3 + 16) >> 5;
215     assign pn3           = (32 * reg_ref4 + 16) >> 5;
216
217     assign i_condition      = reg_i < reg_iterations - 1;
218     assign m_condition      = reg_m < 4;
219
220     assign samples       = reg_l;
221     assign filter        = reg_k;
222
223 endmodule
```

Listing B.7 – Código fonte do seletor de amostras do preditor diretamente angular

```
1
2  module angular_directly_refs_selector
3
4     #(parameter WIDTH = 8, parameter MODE = 10)
5
6     (
7        input clock,
8
9        input[WIDTH-2:0]        index,
10
11       input signed [WIDTH-1:0]   selector,
12
13       input[WIDTH-1:0]  sample_0n,  sample_32n, sample_n0,  sample_n32,
14                      sample_1n,  sample_33n, sample_n1,  sample_n33,
15                      sample_2n,  sample_34n, sample_n2,  sample_n34,
16                      sample_3n,  sample_35n, sample_n3,  sample_n35,
17                      sample_4n,  sample_36n, sample_n4,  sample_n36,
18                      sample_5n,  sample_37n, sample_n5,  sample_n37,
19                      sample_6n,  sample_38n, sample_n6,  sample_n38,
20                      sample_7n,  sample_39n, sample_n7,  sample_n39,
21                      sample_8n,  sample_40n, sample_n8,  sample_n40,
22                      sample_9n,  sample_41n, sample_n9,  sample_n41,
23                      sample_10n, sample_42n, sample_n10, sample_n42,
24                      sample_11n, sample_43n, sample_n11, sample_n43,
25                      sample_12n, sample_44n, sample_n12, sample_n44,
26                      sample_13n, sample_45n, sample_n13, sample_n45,
27                      sample_14n, sample_46n, sample_n14, sample_n46,
28                      sample_15n, sample_47n, sample_n15, sample_n47,
29                      sample_16n, sample_48n, sample_n16, sample_n48,
30                      sample_17n, sample_49n, sample_n17, sample_n49,
31                      sample_18n, sample_50n, sample_n18, sample_n50,
32                      sample_19n, sample_51n, sample_n19, sample_n51,
33                      sample_20n, sample_52n, sample_n20, sample_n52,
34                      sample_21n, sample_53n, sample_n21, sample_n53,
35                      sample_22n, sample_54n, sample_n22, sample_n54,
36                      sample_23n, sample_55n, sample_n23, sample_n55,
37                      sample_24n, sample_56n, sample_n24, sample_n56,
38                      sample_25n, sample_57n, sample_n25, sample_n57,
39                      sample_26n, sample_58n, sample_n26, sample_n58,
40                      sample_27n, sample_59n, sample_n27, sample_n59,
41                      sample_28n, sample_60n, sample_n28, sample_n60,
```

```
42                      sample_29n, sample_61n, sample_n29, sample_n61,
43                      sample_30n, sample_62n, sample_n30, sample_n62,
44                      sample_31n, sample_63n, sample_n31, sample_n63,
45
46                      sample_NN,
47
48      output[WIDTH-1:0] ref0,
49                      ref1,
50                      ref2,
51                      ref3,
52                      ref4,
53                      ref5
54  );
55
56  assign ref0 = sample_NN;
57
58  assign ref1 = MODE < 18 ? ( (selector == -32 && (MODE == 15 || MODE == 21)) ?
        sample_57n
59                      : ( (selector == -32 && (MODE == 16 || MODE == 20)) ?
                            sample_46n
60                      : ( (selector == -32 && (MODE == 17 || MODE == 19)) ?
                            sample_37n
61                      : ( (selector == -32 &&  MODE == 18)            ?
                            sample_30n
62                      : ( (selector == -31 && (MODE == 15 || MODE == 21)) ?
                            sample_55n
63                      : ( (selector == -31 && (MODE == 16 || MODE == 20)) ?
                            sample_45n
64                      : ( (selector == -31 && (MODE == 17 || MODE == 19)) ?
                            sample_36n
65                      : ( (selector == -31 &&  MODE == 18)            ?
                            sample_29n
66                      : ( (selector == -30 && (MODE == 15 || MODE == 21)) ?
                            sample_54n
67                      : ( (selector == -30 && (MODE == 16 || MODE == 20)) ?
                            sample_43n
68                      : ( (selector == -30 && (MODE == 17 || MODE == 19)) ?
                            sample_35n
69                      : ( (selector == -30 &&  MODE == 18)            ?
                            sample_28n
70                      : ( (selector == -29 && (MODE == 15 || MODE == 21)) ?
                            sample_52n
71                      : ( (selector == -29 && (MODE == 16 || MODE == 20)) ?
                            sample_42n
72                      : ( (selector == -29 && (MODE == 17 || MODE == 19)) ?
                            sample_33n
73                      : ( (selector == -29 &&  MODE == 18)            ?
                            sample_27n
74                      : ( (selector == -28 && (MODE == 15 || MODE == 21)) ?
                            sample_50n
75                      : ( (selector == -28 && (MODE == 16 || MODE == 20)) ?
                            sample_40n
76                      : ( (selector == -28 && (MODE == 17 || MODE == 19)) ?
                            sample_32n
77                      : ( (selector == -28 &&  MODE == 18)            ?
                            sample_26n
78                      : ( (selector == -27 && (MODE == 14 || MODE == 22)) ?
```

```
                    sample_63n
79                : ( (selector == -27 && (MODE == 15 || MODE == 21)) ?
                    sample_48n
80                : ( (selector == -27 && (MODE == 16 || MODE == 20)) ?
                    sample_39n
81                : ( (selector == -27 && (MODE == 17 || MODE == 19)) ?
                    sample_31n
82                : ( (selector == -27 &&  MODE == 18)            ?
                    sample_25n
83                : ( (selector == -26 && (MODE == 14 || MODE == 22)) ?
                    sample_61n
84                : ( (selector == -26 && (MODE == 15 || MODE == 21)) ?
                    sample_46n
85                : ( (selector == -26 && (MODE == 16 || MODE == 20)) ?
                    sample_37n
86                : ( (selector == -26 && (MODE == 17 || MODE == 19)) ?
                    sample_30n
87                : ( (selector == -26 &&  MODE == 18)            ?
                    sample_24n
88                : ( (selector == -25 && (MODE == 14 || MODE == 22)) ?
                    sample_58n
89                : ( (selector == -25 && (MODE == 15 || MODE == 21)) ?
                    sample_44n
90                : ( (selector == -25 && (MODE == 16 || MODE == 20)) ?
                    sample_36n
91                : ( (selector == -25 && (MODE == 17 || MODE == 19)) ?
                    sample_29n
92                : ( (selector == -25 &&  MODE == 18)            ?
                    sample_23n
93                : ( (selector == -24 && (MODE == 14 || MODE == 22)) ?
                    sample_56n
94                : ( (selector == -24 && (MODE == 15 || MODE == 21)) ?
                    sample_42n
95                : ( (selector == -24 && (MODE == 16 || MODE == 20)) ?
                    sample_34n
96                : ( (selector == -24 && (MODE == 17 || MODE == 19)) ?
                    sample_27n
97                : ( (selector == -24 &&  MODE == 18)            ?
                    sample_22n
98                : ( (selector == -23 && (MODE == 14 || MODE == 22)) ?
                    sample_53n
99                : ( (selector == -23 && (MODE == 15 || MODE == 21)) ?
                    sample_40n
100               : ( (selector == -23 && (MODE == 16 || MODE == 20)) ?
                    sample_33n
101               : ( (selector == -23 && (MODE == 17 || MODE == 19)) ?
                    sample_26n
102               : ( (selector == -23 &&  MODE == 18)            ?
                    sample_21n
103               : ( (selector == -22 && (MODE == 14 || MODE == 22)) ?
                    sample_51n
104               : ( (selector == -22 && (MODE == 15 || MODE == 21)) ?
                    sample_39n
105               : ( (selector == -22 && (MODE == 16 || MODE == 20)) ?
                    sample_31n
106               : ( (selector == -22 && (MODE == 17 || MODE == 19)) ?
                    sample_25n
```

```
107                         : ( (selector == -22 &&  MODE == 18)            ?
                              sample_20n
108                         : ( (selector == -21 && (MODE == 14 || MODE == 22)) ?
                              sample_48n
109                         : ( (selector == -21 && (MODE == 15 || MODE == 21)) ?
                              sample_37n
110                         : ( (selector == -21 && (MODE == 16 || MODE == 20)) ?
                              sample_29n
111                         : ( (selector == -21 && (MODE == 17 || MODE == 19)) ?
                              sample_24n
112                         : ( (selector == -21 &&  MODE == 18)            ?
                              sample_19n
113                         : ( (selector == -20 && (MODE == 14 || MODE == 22)) ?
                              sample_46n
114                         : ( (selector == -20 && (MODE == 15 || MODE == 21)) ?
                              sample_35n
115                         : ( (selector == -20 && (MODE == 16 || MODE == 20)) ?
                              sample_28n
116                         : ( (selector == -20 && (MODE == 17 || MODE == 19)) ?
                              sample_22n
117                         : ( (selector == -20 &&  MODE == 18)            ?
                              sample_18n
118                         : ( (selector == -19 && (MODE == 13 || MODE == 23)) ?
                              sample_63n
119                         : ( (selector == -19 && (MODE == 14 || MODE == 22)) ?
                              sample_43n
120                         : ( (selector == -19 && (MODE == 15 || MODE == 21)) ?
                              sample_33n
121                         : ( (selector == -19 && (MODE == 16 || MODE == 20)) ?
                              sample_26n
122                         : ( (selector == -19 && (MODE == 17 || MODE == 19)) ?
                              sample_21n
123                         : ( (selector == -19 &&  MODE == 18)            ?
                              sample_17n
124                         : ( (selector == -18 && (MODE == 13 || MODE == 23)) ?
                              sample_59n
125                         : ( (selector == -18 && (MODE == 14 || MODE == 22)) ?
                              sample_41n
126                         : ( (selector == -18 && (MODE == 15 || MODE == 21)) ?
                              sample_31n
127                         : ( (selector == -18 && (MODE == 16 || MODE == 20)) ?
                              sample_25n
128                         : ( (selector == -18 && (MODE == 17 || MODE == 19)) ?
                              sample_20n
129                         : ( (selector == -18 &&  MODE == 18)            ?
                              sample_16n
130                         : ( (selector == -17 && (MODE == 13 || MODE == 23)) ?
                              sample_56n
131                         : ( (selector == -17 && (MODE == 14 || MODE == 22)) ?
                              sample_38n
132                         : ( (selector == -17 && (MODE == 15 || MODE == 21)) ?
                              sample_29n
133                         : ( (selector == -17 && (MODE == 16 || MODE == 20)) ?
                              sample_23n
134                         : ( (selector == -17 && (MODE == 17 || MODE == 19)) ?
                              sample_19n
135                         : ( (selector == -17 &&  MODE == 18)            ?
```

```
                    sample_15n
136                 : ( (selector == -16 && (MODE == 13 || MODE == 23)) ?
                    sample_52n
137                 : ( (selector == -16 && (MODE == 14 || MODE == 22)) ?
                    sample_36n
138                 : ( (selector == -16 && (MODE == 15 || MODE == 21)) ?
                    sample_27n
139                 : ( (selector == -16 && (MODE == 16 || MODE == 20)) ?
                    sample_22n
140                 : ( (selector == -16 && (MODE == 17 || MODE == 19)) ?
                    sample_17n
141                 : ( (selector == -16 &&  MODE == 18)            ?
                    sample_14n
142                 : ( (selector == -15 && (MODE == 13 || MODE == 23)) ?
                    sample_49n
143                 : ( (selector == -15 && (MODE == 14 || MODE == 22)) ?
                    sample_33n
144                 : ( (selector == -15 && (MODE == 15 || MODE == 21)) ?
                    sample_25n
145                 : ( (selector == -15 && (MODE == 16 || MODE == 20)) ?
                    sample_20n
146                 : ( (selector == -15 && (MODE == 17 || MODE == 19)) ?
                    sample_16n
147                 : ( (selector == -15 &&  MODE == 18)            ?
                    sample_13n
148                 : ( (selector == -14 && (MODE == 13 || MODE == 23)) ?
                    sample_45n
149                 : ( (selector == -14 && (MODE == 14 || MODE == 22)) ?
                    sample_31n
150                 : ( (selector == -14 && (MODE == 15 || MODE == 21)) ?
                    sample_23n
151                 : ( (selector == -14 && (MODE == 16 || MODE == 20)) ?
                    sample_19n
152                 : ( (selector == -14 && (MODE == 17 || MODE == 19)) ?
                    sample_15n
153                 : ( (selector == -14 &&  MODE == 18)            ?
                    sample_12n
154                 : ( (selector == -13 && (MODE == 13 || MODE == 23)) ?
                    sample_42n
155                 : ( (selector == -13 && (MODE == 14 || MODE == 22)) ?
                    sample_29n
156                 : ( (selector == -13 && (MODE == 15 || MODE == 21)) ?
                    sample_22n
157                 : ( (selector == -13 && (MODE == 16 || MODE == 20)) ?
                    sample_17n
158                 : ( (selector == -13 && (MODE == 17 || MODE == 19)) ?
                    sample_14n
159                 : ( (selector == -13 &&  MODE == 18)            ?
                    sample_11n
160                 : ( (selector == -12 && (MODE == 13 || MODE == 23)) ?
                    sample_38n
161                 : ( (selector == -12 && (MODE == 14 || MODE == 22)) ?
                    sample_26n
162                 : ( (selector == -12 && (MODE == 15 || MODE == 21)) ?
                    sample_20n
163                 : ( (selector == -12 && (MODE == 16 || MODE == 20)) ?
                    sample_16n
```

```
164                     : ( (selector == -12 && (MODE == 17 || MODE == 19)) ?
                            sample_13n
165                     : ( (selector == -12 &&  MODE == 18)            ?
                            sample_10n
166                     : ( (selector == -11 && (MODE == 12 || MODE == 24)) ?
                            sample_63n
167                     : ( (selector == -11 && (MODE == 13 || MODE == 23)) ?
                            sample_35n
168                     : ( (selector == -11 && (MODE == 14 || MODE == 22)) ?
                            sample_24n
169                     : ( (selector == -11 && (MODE == 15 || MODE == 21)) ?
                            sample_18n
170                     : ( (selector == -11 && (MODE == 16 || MODE == 20)) ?
                            sample_14n
171                     : ( (selector == -11 && (MODE == 17 || MODE == 19)) ?
                            sample_11n
172                     : ( (selector == -11 &&  MODE == 18)            ? sample_9n
173                     : ( (selector == -10 && (MODE == 12 || MODE == 24)) ?
                            sample_57n
174                     : ( (selector == -10 && (MODE == 13 || MODE == 23)) ?
                            sample_31n
175                     : ( (selector == -10 && (MODE == 14 || MODE == 22)) ?
                            sample_21n
176                     : ( (selector == -10 && (MODE == 15 || MODE == 21)) ?
                            sample_16n
177                     : ( (selector == -10 && (MODE == 16 || MODE == 20)) ?
                            sample_13n
178                     : ( (selector == -10 && (MODE == 17 || MODE == 19)) ?
                            sample_10n
179                     : ( (selector == -10 &&  MODE == 18)            ? sample_8n
180                     : ( (selector ==  -9 && (MODE == 12 || MODE == 24)) ?
                            sample_50n
181                     : ( (selector ==  -9 && (MODE == 13 || MODE == 23)) ?
                            sample_27n
182                     : ( (selector ==  -9 && (MODE == 14 || MODE == 22)) ?
                            sample_19n
183                     : ( (selector ==  -9 && (MODE == 15 || MODE == 21)) ?
                            sample_14n
184                     : ( (selector ==  -9 && (MODE == 16 || MODE == 20)) ?
                            sample_11n
185                     : ( (selector ==  -9 && (MODE == 17 || MODE == 19)) ?
                            sample_9n
186                     : ( (selector ==  -9 &&  MODE == 18)            ? sample_7n
187                     : ( (selector ==  -8 && (MODE == 12 || MODE == 24)) ?
                            sample_44n
188                     : ( (selector ==  -8 && (MODE == 13 || MODE == 23)) ?
                            sample_24n
189                     : ( (selector ==  -8 && (MODE == 14 || MODE == 22)) ?
                            sample_16n
190                     : ( (selector ==  -8 && (MODE == 15 || MODE == 21)) ?
                            sample_12n
191                     : ( (selector ==  -8 && (MODE == 16 || MODE == 20)) ?
                            sample_10n
192                     : ( (selector ==  -8 && (MODE == 17 || MODE == 19)) ?
                            sample_8n
193                     : ( (selector ==  -8 &&  MODE == 18)            ? sample_6n
194                     : ( (selector ==  -7 && (MODE == 12 || MODE == 24)) ?
```

```
                    sample_37n
195                 : ( (selector ==  -7 && (MODE == 13 || MODE == 23)) ?
                    sample_20n
196                 : ( (selector ==  -7 && (MODE == 14 || MODE == 22)) ?
                    sample_14n
197                 : ( (selector ==  -7 && (MODE == 15 || MODE == 21)) ?
                    sample_10n
198                 : ( (selector ==  -7 && (MODE == 16 || MODE == 20)) ?
                    sample_8n
199                 : ( (selector ==  -7 && (MODE == 17 || MODE == 19)) ?
                    sample_6n
200                 : ( (selector ==  -7 &&  MODE == 18)            ? sample_5n
201                 : ( (selector ==  -6 && (MODE == 12 || MODE == 24)) ?
                    sample_31n
202                 : ( (selector ==  -6 && (MODE == 13 || MODE == 23)) ?
                    sample_17n
203                 : ( (selector ==  -6 && (MODE == 14 || MODE == 22)) ?
                    sample_11n
204                 : ( (selector ==  -6 && (MODE == 15 || MODE == 21)) ?
                    sample_8n
205                 : ( (selector ==  -6 && (MODE == 16 || MODE == 20)) ?
                    sample_7n
206                 : ( (selector ==  -6 && (MODE == 17 || MODE == 19)) ?
                    sample_5n
207                 : ( (selector ==  -6 &&  MODE == 18)            ? sample_4n
208                 : ( (selector ==  -5 && (MODE == 11 || MODE == 25)) ?
                    sample_63n
209                 : ( (selector ==  -5 && (MODE == 12 || MODE == 24)) ?
                    sample_25n
210                 : ( (selector ==  -5 && (MODE == 13 || MODE == 23)) ?
                    sample_13n
211                 : ( (selector ==  -5 && (MODE == 14 || MODE == 22)) ?
                    sample_9n
212                 : ( (selector ==  -5 && (MODE == 15 || MODE == 21)) ?
                    sample_7n
213                 : ( (selector ==  -5 && (MODE == 16 || MODE == 20)) ?
                    sample_5n
214                 : ( (selector ==  -5 && (MODE == 17 || MODE == 19)) ?
                    sample_4n
215                 : ( (selector ==  -5 &&  MODE == 18)            ? sample_3n
216                 : ( (selector ==  -4 && (MODE == 11 || MODE == 25)) ?
                    sample_47n
217                 : ( (selector ==  -4 && (MODE == 12 || MODE == 24)) ?
                    sample_18n
218                 : ( (selector ==  -4 && (MODE == 13 || MODE == 23)) ?
                    sample_10n
219                 : ( (selector ==  -4 && (MODE == 14 || MODE == 22)) ?
                    sample_6n
220                 : ( (selector ==  -4 && (MODE == 15 || MODE == 21)) ?
                    sample_5n
221                 : ( (selector ==  -4 && (MODE == 16 || MODE == 20)) ?
                    sample_4n
222                 : ( (selector ==  -4 && (MODE == 17 || MODE == 19)) ?
                    sample_3n
223                 : ( (selector ==  -4 &&  MODE == 18)            ? sample_2n
224                 : ( (selector ==  -3 && (MODE == 11 || MODE == 25)) ?
                    sample_31n
```

```
225                         : ( (selector ==  -3 && (MODE == 12 || MODE == 24)) ?
                              sample_12n
226                         : ( (selector ==  -3 && (MODE == 13 || MODE == 23)) ?
                              sample_6n
227                         : ( (selector ==  -3 && (MODE == 14 || MODE == 22)) ?
                              sample_4n
228                         : ( (selector ==  -3 && (MODE == 15 || MODE == 21)) ?
                              sample_3n
229                         : ( (selector ==  -3 && (MODE == 16 || MODE == 20)) ?
                              sample_2n
230                         : ( (selector ==  -3 && (MODE == 17 || MODE == 19)) ?
                              sample_1n
231                         : ( (selector ==  -3 &&  MODE == 18)            ? sample_1n
232                         : ( (selector ==  -2 && (MODE == 11 || MODE == 25)) ?
                              sample_15n
233                         : ( (selector ==  -2 && (MODE == 12 || MODE == 24)) ?
                              sample_5n
234                         : ( (selector ==  -2 && (MODE == 13 || MODE == 23)) ?
                              sample_3n
235                         : ( (selector ==  -2 && (MODE == 14 || MODE == 22)) ?
                              sample_1n
236                         : ( (selector ==  -2 && (MODE == 15 || MODE == 21)) ?
                              sample_1n
237                         : ( (selector ==  -2 && (MODE == 16 || MODE == 20)) ?
                              sample_1n
238                         : ( (selector ==  -2 && (MODE == 17 || MODE == 19)) ?
                              sample_0n
239                         : ( (selector ==  -2 &&  MODE == 18)            ? sample_0n
240                         : (selector ==  -1 ? sample_NN
241                         : (selector ==   0 ? sample_n0
242                         : (selector ==   1 ? sample_n1
243                         : (selector ==   2 ? sample_n2
244                         : (selector ==   3 ? sample_n3
245                         : (selector ==   4 ? sample_n4
246                         : (selector ==   5 ? sample_n5
247                         : (selector ==   6 ? sample_n6
248                         : (selector ==   7 ? sample_n7
249                         : (selector ==   8 ? sample_n8
250                         : (selector ==   9 ? sample_n9
251                         : (selector ==  10 ? sample_n10
252                         : (selector ==  11 ? sample_n11
253                         : (selector ==  12 ? sample_n12
254                         : (selector ==  13 ? sample_n13
255                         : (selector ==  14 ? sample_n14
256                         : (selector ==  15 ? sample_n15
257                         : (selector ==  16 ? sample_n16
258                         : (selector ==  17 ? sample_n17
259                         : (selector ==  18 ? sample_n18
260                         : (selector ==  19 ? sample_n19
261                         : (selector ==  20 ? sample_n20
262                         : (selector ==  21 ? sample_n21
263                         : (selector ==  22 ? sample_n22
264                         : (selector ==  23 ? sample_n23
265                         : (selector ==  24 ? sample_n24
266                         : (selector ==  25 ? sample_n25
267                         : (selector ==  26 ? sample_n26
268                         : (selector ==  27 ? sample_n27
```

```
                    : (selector ==  28 ? sample_n28
                    : (selector ==  29 ? sample_n29
                    : (selector ==  30 ? sample_n30
                    : (selector ==  31 ? sample_n31
                    : (selector ==  32 ? sample_n32
                    : (selector ==  33 ? sample_n33
                    : (selector ==  34 ? sample_n34
                    : (selector ==  35 ? sample_n35
                    : (selector ==  36 ? sample_n36
                    : (selector ==  37 ? sample_n37
                    : (selector ==  38 ? sample_n38
                    : (selector ==  39 ? sample_n39
                    : (selector ==  40 ? sample_n40
                    : (selector ==  41 ? sample_n41
                    : (selector ==  42 ? sample_n42
                    : (selector ==  43 ? sample_n43
                    : (selector ==  44 ? sample_n44
                    : (selector ==  45 ? sample_n45
                    : (selector ==  46 ? sample_n46
                    : (selector ==  47 ? sample_n47
                    : (selector ==  48 ? sample_n48
                    : (selector ==  49 ? sample_n49
                    : (selector ==  50 ? sample_n50
                    : (selector ==  51 ? sample_n51
                    : (selector ==  52 ? sample_n52
                    : (selector ==  53 ? sample_n53
                    : (selector ==  54 ? sample_n54
                    : (selector ==  55 ? sample_n55
                    : (selector ==  56 ? sample_n56
                    : (selector ==  57 ? sample_n57
                    : (selector ==  58 ? sample_n58
                    : (selector ==  59 ? sample_n59
                    : (selector ==  60 ? sample_n60
                    : (selector ==  61 ? sample_n61
                    : (selector ==  62 ? sample_n62
                    : (selector ==  63 ? sample_n63
                    : 0 )))))))))))))))))))))))))))))))))))))))))))))))))))))))
                      )))))))))))))))))))))))))))))))))))))))))))))))))))))))
                      )))))))))))))))))))))))))))))))))))))))))))))))))))))))
                      )))))))))))))))))))))))))))))))))))))))))))))))))))))))
                      )))))))))))))))))))))))))))
                    : ( (selector == -32 && (MODE == 15 || MODE == 21)) ?
                      sample_57n
                    : ( (selector == -32 && (MODE == 16 || MODE == 20)) ?
                      sample_n46
                    : ( (selector == -32 && (MODE == 17 || MODE == 19)) ?
                      sample_n37
                    : ( (selector == -32 &&  MODE == 18)             ?
                      sample_n30
                    : ( (selector == -31 && (MODE == 15 || MODE == 21)) ?
                      sample_n55
                    : ( (selector == -31 && (MODE == 16 || MODE == 20)) ?
                      sample_n45
                    : ( (selector == -31 && (MODE == 17 || MODE == 19)) ?
                      sample_n36
                    : ( (selector == -31 &&  MODE == 18)             ?
                      sample_n29
```

```
314                         : ( (selector == -30 && (MODE == 15 || MODE == 21)) ?
                              sample_n54
315                         : ( (selector == -30 && (MODE == 16 || MODE == 20)) ?
                              sample_n43
316                         : ( (selector == -30 && (MODE == 17 || MODE == 19)) ?
                              sample_n35
317                         : ( (selector == -30 &&  MODE == 18)            ?
                              sample_n28
318                         : ( (selector == -29 && (MODE == 15 || MODE == 21)) ?
                              sample_n52
319                         : ( (selector == -29 && (MODE == 16 || MODE == 20)) ?
                              sample_n42
320                         : ( (selector == -29 && (MODE == 17 || MODE == 19)) ?
                              sample_n33
321                         : ( (selector == -29 &&  MODE == 18)            ?
                              sample_n27
322                         : ( (selector == -28 && (MODE == 15 || MODE == 21)) ?
                              sample_n50
323                         : ( (selector == -28 && (MODE == 16 || MODE == 20)) ?
                              sample_n40
324                         : ( (selector == -28 && (MODE == 17 || MODE == 19)) ?
                              sample_n32
325                         : ( (selector == -28 &&  MODE == 18)            ?
                              sample_n26
326                         : ( (selector == -27 && (MODE == 14 || MODE == 22)) ?
                              sample_n63
327                         : ( (selector == -27 && (MODE == 15 || MODE == 21)) ?
                              sample_n48
328                         : ( (selector == -27 && (MODE == 16 || MODE == 20)) ?
                              sample_n39
329                         : ( (selector == -27 && (MODE == 17 || MODE == 19)) ?
                              sample_n31
330                         : ( (selector == -27 &&  MODE == 18)            ?
                              sample_n25
331                         : ( (selector == -26 && (MODE == 14 || MODE == 22)) ?
                              sample_n61
332                         : ( (selector == -26 && (MODE == 15 || MODE == 21)) ?
                              sample_n46
333                         : ( (selector == -26 && (MODE == 16 || MODE == 20)) ?
                              sample_n37
334                         : ( (selector == -26 && (MODE == 17 || MODE == 19)) ?
                              sample_n30
335                         : ( (selector == -26 &&  MODE == 18)            ?
                              sample_n24
336                         : ( (selector == -25 && (MODE == 14 || MODE == 22)) ?
                              sample_n58
337                         : ( (selector == -25 && (MODE == 15 || MODE == 21)) ?
                              sample_n44
338                         : ( (selector == -25 && (MODE == 16 || MODE == 20)) ?
                              sample_n36
339                         : ( (selector == -25 && (MODE == 17 || MODE == 19)) ?
                              sample_n29
340                         : ( (selector == -25 &&  MODE == 18)            ?
                              sample_n23
341                         : ( (selector == -24 && (MODE == 14 || MODE == 22)) ?
                              sample_n56
342                         : ( (selector == -24 && (MODE == 15 || MODE == 21)) ?
```

```
                    sample_n42
343                 : ( (selector == -24 && (MODE == 16 || MODE == 20)) ?
                    sample_n34
344                 : ( (selector == -24 && (MODE == 17 || MODE == 19)) ?
                    sample_n27
345                 : ( (selector == -24 &&  MODE == 18)            ?
                    sample_n22
346                 : ( (selector == -23 && (MODE == 14 || MODE == 22)) ?
                    sample_n53
347                 : ( (selector == -23 && (MODE == 15 || MODE == 21)) ?
                    sample_n40
348                 : ( (selector == -23 && (MODE == 16 || MODE == 20)) ?
                    sample_n33
349                 : ( (selector == -23 && (MODE == 17 || MODE == 19)) ?
                    sample_n26
350                 : ( (selector == -23 &&  MODE == 18)            ?
                    sample_n21
351                 : ( (selector == -22 && (MODE == 14 || MODE == 22)) ?
                    sample_n51
352                 : ( (selector == -22 && (MODE == 15 || MODE == 21)) ?
                    sample_n39
353                 : ( (selector == -22 && (MODE == 16 || MODE == 20)) ?
                    sample_n31
354                 : ( (selector == -22 && (MODE == 17 || MODE == 19)) ?
                    sample_n25
355                 : ( (selector == -22 &&  MODE == 18)            ?
                    sample_n20
356                 : ( (selector == -21 && (MODE == 14 || MODE == 22)) ?
                    sample_n48
357                 : ( (selector == -21 && (MODE == 15 || MODE == 21)) ?
                    sample_n37
358                 : ( (selector == -21 && (MODE == 16 || MODE == 20)) ?
                    sample_n29
359                 : ( (selector == -21 && (MODE == 17 || MODE == 19)) ?
                    sample_n24
360                 : ( (selector == -21 &&  MODE == 18)            ?
                    sample_n19
361                 : ( (selector == -20 && (MODE == 14 || MODE == 22)) ?
                    sample_n46
362                 : ( (selector == -20 && (MODE == 15 || MODE == 21)) ?
                    sample_n35
363                 : ( (selector == -20 && (MODE == 16 || MODE == 20)) ?
                    sample_n28
364                 : ( (selector == -20 && (MODE == 17 || MODE == 19)) ?
                    sample_n22
365                 : ( (selector == -20 &&  MODE == 18)            ?
                    sample_n18
366                 : ( (selector == -19 && (MODE == 13 || MODE == 23)) ?
                    sample_n63
367                 : ( (selector == -19 && (MODE == 14 || MODE == 22)) ?
                    sample_n43
368                 : ( (selector == -19 && (MODE == 15 || MODE == 21)) ?
                    sample_n33
369                 : ( (selector == -19 && (MODE == 16 || MODE == 20)) ?
                    sample_n26
370                 : ( (selector == -19 && (MODE == 17 || MODE == 19)) ?
                    sample_n21
```

```
371                         : ( (selector == -19 &&  MODE == 18)            ?
                              sample_n17
372                         : ( (selector == -18 && (MODE == 13 || MODE == 23)) ?
                              sample_n59
373                         : ( (selector == -18 && (MODE == 14 || MODE == 22)) ?
                              sample_n41
374                         : ( (selector == -18 && (MODE == 15 || MODE == 21)) ?
                              sample_n31
375                         : ( (selector == -18 && (MODE == 16 || MODE == 20)) ?
                              sample_n25
376                         : ( (selector == -18 && (MODE == 17 || MODE == 19)) ?
                              sample_n20
377                         : ( (selector == -18 &&  MODE == 18)            ?
                              sample_n16
378                         : ( (selector == -17 && (MODE == 13 || MODE == 23)) ?
                              sample_n56
379                         : ( (selector == -17 && (MODE == 14 || MODE == 22)) ?
                              sample_n38
380                         : ( (selector == -17 && (MODE == 15 || MODE == 21)) ?
                              sample_n29
381                         : ( (selector == -17 && (MODE == 16 || MODE == 20)) ?
                              sample_n23
382                         : ( (selector == -17 && (MODE == 17 || MODE == 19)) ?
                              sample_n19
383                         : ( (selector == -17 &&  MODE == 18)            ?
                              sample_n15
384                         : ( (selector == -16 && (MODE == 13 || MODE == 23)) ?
                              sample_n52
385                         : ( (selector == -16 && (MODE == 14 || MODE == 22)) ?
                              sample_n36
386                         : ( (selector == -16 && (MODE == 15 || MODE == 21)) ?
                              sample_n27
387                         : ( (selector == -16 && (MODE == 16 || MODE == 20)) ?
                              sample_n22
388                         : ( (selector == -16 && (MODE == 17 || MODE == 19)) ?
                              sample_n17
389                         : ( (selector == -16 &&  MODE == 18)            ?
                              sample_n14
390                         : ( (selector == -15 && (MODE == 13 || MODE == 23)) ?
                              sample_n49
391                         : ( (selector == -15 && (MODE == 14 || MODE == 22)) ?
                              sample_n33
392                         : ( (selector == -15 && (MODE == 15 || MODE == 21)) ?
                              sample_n25
393                         : ( (selector == -15 && (MODE == 16 || MODE == 20)) ?
                              sample_n20
394                         : ( (selector == -15 && (MODE == 17 || MODE == 19)) ?
                              sample_n16
395                         : ( (selector == -15 &&  MODE == 18)            ?
                              sample_n13
396                         : ( (selector == -14 && (MODE == 13 || MODE == 23)) ?
                              sample_n45
397                         : ( (selector == -14 && (MODE == 14 || MODE == 22)) ?
                              sample_n31
398                         : ( (selector == -14 && (MODE == 15 || MODE == 21)) ?
                              sample_n23
399                         : ( (selector == -14 && (MODE == 16 || MODE == 20)) ?
```

```
                    sample_n19
400                 : ( (selector == -14 && (MODE == 17 || MODE == 19)) ?
                    sample_n15
401                 : ( (selector == -14 &&  MODE == 18)            ?
                    sample_n12
402                 : ( (selector == -13 && (MODE == 13 || MODE == 23)) ?
                    sample_n42
403                 : ( (selector == -13 && (MODE == 14 || MODE == 22)) ?
                    sample_n29
404                 : ( (selector == -13 && (MODE == 15 || MODE == 21)) ?
                    sample_n22
405                 : ( (selector == -13 && (MODE == 16 || MODE == 20)) ?
                    sample_n17
406                 : ( (selector == -13 && (MODE == 17 || MODE == 19)) ?
                    sample_n14
407                 : ( (selector == -13 &&  MODE == 18)            ?
                    sample_n11
408                 : ( (selector == -12 && (MODE == 13 || MODE == 23)) ?
                    sample_n38
409                 : ( (selector == -12 && (MODE == 14 || MODE == 22)) ?
                    sample_n26
410                 : ( (selector == -12 && (MODE == 15 || MODE == 21)) ?
                    sample_n20
411                 : ( (selector == -12 && (MODE == 16 || MODE == 20)) ?
                    sample_n16
412                 : ( (selector == -12 && (MODE == 17 || MODE == 19)) ?
                    sample_n13
413                 : ( (selector == -12 &&  MODE == 18)            ?
                    sample_n10
414                 : ( (selector == -11 && (MODE == 12 || MODE == 24)) ?
                    sample_n63
415                 : ( (selector == -11 && (MODE == 13 || MODE == 23)) ?
                    sample_n35
416                 : ( (selector == -11 && (MODE == 14 || MODE == 22)) ?
                    sample_n24
417                 : ( (selector == -11 && (MODE == 15 || MODE == 21)) ?
                    sample_n18
418                 : ( (selector == -11 && (MODE == 16 || MODE == 20)) ?
                    sample_n14
419                 : ( (selector == -11 && (MODE == 17 || MODE == 19)) ?
                    sample_n11
420                 : ( (selector == -11 &&  MODE == 18)            ? sample_n9
421                 : ( (selector == -10 && (MODE == 12 || MODE == 24)) ?
                    sample_n57
422                 : ( (selector == -10 && (MODE == 13 || MODE == 23)) ?
                    sample_n31
423                 : ( (selector == -10 && (MODE == 14 || MODE == 22)) ?
                    sample_n21
424                 : ( (selector == -10 && (MODE == 15 || MODE == 21)) ?
                    sample_n16
425                 : ( (selector == -10 && (MODE == 16 || MODE == 20)) ?
                    sample_n13
426                 : ( (selector == -10 && (MODE == 17 || MODE == 19)) ?
                    sample_n10
427                 : ( (selector == -10 &&  MODE == 18)            ? sample_n8
428                 : ( (selector ==  -9 && (MODE == 12 || MODE == 24)) ?
                    sample_n50
```

```
429                     : ( (selector ==  -9 && (MODE == 13 || MODE == 23)) ?
                            sample_n27
430                     : ( (selector ==  -9 && (MODE == 14 || MODE == 22)) ?
                            sample_n19
431                     : ( (selector ==  -9 && (MODE == 15 || MODE == 21)) ?
                            sample_n14
432                     : ( (selector ==  -9 && (MODE == 16 || MODE == 20)) ?
                            sample_n11
433                     : ( (selector ==  -9 && (MODE == 17 || MODE == 19)) ?
                            sample_n9
434                     : ( (selector ==  -9 &&  MODE == 18)            ? sample_n7
435                     : ( (selector ==  -8 && (MODE == 12 || MODE == 24)) ?
                            sample_n44
436                     : ( (selector ==  -8 && (MODE == 13 || MODE == 23)) ?
                            sample_n24
437                     : ( (selector ==  -8 && (MODE == 14 || MODE == 22)) ?
                            sample_n16
438                     : ( (selector ==  -8 && (MODE == 15 || MODE == 21)) ?
                            sample_n12
439                     : ( (selector ==  -8 && (MODE == 16 || MODE == 20)) ?
                            sample_n10
440                     : ( (selector ==  -8 && (MODE == 17 || MODE == 19)) ?
                            sample_n8
441                     : ( (selector ==  -8 &&  MODE == 18)            ? sample_n6
442                     : ( (selector ==  -7 && (MODE == 12 || MODE == 24)) ?
                            sample_n37
443                     : ( (selector ==  -7 && (MODE == 13 || MODE == 23)) ?
                            sample_n20
444                     : ( (selector ==  -7 && (MODE == 14 || MODE == 22)) ?
                            sample_n14
445                     : ( (selector ==  -7 && (MODE == 15 || MODE == 21)) ?
                            sample_n10
446                     : ( (selector ==  -7 && (MODE == 16 || MODE == 20)) ?
                            sample_n8
447                     : ( (selector ==  -7 && (MODE == 17 || MODE == 19)) ?
                            sample_n6
448                     : ( (selector ==  -7 &&  MODE == 18)            ? sample_n5
449                     : ( (selector ==  -6 && (MODE == 12 || MODE == 24)) ?
                            sample_n31
450                     : ( (selector ==  -6 && (MODE == 13 || MODE == 23)) ?
                            sample_n17
451                     : ( (selector ==  -6 && (MODE == 14 || MODE == 22)) ?
                            sample_n11
452                     : ( (selector ==  -6 && (MODE == 15 || MODE == 21)) ?
                            sample_n8
453                     : ( (selector ==  -6 && (MODE == 16 || MODE == 20)) ?
                            sample_n7
454                     : ( (selector ==  -6 && (MODE == 17 || MODE == 19)) ?
                            sample_n5
455                     : ( (selector ==  -6 &&  MODE == 18)            ? sample_n4
456                     : ( (selector ==  -5 && (MODE == 11 || MODE == 25)) ?
                            sample_n63
457                     : ( (selector ==  -5 && (MODE == 12 || MODE == 24)) ?
                            sample_n25
458                     : ( (selector ==  -5 && (MODE == 13 || MODE == 23)) ?
                            sample_n13
459                     : ( (selector ==  -5 && (MODE == 14 || MODE == 22)) ?
```

```
                    sample_n9
460                 : ( (selector ==  -5 && (MODE == 15 || MODE == 21)) ?
                    sample_n7
461                 : ( (selector ==  -5 && (MODE == 16 || MODE == 20)) ?
                    sample_n5
462                 : ( (selector ==  -5 && (MODE == 17 || MODE == 19)) ?
                    sample_n4
463                 : ( (selector ==  -5 &&  MODE == 18)            ? sample_n3
464                 : ( (selector ==  -4 && (MODE == 11 || MODE == 25)) ?
                    sample_n47
465                 : ( (selector ==  -4 && (MODE == 12 || MODE == 24)) ?
                    sample_n18
466                 : ( (selector ==  -4 && (MODE == 13 || MODE == 23)) ?
                    sample_n10
467                 : ( (selector ==  -4 && (MODE == 14 || MODE == 22)) ?
                    sample_n6
468                 : ( (selector ==  -4 && (MODE == 15 || MODE == 21)) ?
                    sample_n5
469                 : ( (selector ==  -4 && (MODE == 16 || MODE == 20)) ?
                    sample_n4
470                 : ( (selector ==  -4 && (MODE == 17 || MODE == 19)) ?
                    sample_n3
471                 : ( (selector ==  -4 &&  MODE == 18)            ? sample_n2
472                 : ( (selector ==  -3 && (MODE == 11 || MODE == 25)) ?
                    sample_n31
473                 : ( (selector ==  -3 && (MODE == 12 || MODE == 24)) ?
                    sample_n12
474                 : ( (selector ==  -3 && (MODE == 13 || MODE == 23)) ?
                    sample_n6
475                 : ( (selector ==  -3 && (MODE == 14 || MODE == 22)) ?
                    sample_n4
476                 : ( (selector ==  -3 && (MODE == 15 || MODE == 21)) ?
                    sample_n3
477                 : ( (selector ==  -3 && (MODE == 16 || MODE == 20)) ?
                    sample_n2
478                 : ( (selector ==  -3 && (MODE == 17 || MODE == 19)) ?
                    sample_n1
479                 : ( (selector ==  -3 &&  MODE == 18)            ? sample_n1
480                 : ( (selector ==  -2 && (MODE == 11 || MODE == 25)) ?
                    sample_n15
481                 : ( (selector ==  -2 && (MODE == 12 || MODE == 24)) ?
                    sample_n5
482                 : ( (selector ==  -2 && (MODE == 13 || MODE == 23)) ?
                    sample_n3
483                 : ( (selector ==  -2 && (MODE == 14 || MODE == 22)) ?
                    sample_n1
484                 : ( (selector ==  -2 && (MODE == 15 || MODE == 21)) ?
                    sample_n1
485                 : ( (selector ==  -2 && (MODE == 16 || MODE == 20)) ?
                    sample_n1
486                 : ( (selector ==  -2 && (MODE == 17 || MODE == 19)) ?
                    sample_n0
487                 : ( (selector ==  -2 &&  MODE == 18)            ? sample_n0
488                 : (selector ==  -1 ? sample_NN
489                 : (selector ==   0 ? sample_0n
490                 : (selector ==   1 ? sample_1n
491                 : (selector ==   2 ? sample_2n
```

```
                        : (selector ==   3 ? sample_3n
                        : (selector ==   4 ? sample_4n
                        : (selector ==   5 ? sample_5n
                        : (selector ==   6 ? sample_6n
                        : (selector ==   7 ? sample_7n
                        : (selector ==   8 ? sample_8n
                        : (selector ==   9 ? sample_9n
                        : (selector ==  10 ? sample_10n
                        : (selector ==  11 ? sample_11n
                        : (selector ==  12 ? sample_12n
                        : (selector ==  13 ? sample_13n
                        : (selector ==  14 ? sample_14n
                        : (selector ==  15 ? sample_15n
                        : (selector ==  16 ? sample_16n
                        : (selector ==  17 ? sample_17n
                        : (selector ==  18 ? sample_18n
                        : (selector ==  19 ? sample_19n
                        : (selector ==  20 ? sample_20n
                        : (selector ==  21 ? sample_21n
                        : (selector ==  22 ? sample_22n
                        : (selector ==  23 ? sample_23n
                        : (selector ==  24 ? sample_24n
                        : (selector ==  25 ? sample_25n
                        : (selector ==  26 ? sample_26n
                        : (selector ==  27 ? sample_27n
                        : (selector ==  28 ? sample_28n
                        : (selector ==  29 ? sample_29n
                        : (selector ==  30 ? sample_30n
                        : (selector ==  31 ? sample_31n
                        : (selector ==  32 ? sample_32n
                        : (selector ==  33 ? sample_33n
                        : (selector ==  34 ? sample_34n
                        : (selector ==  35 ? sample_35n
                        : (selector ==  36 ? sample_36n
                        : (selector ==  37 ? sample_37n
                        : (selector ==  38 ? sample_38n
                        : (selector ==  39 ? sample_39n
                        : (selector ==  40 ? sample_40n
                        : (selector ==  41 ? sample_41n
                        : (selector ==  42 ? sample_42n
                        : (selector ==  43 ? sample_43n
                        : (selector ==  44 ? sample_44n
                        : (selector ==  45 ? sample_45n
                        : (selector ==  46 ? sample_46n
                        : (selector ==  47 ? sample_47n
                        : (selector ==  48 ? sample_48n
                        : (selector ==  49 ? sample_49n
                        : (selector ==  50 ? sample_50n
                        : (selector ==  51 ? sample_51n
                        : (selector ==  52 ? sample_52n
                        : (selector ==  53 ? sample_53n
                        : (selector ==  54 ? sample_54n
                        : (selector ==  55 ? sample_55n
                        : (selector ==  56 ? sample_56n
                        : (selector ==  57 ? sample_57n
                        : (selector ==  58 ? sample_58n
                        : (selector ==  59 ? sample_59n
```

```
549                         : (selector ==  60 ? sample_60n
550                         : (selector ==  61 ? sample_61n
551                         : (selector ==  62 ? sample_62n
552                         : (selector ==  63 ? sample_63n
553                         : 0 ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                              ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                              ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                              ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                              ))))))))))))))))))))))))));
554
555     assign ref2 = MODE < 18 ? ( (selector == -32 && (MODE == 15 || MODE == 21)) ?
            sample_55n
556                         : ( (selector == -32 && (MODE == 16 || MODE == 20)) ?
                              sample_45n
557                         : ( (selector == -32 && (MODE == 17 || MODE == 19)) ?
                              sample_36n
558                         : ( (selector == -32 &&  MODE == 18)             ?
                              sample_29n
559                         : ( (selector == -31 && (MODE == 15 || MODE == 21)) ?
                              sample_54n
560                         : ( (selector == -31 && (MODE == 16 || MODE == 20)) ?
                              sample_43n
561                         : ( (selector == -31 && (MODE == 17 || MODE == 19)) ?
                              sample_35n
562                         : ( (selector == -31 &&  MODE == 18)             ?
                              sample_28n
563                         : ( (selector == -30 && (MODE == 15 || MODE == 21)) ?
                              sample_52n
564                         : ( (selector == -30 && (MODE == 16 || MODE == 20)) ?
                              sample_42n
565                         : ( (selector == -30 && (MODE == 17 || MODE == 19)) ?
                              sample_33n
566                         : ( (selector == -30 &&  MODE == 18)             ?
                              sample_27n
567                         : ( (selector == -29 && (MODE == 15 || MODE == 21)) ?
                              sample_50n
568                         : ( (selector == -29 && (MODE == 16 || MODE == 20)) ?
                              sample_40n
569                         : ( (selector == -29 && (MODE == 17 || MODE == 19)) ?
                              sample_32n
570                         : ( (selector == -29 &&  MODE == 18)             ?
                              sample_26n
571                         : ( (selector == -28 && (MODE == 14 || MODE == 22)) ?
                              sample_63n
572                         : ( (selector == -28 && (MODE == 15 || MODE == 21)) ?
                              sample_48n
573                         : ( (selector == -28 && (MODE == 16 || MODE == 20)) ?
                              sample_39n
574                         : ( (selector == -28 && (MODE == 17 || MODE == 19)) ?
                              sample_31n
575                         : ( (selector == -28 &&  MODE == 18)             ?
                              sample_25n
576                         : ( (selector == -27 && (MODE == 14 || MODE == 22)) ?
                              sample_61n
577                         : ( (selector == -27 && (MODE == 15 || MODE == 21)) ?
                              sample_46n
578                         : ( (selector == -27 && (MODE == 16 || MODE == 20)) ?
```

```
                    sample_37n
579                 : ( (selector == -27 && (MODE == 17 || MODE == 19)) ?
                    sample_30n
580                 : ( (selector == -27 &&  MODE == 18)            ?
                    sample_24n
581                 : ( (selector == -26 && (MODE == 14 || MODE == 22)) ?
                    sample_58n
582                 : ( (selector == -26 && (MODE == 15 || MODE == 21)) ?
                    sample_44n
583                 : ( (selector == -26 && (MODE == 16 || MODE == 20)) ?
                    sample_36n
584                 : ( (selector == -26 && (MODE == 17 || MODE == 19)) ?
                    sample_29n
585                 : ( (selector == -26 &&  MODE == 18)            ?
                    sample_23n
586                 : ( (selector == -25 && (MODE == 14 || MODE == 22)) ?
                    sample_56n
587                 : ( (selector == -25 && (MODE == 15 || MODE == 21)) ?
                    sample_42n
588                 : ( (selector == -25 && (MODE == 16 || MODE == 20)) ?
                    sample_34n
589                 : ( (selector == -25 && (MODE == 17 || MODE == 19)) ?
                    sample_27n
590                 : ( (selector == -25 &&  MODE == 18)            ?
                    sample_22n
591                 : ( (selector == -24 && (MODE == 14 || MODE == 22)) ?
                    sample_53n
592                 : ( (selector == -24 && (MODE == 15 || MODE == 21)) ?
                    sample_40n
593                 : ( (selector == -24 && (MODE == 16 || MODE == 20)) ?
                    sample_33n
594                 : ( (selector == -24 && (MODE == 17 || MODE == 19)) ?
                    sample_26n
595                 : ( (selector == -24 &&  MODE == 18)            ?
                    sample_21n
596                 : ( (selector == -23 && (MODE == 14 || MODE == 22)) ?
                    sample_51n
597                 : ( (selector == -23 && (MODE == 15 || MODE == 21)) ?
                    sample_39n
598                 : ( (selector == -23 && (MODE == 16 || MODE == 20)) ?
                    sample_31n
599                 : ( (selector == -23 && (MODE == 17 || MODE == 19)) ?
                    sample_25n
600                 : ( (selector == -23 &&  MODE == 18)            ?
                    sample_20n
601                 : ( (selector == -22 && (MODE == 14 || MODE == 22)) ?
                    sample_48n
602                 : ( (selector == -22 && (MODE == 15 || MODE == 21)) ?
                    sample_37n
603                 : ( (selector == -22 && (MODE == 16 || MODE == 20)) ?
                    sample_29n
604                 : ( (selector == -22 && (MODE == 17 || MODE == 19)) ?
                    sample_24n
605                 : ( (selector == -22 &&  MODE == 18)            ?
                    sample_19n
606                 : ( (selector == -21 && (MODE == 14 || MODE == 22)) ?
                    sample_46n
```

```
607                     : ( (selector == -21 && (MODE == 15 || MODE == 21)) ?
                          sample_35n
608                     : ( (selector == -21 && (MODE == 16 || MODE == 20)) ?
                          sample_28n
609                     : ( (selector == -21 && (MODE == 17 || MODE == 19)) ?
                          sample_22n
610                     : ( (selector == -21 &&  MODE == 18)            ?
                          sample_18n
611                     : ( (selector == -20 && (MODE == 13 || MODE == 23)) ?
                          sample_63n
612                     : ( (selector == -20 && (MODE == 14 || MODE == 22)) ?
                          sample_43n
613                     : ( (selector == -20 && (MODE == 15 || MODE == 21)) ?
                          sample_33n
614                     : ( (selector == -20 && (MODE == 16 || MODE == 20)) ?
                          sample_26n
615                     : ( (selector == -20 && (MODE == 17 || MODE == 19)) ?
                          sample_21n
616                     : ( (selector == -20 &&  MODE == 18)            ?
                          sample_17n
617                     : ( (selector == -19 && (MODE == 13 || MODE == 23)) ?
                          sample_59n
618                     : ( (selector == -19 && (MODE == 14 || MODE == 22)) ?
                          sample_41n
619                     : ( (selector == -19 && (MODE == 15 || MODE == 21)) ?
                          sample_31n
620                     : ( (selector == -19 && (MODE == 16 || MODE == 20)) ?
                          sample_25n
621                     : ( (selector == -19 && (MODE == 17 || MODE == 19)) ?
                          sample_20n
622                     : ( (selector == -19 &&  MODE == 18)            ?
                          sample_16n
623                     : ( (selector == -18 && (MODE == 13 || MODE == 23)) ?
                          sample_56n
624                     : ( (selector == -18 && (MODE == 14 || MODE == 22)) ?
                          sample_38n
625                     : ( (selector == -18 && (MODE == 15 || MODE == 21)) ?
                          sample_29n
626                     : ( (selector == -18 && (MODE == 16 || MODE == 20)) ?
                          sample_23n
627                     : ( (selector == -18 && (MODE == 17 || MODE == 19)) ?
                          sample_19n
628                     : ( (selector == -18 &&  MODE == 18)            ?
                          sample_15n
629                     : ( (selector == -17 && (MODE == 13 || MODE == 23)) ?
                          sample_52n
630                     : ( (selector == -17 && (MODE == 14 || MODE == 22)) ?
                          sample_36n
631                     : ( (selector == -17 && (MODE == 15 || MODE == 21)) ?
                          sample_27n
632                     : ( (selector == -17 && (MODE == 16 || MODE == 20)) ?
                          sample_22n
633                     : ( (selector == -17 && (MODE == 17 || MODE == 19)) ?
                          sample_17n
634                     : ( (selector == -17 &&  MODE == 18)            ?
                          sample_14n
635                     : ( (selector == -16 && (MODE == 13 || MODE == 23)) ?
```

```
                    sample_49n
636                 : ( (selector == -16 && (MODE == 14 || MODE == 22)) ?
                    sample_33n
637                 : ( (selector == -16 && (MODE == 15 || MODE == 21)) ?
                    sample_25n
638                 : ( (selector == -16 && (MODE == 16 || MODE == 20)) ?
                    sample_20n
639                 : ( (selector == -16 && (MODE == 17 || MODE == 19)) ?
                    sample_16n
640                 : ( (selector == -16 &&  MODE == 18)            ?
                    sample_13n
641                 : ( (selector == -15 && (MODE == 13 || MODE == 23)) ?
                    sample_45n
642                 : ( (selector == -15 && (MODE == 14 || MODE == 22)) ?
                    sample_31n
643                 : ( (selector == -15 && (MODE == 15 || MODE == 21)) ?
                    sample_23n
644                 : ( (selector == -15 && (MODE == 16 || MODE == 20)) ?
                    sample_19n
645                 : ( (selector == -15 && (MODE == 17 || MODE == 19)) ?
                    sample_15n
646                 : ( (selector == -15 &&  MODE == 18)            ?
                    sample_12n
647                 : ( (selector == -14 && (MODE == 13 || MODE == 23)) ?
                    sample_42n
648                 : ( (selector == -14 && (MODE == 14 || MODE == 22)) ?
                    sample_29n
649                 : ( (selector == -14 && (MODE == 15 || MODE == 21)) ?
                    sample_22n
650                 : ( (selector == -14 && (MODE == 16 || MODE == 20)) ?
                    sample_17n
651                 : ( (selector == -14 && (MODE == 17 || MODE == 19)) ?
                    sample_14n
652                 : ( (selector == -14 &&  MODE == 18)            ?
                    sample_11n
653                 : ( (selector == -13 && (MODE == 13 || MODE == 23)) ?
                    sample_38n
654                 : ( (selector == -13 && (MODE == 14 || MODE == 22)) ?
                    sample_26n
655                 : ( (selector == -13 && (MODE == 15 || MODE == 21)) ?
                    sample_20n
656                 : ( (selector == -13 && (MODE == 16 || MODE == 20)) ?
                    sample_16n
657                 : ( (selector == -13 && (MODE == 17 || MODE == 19)) ?
                    sample_13n
658                 : ( (selector == -13 &&  MODE == 18)            ?
                    sample_10n
659                 : ( (selector == -12 && (MODE == 12 || MODE == 24)) ?
                    sample_63n
660                 : ( (selector == -12 && (MODE == 13 || MODE == 23)) ?
                    sample_35n
661                 : ( (selector == -12 && (MODE == 14 || MODE == 22)) ?
                    sample_24n
662                 : ( (selector == -12 && (MODE == 15 || MODE == 21)) ?
                    sample_18n
663                 : ( (selector == -12 && (MODE == 16 || MODE == 20)) ?
                    sample_14n
```

```
664                     : ( (selector == -12 && (MODE == 17 || MODE == 19)) ?
                            sample_11n
665                     : ( (selector == -12 &&  MODE == 18)            ? sample_9n
666                     : ( (selector == -11 && (MODE == 12 || MODE == 24)) ?
                            sample_57n
667                     : ( (selector == -11 && (MODE == 13 || MODE == 23)) ?
                            sample_31n
668                     : ( (selector == -11 && (MODE == 14 || MODE == 22)) ?
                            sample_21n
669                     : ( (selector == -11 && (MODE == 15 || MODE == 21)) ?
                            sample_16n
670                     : ( (selector == -11 && (MODE == 16 || MODE == 20)) ?
                            sample_13n
671                     : ( (selector == -11 && (MODE == 17 || MODE == 19)) ?
                            sample_10n
672                     : ( (selector == -11 &&  MODE == 18)            ? sample_8n
673                     : ( (selector == -10 && (MODE == 12 || MODE == 24)) ?
                            sample_50n
674                     : ( (selector == -10 && (MODE == 13 || MODE == 23)) ?
                            sample_27n
675                     : ( (selector == -10 && (MODE == 14 || MODE == 22)) ?
                            sample_19n
676                     : ( (selector == -10 && (MODE == 15 || MODE == 21)) ?
                            sample_14n
677                     : ( (selector == -10 && (MODE == 16 || MODE == 20)) ?
                            sample_11n
678                     : ( (selector == -10 && (MODE == 17 || MODE == 19)) ?
                            sample_9n
679                     : ( (selector == -10 &&  MODE == 18)            ? sample_7n
680                     : ( (selector ==  -9 && (MODE == 12 || MODE == 24)) ?
                            sample_44n
681                     : ( (selector ==  -9 && (MODE == 13 || MODE == 23)) ?
                            sample_24n
682                     : ( (selector ==  -9 && (MODE == 14 || MODE == 22)) ?
                            sample_16n
683                     : ( (selector ==  -9 && (MODE == 15 || MODE == 21)) ?
                            sample_12n
684                     : ( (selector ==  -9 && (MODE == 16 || MODE == 20)) ?
                            sample_10n
685                     : ( (selector ==  -9 && (MODE == 17 || MODE == 19)) ?
                            sample_8n
686                     : ( (selector ==  -9 &&  MODE == 18)            ? sample_6n
687                     : ( (selector ==  -8 && (MODE == 12 || MODE == 24)) ?
                            sample_37n
688                     : ( (selector ==  -8 && (MODE == 13 || MODE == 23)) ?
                            sample_20n
689                     : ( (selector ==  -8 && (MODE == 14 || MODE == 22)) ?
                            sample_14n
690                     : ( (selector ==  -8 && (MODE == 15 || MODE == 21)) ?
                            sample_10n
691                     : ( (selector ==  -8 && (MODE == 16 || MODE == 20)) ?
                            sample_8n
692                     : ( (selector ==  -8 && (MODE == 17 || MODE == 19)) ?
                            sample_6n
693                     : ( (selector ==  -8 &&  MODE == 18)            ? sample_5n
694                     : ( (selector ==  -7 && (MODE == 12 || MODE == 24)) ?
                            sample_31n
```

```
695                     : ( (selector ==  -7 && (MODE == 13 || MODE == 23)) ?
                            sample_17n
696                     : ( (selector ==  -7 && (MODE == 14 || MODE == 22)) ?
                            sample_11n
697                     : ( (selector ==  -7 && (MODE == 15 || MODE == 21)) ?
                            sample_8n
698                     : ( (selector ==  -7 && (MODE == 16 || MODE == 20)) ?
                            sample_7n
699                     : ( (selector ==  -7 && (MODE == 17 || MODE == 19)) ?
                            sample_5n
700                     : ( (selector ==  -7 &&  MODE == 18)            ? sample_4n
701                     : ( (selector ==  -6 && (MODE == 11 || MODE == 25)) ?
                            sample_63n
702                     : ( (selector ==  -6 && (MODE == 12 || MODE == 24)) ?
                            sample_25n
703                     : ( (selector ==  -6 && (MODE == 13 || MODE == 23)) ?
                            sample_13n
704                     : ( (selector ==  -6 && (MODE == 14 || MODE == 22)) ?
                            sample_9n
705                     : ( (selector ==  -6 && (MODE == 15 || MODE == 21)) ?
                            sample_7n
706                     : ( (selector ==  -6 && (MODE == 16 || MODE == 20)) ?
                            sample_5n
707                     : ( (selector ==  -6 && (MODE == 17 || MODE == 19)) ?
                            sample_4n
708                     : ( (selector ==  -6 &&  MODE == 18)            ? sample_3n
709                     : ( (selector ==  -5 && (MODE == 11 || MODE == 25)) ?
                            sample_47n
710                     : ( (selector ==  -5 && (MODE == 12 || MODE == 24)) ?
                            sample_18n
711                     : ( (selector ==  -5 && (MODE == 13 || MODE == 23)) ?
                            sample_10n
712                     : ( (selector ==  -5 && (MODE == 14 || MODE == 22)) ?
                            sample_6n
713                     : ( (selector ==  -5 && (MODE == 15 || MODE == 21)) ?
                            sample_5n
714                     : ( (selector ==  -5 && (MODE == 16 || MODE == 20)) ?
                            sample_4n
715                     : ( (selector ==  -5 && (MODE == 17 || MODE == 19)) ?
                            sample_3n
716                     : ( (selector ==  -5 &&  MODE == 18)            ? sample_2n
717                     : ( (selector ==  -4 && (MODE == 11 || MODE == 25)) ?
                            sample_31n
718                     : ( (selector ==  -4 && (MODE == 12 || MODE == 24)) ?
                            sample_12n
719                     : ( (selector ==  -4 && (MODE == 13 || MODE == 23)) ?
                            sample_6n
720                     : ( (selector ==  -4 && (MODE == 14 || MODE == 22)) ?
                            sample_4n
721                     : ( (selector ==  -4 && (MODE == 15 || MODE == 21)) ?
                            sample_3n
722                     : ( (selector ==  -4 && (MODE == 16 || MODE == 20)) ?
                            sample_2n
723                     : ( (selector ==  -4 && (MODE == 17 || MODE == 19)) ?
                            sample_1n
724                     : ( (selector ==  -4 &&  MODE == 18)            ? sample_1n
725                     : ( (selector ==  -3 && (MODE == 11 || MODE == 25)) ?
```

```
                        sample_15n
726                   : ( (selector ==  -3 && (MODE == 12 || MODE == 24)) ?
                        sample_5n
727                   : ( (selector ==  -3 && (MODE == 13 || MODE == 23)) ?
                        sample_3n
728                   : ( (selector ==  -3 && (MODE == 14 || MODE == 22)) ?
                        sample_1n
729                   : ( (selector ==  -3 && (MODE == 15 || MODE == 21)) ?
                        sample_1n
730                   : ( (selector ==  -3 && (MODE == 16 || MODE == 20)) ?
                        sample_1n
731                   : ( (selector ==  -3 && (MODE == 17 || MODE == 19)) ?
                        sample_0n
732                   : ( (selector ==  -3 &&  MODE == 18)            ? sample_0n
733                   : (selector ==  -2 ? sample_NN
734                   : (selector ==  -1 ? sample_n0
735                   : (selector ==   0 ? sample_n1
736                   : (selector ==   1 ? sample_n2
737                   : (selector ==   2 ? sample_n3
738                   : (selector ==   3 ? sample_n4
739                   : (selector ==   4 ? sample_n5
740                   : (selector ==   5 ? sample_n6
741                   : (selector ==   6 ? sample_n7
742                   : (selector ==   7 ? sample_n8
743                   : (selector ==   8 ? sample_n9
744                   : (selector ==   9 ? sample_n10
745                   : (selector ==  10 ? sample_n11
746                   : (selector ==  11 ? sample_n12
747                   : (selector ==  12 ? sample_n13
748                   : (selector ==  13 ? sample_n14
749                   : (selector ==  14 ? sample_n15
750                   : (selector ==  15 ? sample_n16
751                   : (selector ==  16 ? sample_n17
752                   : (selector ==  17 ? sample_n18
753                   : (selector ==  18 ? sample_n19
754                   : (selector ==  19 ? sample_n20
755                   : (selector ==  20 ? sample_n21
756                   : (selector ==  21 ? sample_n22
757                   : (selector ==  22 ? sample_n23
758                   : (selector ==  23 ? sample_n24
759                   : (selector ==  24 ? sample_n25
760                   : (selector ==  25 ? sample_n26
761                   : (selector ==  26 ? sample_n27
762                   : (selector ==  27 ? sample_n28
763                   : (selector ==  28 ? sample_n29
764                   : (selector ==  29 ? sample_n30
765                   : (selector ==  30 ? sample_n31
766                   : (selector ==  31 ? sample_n32
767                   : (selector ==  32 ? sample_n33
768                   : (selector ==  33 ? sample_n34
769                   : (selector ==  34 ? sample_n35
770                   : (selector ==  35 ? sample_n36
771                   : (selector ==  36 ? sample_n37
772                   : (selector ==  37 ? sample_n38
773                   : (selector ==  38 ? sample_n39
774                   : (selector ==  39 ? sample_n40
775                   : (selector ==  40 ? sample_n41
```

```
776                     : (selector ==  41 ? sample_n42
777                     : (selector ==  42 ? sample_n43
778                     : (selector ==  43 ? sample_n44
779                     : (selector ==  44 ? sample_n45
780                     : (selector ==  45 ? sample_n46
781                     : (selector ==  46 ? sample_n47
782                     : (selector ==  47 ? sample_n48
783                     : (selector ==  48 ? sample_n49
784                     : (selector ==  49 ? sample_n50
785                     : (selector ==  50 ? sample_n51
786                     : (selector ==  51 ? sample_n52
787                     : (selector ==  52 ? sample_n53
788                     : (selector ==  53 ? sample_n54
789                     : (selector ==  54 ? sample_n55
790                     : (selector ==  55 ? sample_n56
791                     : (selector ==  56 ? sample_n57
792                     : (selector ==  57 ? sample_n58
793                     : (selector ==  58 ? sample_n59
794                     : (selector ==  59 ? sample_n60
795                     : (selector ==  60 ? sample_n61
796                     : (selector ==  61 ? sample_n62
797                     : (selector ==  62 ? sample_n63
798                     : 0 ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                        ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                        ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                        ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                        ))))))))))))))))))))))
799                     : ( (selector == -32 && (MODE == 15 || MODE == 21)) ?
                          sample_n55
800                     : ( (selector == -32 && (MODE == 16 || MODE == 20)) ?
                          sample_n45
801                     : ( (selector == -32 && (MODE == 17 || MODE == 19)) ?
                          sample_n36
802                     : ( (selector == -32 &&  MODE == 18)            ?
                          sample_n29
803                     : ( (selector == -31 && (MODE == 15 || MODE == 21)) ?
                          sample_n54
804                     : ( (selector == -31 && (MODE == 16 || MODE == 20)) ?
                          sample_n43
805                     : ( (selector == -31 && (MODE == 17 || MODE == 19)) ?
                          sample_n35
806                     : ( (selector == -31 &&  MODE == 18)            ?
                          sample_n28
807                     : ( (selector == -30 && (MODE == 15 || MODE == 21)) ?
                          sample_n52
808                     : ( (selector == -30 && (MODE == 16 || MODE == 20)) ?
                          sample_n42
809                     : ( (selector == -30 && (MODE == 17 || MODE == 19)) ?
                          sample_n33
810                     : ( (selector == -30 &&  MODE == 18)            ?
                          sample_n27
811                     : ( (selector == -29 && (MODE == 15 || MODE == 21)) ?
                          sample_n50
812                     : ( (selector == -29 && (MODE == 16 || MODE == 20)) ?
                          sample_n40
813                     : ( (selector == -29 && (MODE == 17 || MODE == 19)) ?
                          sample_n32
```

```
814                         : ( (selector == -29 &&  MODE == 18)            ?
                              sample_n26
815                         : ( (selector == -28 && (MODE == 14 || MODE == 22)) ?
                              sample_n63
816                         : ( (selector == -28 && (MODE == 15 || MODE == 21)) ?
                              sample_n48
817                         : ( (selector == -28 && (MODE == 16 || MODE == 20)) ?
                              sample_n39
818                         : ( (selector == -28 && (MODE == 17 || MODE == 19)) ?
                              sample_n31
819                         : ( (selector == -28 &&  MODE == 18)            ?
                              sample_n25
820                         : ( (selector == -27 && (MODE == 14 || MODE == 22)) ?
                              sample_n61
821                         : ( (selector == -27 && (MODE == 15 || MODE == 21)) ?
                              sample_n46
822                         : ( (selector == -27 && (MODE == 16 || MODE == 20)) ?
                              sample_n37
823                         : ( (selector == -27 && (MODE == 17 || MODE == 19)) ?
                              sample_n30
824                         : ( (selector == -27 &&  MODE == 18)            ?
                              sample_n24
825                         : ( (selector == -26 && (MODE == 14 || MODE == 22)) ?
                              sample_n58
826                         : ( (selector == -26 && (MODE == 15 || MODE == 21)) ?
                              sample_n44
827                         : ( (selector == -26 && (MODE == 16 || MODE == 20)) ?
                              sample_n36
828                         : ( (selector == -26 && (MODE == 17 || MODE == 19)) ?
                              sample_n29
829                         : ( (selector == -26 &&  MODE == 18)            ?
                              sample_n23
830                         : ( (selector == -25 && (MODE == 14 || MODE == 22)) ?
                              sample_n56
831                         : ( (selector == -25 && (MODE == 15 || MODE == 21)) ?
                              sample_n42
832                         : ( (selector == -25 && (MODE == 16 || MODE == 20)) ?
                              sample_n34
833                         : ( (selector == -25 && (MODE == 17 || MODE == 19)) ?
                              sample_n27
834                         : ( (selector == -25 &&  MODE == 18)            ?
                              sample_n22
835                         : ( (selector == -24 && (MODE == 14 || MODE == 22)) ?
                              sample_n53
836                         : ( (selector == -24 && (MODE == 15 || MODE == 21)) ?
                              sample_n40
837                         : ( (selector == -24 && (MODE == 16 || MODE == 20)) ?
                              sample_n33
838                         : ( (selector == -24 && (MODE == 17 || MODE == 19)) ?
                              sample_n26
839                         : ( (selector == -24 &&  MODE == 18)            ?
                              sample_n21
840                         : ( (selector == -23 && (MODE == 14 || MODE == 22)) ?
                              sample_n51
841                         : ( (selector == -23 && (MODE == 15 || MODE == 21)) ?
                              sample_n39
842                         : ( (selector == -23 && (MODE == 16 || MODE == 20)) ?
```

```
                    sample_n31
843                 : ( (selector == -23 && (MODE == 17 || MODE == 19)) ?
                    sample_n25
844                 : ( (selector == -23 &&  MODE == 18)            ?
                    sample_n20
845                 : ( (selector == -22 && (MODE == 14 || MODE == 22)) ?
                    sample_n48
846                 : ( (selector == -22 && (MODE == 15 || MODE == 21)) ?
                    sample_n37
847                 : ( (selector == -22 && (MODE == 16 || MODE == 20)) ?
                    sample_n29
848                 : ( (selector == -22 && (MODE == 17 || MODE == 19)) ?
                    sample_n24
849                 : ( (selector == -22 &&  MODE == 18)            ?
                    sample_n19
850                 : ( (selector == -21 && (MODE == 14 || MODE == 22)) ?
                    sample_n46
851                 : ( (selector == -21 && (MODE == 15 || MODE == 21)) ?
                    sample_n35
852                 : ( (selector == -21 && (MODE == 16 || MODE == 20)) ?
                    sample_n28
853                 : ( (selector == -21 && (MODE == 17 || MODE == 19)) ?
                    sample_n22
854                 : ( (selector == -21 &&  MODE == 18)            ?
                    sample_n18
855                 : ( (selector == -20 && (MODE == 13 || MODE == 23)) ?
                    sample_n63
856                 : ( (selector == -20 && (MODE == 14 || MODE == 22)) ?
                    sample_n43
857                 : ( (selector == -20 && (MODE == 15 || MODE == 21)) ?
                    sample_n33
858                 : ( (selector == -20 && (MODE == 16 || MODE == 20)) ?
                    sample_n26
859                 : ( (selector == -20 && (MODE == 17 || MODE == 19)) ?
                    sample_n21
860                 : ( (selector == -20 &&  MODE == 18)            ?
                    sample_n17
861                 : ( (selector == -19 && (MODE == 13 || MODE == 23)) ?
                    sample_n59
862                 : ( (selector == -19 && (MODE == 14 || MODE == 22)) ?
                    sample_n41
863                 : ( (selector == -19 && (MODE == 15 || MODE == 21)) ?
                    sample_n31
864                 : ( (selector == -19 && (MODE == 16 || MODE == 20)) ?
                    sample_n25
865                 : ( (selector == -19 && (MODE == 17 || MODE == 19)) ?
                    sample_n20
866                 : ( (selector == -19 &&  MODE == 18)            ?
                    sample_n16
867                 : ( (selector == -18 && (MODE == 13 || MODE == 23)) ?
                    sample_n56
868                 : ( (selector == -18 && (MODE == 14 || MODE == 22)) ?
                    sample_n38
869                 : ( (selector == -18 && (MODE == 15 || MODE == 21)) ?
                    sample_n29
870                 : ( (selector == -18 && (MODE == 16 || MODE == 20)) ?
                    sample_n23
```

```
871                         : ( (selector == -18 && (MODE == 17 || MODE == 19)) ?
                              sample_n19
872                         : ( (selector == -18 &&  MODE == 18)            ?
                              sample_n15
873                         : ( (selector == -17 && (MODE == 13 || MODE == 23)) ?
                              sample_n52
874                         : ( (selector == -17 && (MODE == 14 || MODE == 22)) ?
                              sample_n36
875                         : ( (selector == -17 && (MODE == 15 || MODE == 21)) ?
                              sample_n27
876                         : ( (selector == -17 && (MODE == 16 || MODE == 20)) ?
                              sample_n22
877                         : ( (selector == -17 && (MODE == 17 || MODE == 19)) ?
                              sample_n17
878                         : ( (selector == -17 &&  MODE == 18)            ?
                              sample_n14
879                         : ( (selector == -16 && (MODE == 13 || MODE == 23)) ?
                              sample_n49
880                         : ( (selector == -16 && (MODE == 14 || MODE == 22)) ?
                              sample_n33
881                         : ( (selector == -16 && (MODE == 15 || MODE == 21)) ?
                              sample_n25
882                         : ( (selector == -16 && (MODE == 16 || MODE == 20)) ?
                              sample_n20
883                         : ( (selector == -16 && (MODE == 17 || MODE == 19)) ?
                              sample_n16
884                         : ( (selector == -16 &&  MODE == 18)            ?
                              sample_n13
885                         : ( (selector == -15 && (MODE == 13 || MODE == 23)) ?
                              sample_n45
886                         : ( (selector == -15 && (MODE == 14 || MODE == 22)) ?
                              sample_n31
887                         : ( (selector == -15 && (MODE == 15 || MODE == 21)) ?
                              sample_n23
888                         : ( (selector == -15 && (MODE == 16 || MODE == 20)) ?
                              sample_n19
889                         : ( (selector == -15 && (MODE == 17 || MODE == 19)) ?
                              sample_n15
890                         : ( (selector == -15 &&  MODE == 18)            ?
                              sample_n12
891                         : ( (selector == -14 && (MODE == 13 || MODE == 23)) ?
                              sample_n42
892                         : ( (selector == -14 && (MODE == 14 || MODE == 22)) ?
                              sample_n29
893                         : ( (selector == -14 && (MODE == 15 || MODE == 21)) ?
                              sample_n22
894                         : ( (selector == -14 && (MODE == 16 || MODE == 20)) ?
                              sample_n17
895                         : ( (selector == -14 && (MODE == 17 || MODE == 19)) ?
                              sample_n14
896                         : ( (selector == -14 &&  MODE == 18)            ?
                              sample_n11
897                         : ( (selector == -13 && (MODE == 13 || MODE == 23)) ?
                              sample_n38
898                         : ( (selector == -13 && (MODE == 14 || MODE == 22)) ?
                              sample_n26
899                         : ( (selector == -13 && (MODE == 15 || MODE == 21)) ?
```

```
                    sample_n20
900                 : ( (selector == -13 && (MODE == 16 || MODE == 20)) ?
                    sample_n16
901                 : ( (selector == -13 && (MODE == 17 || MODE == 19)) ?
                    sample_n13
902                 : ( (selector == -13 &&  MODE == 18)            ?
                    sample_n10
903                 : ( (selector == -12 && (MODE == 12 || MODE == 24)) ?
                    sample_n63
904                 : ( (selector == -12 && (MODE == 13 || MODE == 23)) ?
                    sample_n35
905                 : ( (selector == -12 && (MODE == 14 || MODE == 22)) ?
                    sample_n24
906                 : ( (selector == -12 && (MODE == 15 || MODE == 21)) ?
                    sample_n18
907                 : ( (selector == -12 && (MODE == 16 || MODE == 20)) ?
                    sample_n14
908                 : ( (selector == -12 && (MODE == 17 || MODE == 19)) ?
                    sample_n11
909                 : ( (selector == -12 &&  MODE == 18)            ? sample_n9
910                 : ( (selector == -11 && (MODE == 12 || MODE == 24)) ?
                    sample_n57
911                 : ( (selector == -11 && (MODE == 13 || MODE == 23)) ?
                    sample_n31
912                 : ( (selector == -11 && (MODE == 14 || MODE == 22)) ?
                    sample_n21
913                 : ( (selector == -11 && (MODE == 15 || MODE == 21)) ?
                    sample_n16
914                 : ( (selector == -11 && (MODE == 16 || MODE == 20)) ?
                    sample_n13
915                 : ( (selector == -11 && (MODE == 17 || MODE == 19)) ?
                    sample_n10
916                 : ( (selector == -11 &&  MODE == 18)            ? sample_n8
917                 : ( (selector == -10 && (MODE == 12 || MODE == 24)) ?
                    sample_n50
918                 : ( (selector == -10 && (MODE == 13 || MODE == 23)) ?
                    sample_n27
919                 : ( (selector == -10 && (MODE == 14 || MODE == 22)) ?
                    sample_n19
920                 : ( (selector == -10 && (MODE == 15 || MODE == 21)) ?
                    sample_n14
921                 : ( (selector == -10 && (MODE == 16 || MODE == 20)) ?
                    sample_n11
922                 : ( (selector == -10 && (MODE == 17 || MODE == 19)) ?
                    sample_n9
923                 : ( (selector == -10 &&  MODE == 18)            ? sample_n7
924                 : ( (selector ==  -9 && (MODE == 12 || MODE == 24)) ?
                    sample_n44
925                 : ( (selector ==  -9 && (MODE == 13 || MODE == 23)) ?
                    sample_n24
926                 : ( (selector ==  -9 && (MODE == 14 || MODE == 22)) ?
                    sample_n16
927                 : ( (selector ==  -9 && (MODE == 15 || MODE == 21)) ?
                    sample_n12
928                 : ( (selector ==  -9 && (MODE == 16 || MODE == 20)) ?
                    sample_n10
929                 : ( (selector ==  -9 && (MODE == 17 || MODE == 19)) ?
```

```
                    sample_n8
930                 : ( (selector ==  -9 &&  MODE == 18)            ? sample_n6
931                 : ( (selector ==  -8 && (MODE == 12 || MODE == 24)) ?
                    sample_n37
932                 : ( (selector ==  -8 && (MODE == 13 || MODE == 23)) ?
                    sample_n20
933                 : ( (selector ==  -8 && (MODE == 14 || MODE == 22)) ?
                    sample_n14
934                 : ( (selector ==  -8 && (MODE == 15 || MODE == 21)) ?
                    sample_n10
935                 : ( (selector ==  -8 && (MODE == 16 || MODE == 20)) ?
                    sample_n8
936                 : ( (selector ==  -8 && (MODE == 17 || MODE == 19)) ?
                    sample_n6
937                 : ( (selector ==  -8 &&  MODE == 18)            ? sample_n5
938                 : ( (selector ==  -7 && (MODE == 12 || MODE == 24)) ?
                    sample_n31
939                 : ( (selector ==  -7 && (MODE == 13 || MODE == 23)) ?
                    sample_n17
940                 : ( (selector ==  -7 && (MODE == 14 || MODE == 22)) ?
                    sample_n11
941                 : ( (selector ==  -7 && (MODE == 15 || MODE == 21)) ?
                    sample_n8
942                 : ( (selector ==  -7 && (MODE == 16 || MODE == 20)) ?
                    sample_n7
943                 : ( (selector ==  -7 && (MODE == 17 || MODE == 19)) ?
                    sample_n5
944                 : ( (selector ==  -7 &&  MODE == 18)            ? sample_n4
945                 : ( (selector ==  -6 && (MODE == 11 || MODE == 25)) ?
                    sample_n63
946                 : ( (selector ==  -6 && (MODE == 12 || MODE == 24)) ?
                    sample_n25
947                 : ( (selector ==  -6 && (MODE == 13 || MODE == 23)) ?
                    sample_n13
948                 : ( (selector ==  -6 && (MODE == 14 || MODE == 22)) ?
                    sample_n9
949                 : ( (selector ==  -6 && (MODE == 15 || MODE == 21)) ?
                    sample_n7
950                 : ( (selector ==  -6 && (MODE == 16 || MODE == 20)) ?
                    sample_n5
951                 : ( (selector ==  -6 && (MODE == 17 || MODE == 19)) ?
                    sample_n4
952                 : ( (selector ==  -6 &&  MODE == 18)            ? sample_n3
953                 : ( (selector ==  -5 && (MODE == 11 || MODE == 25)) ?
                    sample_n47
954                 : ( (selector ==  -5 && (MODE == 12 || MODE == 24)) ?
                    sample_n18
955                 : ( (selector ==  -5 && (MODE == 13 || MODE == 23)) ?
                    sample_n10
956                 : ( (selector ==  -5 && (MODE == 14 || MODE == 22)) ?
                    sample_n6
957                 : ( (selector ==  -5 && (MODE == 15 || MODE == 21)) ?
                    sample_n5
958                 : ( (selector ==  -5 && (MODE == 16 || MODE == 20)) ?
                    sample_n4
959                 : ( (selector ==  -5 && (MODE == 17 || MODE == 19)) ?
                    sample_n3
```

```
960                         : ( (selector ==  -5 &&  MODE == 18)            ? sample_n2
961                         : ( (selector ==  -4 && (MODE == 11 || MODE == 25)) ?
                              sample_n31
962                         : ( (selector ==  -4 && (MODE == 12 || MODE == 24)) ?
                              sample_n12
963                         : ( (selector ==  -4 && (MODE == 13 || MODE == 23)) ?
                              sample_n6
964                         : ( (selector ==  -4 && (MODE == 14 || MODE == 22)) ?
                              sample_n4
965                         : ( (selector ==  -4 && (MODE == 15 || MODE == 21)) ?
                              sample_n3
966                         : ( (selector ==  -4 && (MODE == 16 || MODE == 20)) ?
                              sample_n2
967                         : ( (selector ==  -4 && (MODE == 17 || MODE == 19)) ?
                              sample_n1
968                         : ( (selector ==  -4 &&  MODE == 18)            ? sample_n1
969                         : ( (selector ==  -3 && (MODE == 11 || MODE == 25)) ?
                              sample_n15
970                         : ( (selector ==  -3 && (MODE == 12 || MODE == 24)) ?
                              sample_n5
971                         : ( (selector ==  -3 && (MODE == 13 || MODE == 23)) ?
                              sample_n3
972                         : ( (selector ==  -3 && (MODE == 14 || MODE == 22)) ?
                              sample_n1
973                         : ( (selector ==  -3 && (MODE == 15 || MODE == 21)) ?
                              sample_n1
974                         : ( (selector ==  -3 && (MODE == 16 || MODE == 20)) ?
                              sample_n1
975                         : ( (selector ==  -3 && (MODE == 17 || MODE == 19)) ?
                              sample_n0
976                         : ( (selector ==  -3 &&  MODE == 18)            ? sample_n0
977                         : (selector ==  -2 ? sample_NN
978                         : (selector ==  -1 ? sample_0n
979                         : (selector ==   0 ? sample_1n
980                         : (selector ==   1 ? sample_2n
981                         : (selector ==   2 ? sample_3n
982                         : (selector ==   3 ? sample_4n
983                         : (selector ==   4 ? sample_5n
984                         : (selector ==   5 ? sample_6n
985                         : (selector ==   6 ? sample_7n
986                         : (selector ==   7 ? sample_8n
987                         : (selector ==   8 ? sample_9n
988                         : (selector ==   9 ? sample_10n
989                         : (selector ==  10 ? sample_11n
990                         : (selector ==  11 ? sample_12n
991                         : (selector ==  12 ? sample_13n
992                         : (selector ==  13 ? sample_14n
993                         : (selector ==  14 ? sample_15n
994                         : (selector ==  15 ? sample_16n
995                         : (selector ==  16 ? sample_17n
996                         : (selector ==  17 ? sample_18n
997                         : (selector ==  18 ? sample_19n
998                         : (selector ==  19 ? sample_20n
999                         : (selector ==  20 ? sample_21n
1000                        : (selector ==  21 ? sample_22n
1001                        : (selector ==  22 ? sample_23n
1002                        : (selector ==  23 ? sample_24n
```

```
                        : (selector ==  24 ? sample_25n
                        : (selector ==  25 ? sample_26n
                        : (selector ==  26 ? sample_27n
                        : (selector ==  27 ? sample_28n
                        : (selector ==  28 ? sample_29n
                        : (selector ==  29 ? sample_30n
                        : (selector ==  30 ? sample_31n
                        : (selector ==  31 ? sample_32n
                        : (selector ==  32 ? sample_33n
                        : (selector ==  33 ? sample_34n
                        : (selector ==  34 ? sample_35n
                        : (selector ==  35 ? sample_36n
                        : (selector ==  36 ? sample_37n
                        : (selector ==  37 ? sample_38n
                        : (selector ==  38 ? sample_39n
                        : (selector ==  39 ? sample_40n
                        : (selector ==  40 ? sample_41n
                        : (selector ==  41 ? sample_42n
                        : (selector ==  42 ? sample_43n
                        : (selector ==  43 ? sample_44n
                        : (selector ==  44 ? sample_45n
                        : (selector ==  45 ? sample_46n
                        : (selector ==  46 ? sample_47n
                        : (selector ==  47 ? sample_48n
                        : (selector ==  48 ? sample_49n
                        : (selector ==  49 ? sample_50n
                        : (selector ==  50 ? sample_51n
                        : (selector ==  51 ? sample_52n
                        : (selector ==  52 ? sample_53n
                        : (selector ==  53 ? sample_54n
                        : (selector ==  54 ? sample_55n
                        : (selector ==  55 ? sample_56n
                        : (selector ==  56 ? sample_57n
                        : (selector ==  57 ? sample_58n
                        : (selector ==  58 ? sample_59n
                        : (selector ==  59 ? sample_60n
                        : (selector ==  60 ? sample_61n
                        : (selector ==  61 ? sample_62n
                        : (selector ==  62 ? sample_63n
                        : 0 ))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                           ))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                           ))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                           ))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                           ))))))))))))))))))));

    assign ref3 = MODE < 18 ? ( (selector == -32 && (MODE == 15 || MODE == 21)) ?
        sample_54n
                        : ( (selector == -32 && (MODE == 16 || MODE == 20)) ?
                           sample_43n
                        : ( (selector == -32 && (MODE == 17 || MODE == 19)) ?
                           sample_35n
                        : ( (selector == -32 &&  MODE == 18)            ?
                           sample_28n
                        : ( (selector == -31 && (MODE == 15 || MODE == 21)) ?
                           sample_52n
                        : ( (selector == -31 && (MODE == 16 || MODE == 20)) ?
                           sample_42n
```

```
1050                        : ( (selector == -31 && (MODE == 17 || MODE == 19)) ?
                              sample_33n
1051                        : ( (selector == -31 &&  MODE == 18)            ?
                              sample_27n
1052                        : ( (selector == -30 && (MODE == 15 || MODE == 21)) ?
                              sample_50n
1053                        : ( (selector == -30 && (MODE == 16 || MODE == 20)) ?
                              sample_40n
1054                        : ( (selector == -30 && (MODE == 17 || MODE == 19)) ?
                              sample_32n
1055                        : ( (selector == -30 &&  MODE == 18)            ?
                              sample_26n
1056                        : ( (selector == -29 && (MODE == 14 || MODE == 22)) ?
                              sample_63n
1057                        : ( (selector == -29 && (MODE == 15 || MODE == 21)) ?
                              sample_48n
1058                        : ( (selector == -29 && (MODE == 16 || MODE == 20)) ?
                              sample_39n
1059                        : ( (selector == -29 && (MODE == 17 || MODE == 19)) ?
                              sample_31n
1060                        : ( (selector == -29 &&  MODE == 18)            ?
                              sample_25n
1061                        : ( (selector == -28 && (MODE == 14 || MODE == 22)) ?
                              sample_61n
1062                        : ( (selector == -28 && (MODE == 15 || MODE == 21)) ?
                              sample_46n
1063                        : ( (selector == -28 && (MODE == 16 || MODE == 20)) ?
                              sample_37n
1064                        : ( (selector == -28 && (MODE == 17 || MODE == 19)) ?
                              sample_30n
1065                        : ( (selector == -28 &&  MODE == 18)            ?
                              sample_24n
1066                        : ( (selector == -27 && (MODE == 14 || MODE == 22)) ?
                              sample_58n
1067                        : ( (selector == -27 && (MODE == 15 || MODE == 21)) ?
                              sample_44n
1068                        : ( (selector == -27 && (MODE == 16 || MODE == 20)) ?
                              sample_36n
1069                        : ( (selector == -27 && (MODE == 17 || MODE == 19)) ?
                              sample_29n
1070                        : ( (selector == -27 &&  MODE == 18)            ?
                              sample_23n
1071                        : ( (selector == -26 && (MODE == 14 || MODE == 22)) ?
                              sample_56n
1072                        : ( (selector == -26 && (MODE == 15 || MODE == 21)) ?
                              sample_42n
1073                        : ( (selector == -26 && (MODE == 16 || MODE == 20)) ?
                              sample_34n
1074                        : ( (selector == -26 && (MODE == 17 || MODE == 19)) ?
                              sample_27n
1075                        : ( (selector == -26 &&  MODE == 18)            ?
                              sample_22n
1076                        : ( (selector == -25 && (MODE == 14 || MODE == 22)) ?
                              sample_53n
1077                        : ( (selector == -25 && (MODE == 15 || MODE == 21)) ?
                              sample_40n
1078                        : ( (selector == -25 && (MODE == 16 || MODE == 20)) ?
```

```
                    sample_33n
1079                : ( (selector == -25 && (MODE == 17 || MODE == 19)) ?
                    sample_26n
1080                : ( (selector == -25 &&  MODE == 18)            ?
                    sample_21n
1081                : ( (selector == -24 && (MODE == 14 || MODE == 22)) ?
                    sample_51n
1082                : ( (selector == -24 && (MODE == 15 || MODE == 21)) ?
                    sample_39n
1083                : ( (selector == -24 && (MODE == 16 || MODE == 20)) ?
                    sample_31n
1084                : ( (selector == -24 && (MODE == 17 || MODE == 19)) ?
                    sample_25n
1085                : ( (selector == -24 &&  MODE == 18)            ?
                    sample_20n
1086                : ( (selector == -23 && (MODE == 14 || MODE == 22)) ?
                    sample_48n
1087                : ( (selector == -23 && (MODE == 15 || MODE == 21)) ?
                    sample_37n
1088                : ( (selector == -23 && (MODE == 16 || MODE == 20)) ?
                    sample_29n
1089                : ( (selector == -23 && (MODE == 17 || MODE == 19)) ?
                    sample_24n
1090                : ( (selector == -23 &&  MODE == 18)            ?
                    sample_19n
1091                : ( (selector == -22 && (MODE == 14 || MODE == 22)) ?
                    sample_46n
1092                : ( (selector == -22 && (MODE == 15 || MODE == 21)) ?
                    sample_35n
1093                : ( (selector == -22 && (MODE == 16 || MODE == 20)) ?
                    sample_28n
1094                : ( (selector == -22 && (MODE == 17 || MODE == 19)) ?
                    sample_22n
1095                : ( (selector == -22 &&  MODE == 18)            ?
                    sample_18n
1096                : ( (selector == -21 && (MODE == 13 || MODE == 23)) ?
                    sample_63n
1097                : ( (selector == -21 && (MODE == 14 || MODE == 22)) ?
                    sample_43n
1098                : ( (selector == -21 && (MODE == 15 || MODE == 21)) ?
                    sample_33n
1099                : ( (selector == -21 && (MODE == 16 || MODE == 20)) ?
                    sample_26n
1100                : ( (selector == -21 && (MODE == 17 || MODE == 19)) ?
                    sample_21n
1101                : ( (selector == -21 &&  MODE == 18)            ?
                    sample_17n
1102                : ( (selector == -20 && (MODE == 13 || MODE == 23)) ?
                    sample_59n
1103                : ( (selector == -20 && (MODE == 14 || MODE == 22)) ?
                    sample_41n
1104                : ( (selector == -20 && (MODE == 15 || MODE == 21)) ?
                    sample_31n
1105                : ( (selector == -20 && (MODE == 16 || MODE == 20)) ?
                    sample_25n
1106                : ( (selector == -20 && (MODE == 17 || MODE == 19)) ?
                    sample_20n
```

```
1107                        : ( (selector == -20 &&  MODE == 18)            ?
                              sample_16n
1108                        : ( (selector == -19 && (MODE == 13 || MODE == 23)) ?
                              sample_56n
1109                        : ( (selector == -19 && (MODE == 14 || MODE == 22)) ?
                              sample_38n
1110                        : ( (selector == -19 && (MODE == 15 || MODE == 21)) ?
                              sample_29n
1111                        : ( (selector == -19 && (MODE == 16 || MODE == 20)) ?
                              sample_23n
1112                        : ( (selector == -19 && (MODE == 17 || MODE == 19)) ?
                              sample_19n
1113                        : ( (selector == -19 &&  MODE == 18)            ?
                              sample_15n
1114                        : ( (selector == -18 && (MODE == 13 || MODE == 23)) ?
                              sample_52n
1115                        : ( (selector == -18 && (MODE == 14 || MODE == 22)) ?
                              sample_36n
1116                        : ( (selector == -18 && (MODE == 15 || MODE == 21)) ?
                              sample_27n
1117                        : ( (selector == -18 && (MODE == 16 || MODE == 20)) ?
                              sample_22n
1118                        : ( (selector == -18 && (MODE == 17 || MODE == 19)) ?
                              sample_17n
1119                        : ( (selector == -18 &&  MODE == 18)            ?
                              sample_14n
1120                        : ( (selector == -17 && (MODE == 13 || MODE == 23)) ?
                              sample_49n
1121                        : ( (selector == -17 && (MODE == 14 || MODE == 22)) ?
                              sample_33n
1122                        : ( (selector == -17 && (MODE == 15 || MODE == 21)) ?
                              sample_25n
1123                        : ( (selector == -17 && (MODE == 16 || MODE == 20)) ?
                              sample_20n
1124                        : ( (selector == -17 && (MODE == 17 || MODE == 19)) ?
                              sample_16n
1125                        : ( (selector == -17 &&  MODE == 18)            ?
                              sample_13n
1126                        : ( (selector == -16 && (MODE == 13 || MODE == 23)) ?
                              sample_45n
1127                        : ( (selector == -16 && (MODE == 14 || MODE == 22)) ?
                              sample_31n
1128                        : ( (selector == -16 && (MODE == 15 || MODE == 21)) ?
                              sample_23n
1129                        : ( (selector == -16 && (MODE == 16 || MODE == 20)) ?
                              sample_19n
1130                        : ( (selector == -16 && (MODE == 17 || MODE == 19)) ?
                              sample_15n
1131                        : ( (selector == -16 &&  MODE == 18)            ?
                              sample_12n
1132                        : ( (selector == -15 && (MODE == 13 || MODE == 23)) ?
                              sample_42n
1133                        : ( (selector == -15 && (MODE == 14 || MODE == 22)) ?
                              sample_29n
1134                        : ( (selector == -15 && (MODE == 15 || MODE == 21)) ?
                              sample_22n
1135                        : ( (selector == -15 && (MODE == 16 || MODE == 20)) ?
```

```
                        sample_17n
1136                  : ( (selector == -15 && (MODE == 17 || MODE == 19)) ?
                        sample_14n
1137                  : ( (selector == -15 &&  MODE == 18)            ?
                        sample_11n
1138                  : ( (selector == -14 && (MODE == 13 || MODE == 23)) ?
                        sample_38n
1139                  : ( (selector == -14 && (MODE == 14 || MODE == 22)) ?
                        sample_26n
1140                  : ( (selector == -14 && (MODE == 15 || MODE == 21)) ?
                        sample_20n
1141                  : ( (selector == -14 && (MODE == 16 || MODE == 20)) ?
                        sample_16n
1142                  : ( (selector == -14 && (MODE == 17 || MODE == 19)) ?
                        sample_13n
1143                  : ( (selector == -14 &&  MODE == 18)            ?
                        sample_10n
1144                  : ( (selector == -13 && (MODE == 12 || MODE == 24)) ?
                        sample_63n
1145                  : ( (selector == -13 && (MODE == 13 || MODE == 23)) ?
                        sample_35n
1146                  : ( (selector == -13 && (MODE == 14 || MODE == 22)) ?
                        sample_24n
1147                  : ( (selector == -13 && (MODE == 15 || MODE == 21)) ?
                        sample_18n
1148                  : ( (selector == -13 && (MODE == 16 || MODE == 20)) ?
                        sample_14n
1149                  : ( (selector == -13 && (MODE == 17 || MODE == 19)) ?
                        sample_11n
1150                  : ( (selector == -13 &&  MODE == 18)            ? sample_9n
1151                  : ( (selector == -12 && (MODE == 12 || MODE == 24)) ?
                        sample_57n
1152                  : ( (selector == -12 && (MODE == 13 || MODE == 23)) ?
                        sample_31n
1153                  : ( (selector == -12 && (MODE == 14 || MODE == 22)) ?
                        sample_21n
1154                  : ( (selector == -12 && (MODE == 15 || MODE == 21)) ?
                        sample_16n
1155                  : ( (selector == -12 && (MODE == 16 || MODE == 20)) ?
                        sample_13n
1156                  : ( (selector == -12 && (MODE == 17 || MODE == 19)) ?
                        sample_10n
1157                  : ( (selector == -12 &&  MODE == 18)            ? sample_8n
1158                  : ( (selector == -11 && (MODE == 12 || MODE == 24)) ?
                        sample_50n
1159                  : ( (selector == -11 && (MODE == 13 || MODE == 23)) ?
                        sample_27n
1160                  : ( (selector == -11 && (MODE == 14 || MODE == 22)) ?
                        sample_19n
1161                  : ( (selector == -11 && (MODE == 15 || MODE == 21)) ?
                        sample_14n
1162                  : ( (selector == -11 && (MODE == 16 || MODE == 20)) ?
                        sample_11n
1163                  : ( (selector == -11 && (MODE == 17 || MODE == 19)) ?
                        sample_9n
1164                  : ( (selector == -11 &&  MODE == 18)            ? sample_7n
1165                  : ( (selector == -10 && (MODE == 12 || MODE == 24)) ?
```

```
                    sample_44n
1166                : ( (selector == -10 && (MODE == 13 || MODE == 23)) ?
                    sample_24n
1167                : ( (selector == -10 && (MODE == 14 || MODE == 22)) ?
                    sample_16n
1168                : ( (selector == -10 && (MODE == 15 || MODE == 21)) ?
                    sample_12n
1169                : ( (selector == -10 && (MODE == 16 || MODE == 20)) ?
                    sample_10n
1170                : ( (selector == -10 && (MODE == 17 || MODE == 19)) ?
                    sample_8n
1171                : ( (selector == -10 &&  MODE == 18)            ? sample_6n
1172                : ( (selector ==  -9 && (MODE == 12 || MODE == 24)) ?
                    sample_37n
1173                : ( (selector ==  -9 && (MODE == 13 || MODE == 23)) ?
                    sample_20n
1174                : ( (selector ==  -9 && (MODE == 14 || MODE == 22)) ?
                    sample_14n
1175                : ( (selector ==  -9 && (MODE == 15 || MODE == 21)) ?
                    sample_10n
1176                : ( (selector ==  -9 && (MODE == 16 || MODE == 20)) ?
                    sample_8n
1177                : ( (selector ==  -9 && (MODE == 17 || MODE == 19)) ?
                    sample_6n
1178                : ( (selector ==  -9 &&  MODE == 18)            ? sample_5n
1179                : ( (selector ==  -8 && (MODE == 12 || MODE == 24)) ?
                    sample_31n
1180                : ( (selector ==  -8 && (MODE == 13 || MODE == 23)) ?
                    sample_17n
1181                : ( (selector ==  -8 && (MODE == 14 || MODE == 22)) ?
                    sample_11n
1182                : ( (selector ==  -8 && (MODE == 15 || MODE == 21)) ?
                    sample_8n
1183                : ( (selector ==  -8 && (MODE == 16 || MODE == 20)) ?
                    sample_7n
1184                : ( (selector ==  -8 && (MODE == 17 || MODE == 19)) ?
                    sample_5n
1185                : ( (selector ==  -8 &&  MODE == 18)            ? sample_4n
1186                : ( (selector ==  -7 && (MODE == 11 || MODE == 25)) ?
                    sample_63n
1187                : ( (selector ==  -7 && (MODE == 12 || MODE == 24)) ?
                    sample_25n
1188                : ( (selector ==  -7 && (MODE == 13 || MODE == 23)) ?
                    sample_13n
1189                : ( (selector ==  -7 && (MODE == 14 || MODE == 22)) ?
                    sample_9n
1190                : ( (selector ==  -7 && (MODE == 15 || MODE == 21)) ?
                    sample_7n
1191                : ( (selector ==  -7 && (MODE == 16 || MODE == 20)) ?
                    sample_5n
1192                : ( (selector ==  -7 && (MODE == 17 || MODE == 19)) ?
                    sample_4n
1193                : ( (selector ==  -7 &&  MODE == 18)            ? sample_3n
1194                : ( (selector ==  -6 && (MODE == 11 || MODE == 25)) ?
                    sample_47n
1195                : ( (selector ==  -6 && (MODE == 12 || MODE == 24)) ?
                    sample_18n
```

```
1196                        : ( (selector ==  -6 && (MODE == 13 || MODE == 23)) ?
                              sample_10n
1197                        : ( (selector ==  -6 && (MODE == 14 || MODE == 22)) ?
                              sample_6n
1198                        : ( (selector ==  -6 && (MODE == 15 || MODE == 21)) ?
                              sample_5n
1199                        : ( (selector ==  -6 && (MODE == 16 || MODE == 20)) ?
                              sample_4n
1200                        : ( (selector ==  -6 && (MODE == 17 || MODE == 19)) ?
                              sample_3n
1201                        : ( (selector ==  -6 &&  MODE == 18)            ? sample_2n
1202                        : ( (selector ==  -5 && (MODE == 11 || MODE == 25)) ?
                              sample_31n
1203                        : ( (selector ==  -5 && (MODE == 12 || MODE == 24)) ?
                              sample_12n
1204                        : ( (selector ==  -5 && (MODE == 13 || MODE == 23)) ?
                              sample_6n
1205                        : ( (selector ==  -5 && (MODE == 14 || MODE == 22)) ?
                              sample_4n
1206                        : ( (selector ==  -5 && (MODE == 15 || MODE == 21)) ?
                              sample_3n
1207                        : ( (selector ==  -5 && (MODE == 16 || MODE == 20)) ?
                              sample_2n
1208                        : ( (selector ==  -5 && (MODE == 17 || MODE == 19)) ?
                              sample_1n
1209                        : ( (selector ==  -5 &&  MODE == 18)            ? sample_1n
1210                        : ( (selector ==  -4 && (MODE == 11 || MODE == 25)) ?
                              sample_15n
1211                        : ( (selector ==  -4 && (MODE == 12 || MODE == 24)) ?
                              sample_5n
1212                        : ( (selector ==  -4 && (MODE == 13 || MODE == 23)) ?
                              sample_3n
1213                        : ( (selector ==  -4 && (MODE == 14 || MODE == 22)) ?
                              sample_1n
1214                        : ( (selector ==  -4 && (MODE == 15 || MODE == 21)) ?
                              sample_1n
1215                        : ( (selector ==  -4 && (MODE == 16 || MODE == 20)) ?
                              sample_1n
1216                        : ( (selector ==  -4 && (MODE == 17 || MODE == 19)) ?
                              sample_0n
1217                        : ( (selector ==  -4 &&  MODE == 18)            ? sample_0n
1218                        : (selector ==  -3 ? sample_NN
1219                        : (selector ==  -2 ? sample_n0
1220                        : (selector ==  -1 ? sample_n1
1221                        : (selector ==   0 ? sample_n2
1222                        : (selector ==   1 ? sample_n3
1223                        : (selector ==   2 ? sample_n4
1224                        : (selector ==   3 ? sample_n5
1225                        : (selector ==   4 ? sample_n6
1226                        : (selector ==   5 ? sample_n7
1227                        : (selector ==   6 ? sample_n8
1228                        : (selector ==   7 ? sample_n9
1229                        : (selector ==   8 ? sample_n10
1230                        : (selector ==   9 ? sample_n11
1231                        : (selector ==  10 ? sample_n12
1232                        : (selector ==  11 ? sample_n13
1233                        : (selector ==  12 ? sample_n14
```

```
                              : (selector ==  13 ? sample_n15
                              : (selector ==  14 ? sample_n16
                              : (selector ==  15 ? sample_n17
                              : (selector ==  16 ? sample_n18
                              : (selector ==  17 ? sample_n19
                              : (selector ==  18 ? sample_n20
                              : (selector ==  19 ? sample_n21
                              : (selector ==  20 ? sample_n22
                              : (selector ==  21 ? sample_n23
                              : (selector ==  22 ? sample_n24
                              : (selector ==  23 ? sample_n25
                              : (selector ==  24 ? sample_n26
                              : (selector ==  25 ? sample_n27
                              : (selector ==  26 ? sample_n28
                              : (selector ==  27 ? sample_n29
                              : (selector ==  28 ? sample_n30
                              : (selector ==  29 ? sample_n31
                              : (selector ==  30 ? sample_n32
                              : (selector ==  31 ? sample_n33
                              : (selector ==  32 ? sample_n34
                              : (selector ==  33 ? sample_n35
                              : (selector ==  34 ? sample_n36
                              : (selector ==  35 ? sample_n37
                              : (selector ==  36 ? sample_n38
                              : (selector ==  37 ? sample_n39
                              : (selector ==  38 ? sample_n40
                              : (selector ==  39 ? sample_n41
                              : (selector ==  40 ? sample_n42
                              : (selector ==  41 ? sample_n43
                              : (selector ==  42 ? sample_n44
                              : (selector ==  43 ? sample_n45
                              : (selector ==  44 ? sample_n46
                              : (selector ==  45 ? sample_n47
                              : (selector ==  46 ? sample_n48
                              : (selector ==  47 ? sample_n49
                              : (selector ==  48 ? sample_n50
                              : (selector ==  49 ? sample_n51
                              : (selector ==  50 ? sample_n52
                              : (selector ==  51 ? sample_n53
                              : (selector ==  52 ? sample_n54
                              : (selector ==  53 ? sample_n55
                              : (selector ==  54 ? sample_n56
                              : (selector ==  55 ? sample_n57
                              : (selector ==  56 ? sample_n58
                              : (selector ==  57 ? sample_n59
                              : (selector ==  58 ? sample_n60
                              : (selector ==  59 ? sample_n61
                              : (selector ==  60 ? sample_n62
                              : (selector ==  61 ? sample_n63
                              : 0 )))))))))))))))))))))))))))))))))))))))))))))))))))))))
                                 )))))))))))))))))))))))))))))))))))))))))))))))))))))))
                                 )))))))))))))))))))))))))))))))))))))))))))))))))))))))
                                 )))))))))))))))))))))))))))))))))))))))))))))))))))))))
                                 )))))))))))))))))))
                              : ( (selector == -32 && (MODE == 15 || MODE == 21)) ?
                                 sample_n54
                              : ( (selector == -32 && (MODE == 16 || MODE == 20)) ?
```

```
                    sample_n43
1286                : ( (selector == -32 && (MODE == 17 || MODE == 19)) ?
                    sample_n35
1287                : ( (selector == -32 &&  MODE == 18)            ?
                    sample_n28
1288                : ( (selector == -31 && (MODE == 15 || MODE == 21)) ?
                    sample_n52
1289                : ( (selector == -31 && (MODE == 16 || MODE == 20)) ?
                    sample_n42
1290                : ( (selector == -31 && (MODE == 17 || MODE == 19)) ?
                    sample_n33
1291                : ( (selector == -31 &&  MODE == 18)            ?
                    sample_n27
1292                : ( (selector == -30 && (MODE == 15 || MODE == 21)) ?
                    sample_n50
1293                : ( (selector == -30 && (MODE == 16 || MODE == 20)) ?
                    sample_n40
1294                : ( (selector == -30 && (MODE == 17 || MODE == 19)) ?
                    sample_n32
1295                : ( (selector == -30 &&  MODE == 18)            ?
                    sample_n26
1296                : ( (selector == -29 && (MODE == 14 || MODE == 22)) ?
                    sample_n63
1297                : ( (selector == -29 && (MODE == 15 || MODE == 21)) ?
                    sample_n48
1298                : ( (selector == -29 && (MODE == 16 || MODE == 20)) ?
                    sample_n39
1299                : ( (selector == -29 && (MODE == 17 || MODE == 19)) ?
                    sample_n31
1300                : ( (selector == -29 &&  MODE == 18)            ?
                    sample_n25
1301                : ( (selector == -28 && (MODE == 14 || MODE == 22)) ?
                    sample_n61
1302                : ( (selector == -28 && (MODE == 15 || MODE == 21)) ?
                    sample_n46
1303                : ( (selector == -28 && (MODE == 16 || MODE == 20)) ?
                    sample_n37
1304                : ( (selector == -28 && (MODE == 17 || MODE == 19)) ?
                    sample_n30
1305                : ( (selector == -28 &&  MODE == 18)            ?
                    sample_n24
1306                : ( (selector == -27 && (MODE == 14 || MODE == 22)) ?
                    sample_n58
1307                : ( (selector == -27 && (MODE == 15 || MODE == 21)) ?
                    sample_n44
1308                : ( (selector == -27 && (MODE == 16 || MODE == 20)) ?
                    sample_n36
1309                : ( (selector == -27 && (MODE == 17 || MODE == 19)) ?
                    sample_n29
1310                : ( (selector == -27 &&  MODE == 18)            ?
                    sample_n23
1311                : ( (selector == -26 && (MODE == 14 || MODE == 22)) ?
                    sample_n56
1312                : ( (selector == -26 && (MODE == 15 || MODE == 21)) ?
                    sample_n42
1313                : ( (selector == -26 && (MODE == 16 || MODE == 20)) ?
                    sample_n34
```

```
1314                        : ( (selector == -26 && (MODE == 17 || MODE == 19)) ?
                              sample_n27
1315                        : ( (selector == -26 &&  MODE == 18)            ?
                              sample_n22
1316                        : ( (selector == -25 && (MODE == 14 || MODE == 22)) ?
                              sample_n53
1317                        : ( (selector == -25 && (MODE == 15 || MODE == 21)) ?
                              sample_n40
1318                        : ( (selector == -25 && (MODE == 16 || MODE == 20)) ?
                              sample_n33
1319                        : ( (selector == -25 && (MODE == 17 || MODE == 19)) ?
                              sample_n26
1320                        : ( (selector == -25 &&  MODE == 18)            ?
                              sample_n21
1321                        : ( (selector == -24 && (MODE == 14 || MODE == 22)) ?
                              sample_n51
1322                        : ( (selector == -24 && (MODE == 15 || MODE == 21)) ?
                              sample_n39
1323                        : ( (selector == -24 && (MODE == 16 || MODE == 20)) ?
                              sample_n31
1324                        : ( (selector == -24 && (MODE == 17 || MODE == 19)) ?
                              sample_n25
1325                        : ( (selector == -24 &&  MODE == 18)            ?
                              sample_n20
1326                        : ( (selector == -23 && (MODE == 14 || MODE == 22)) ?
                              sample_n48
1327                        : ( (selector == -23 && (MODE == 15 || MODE == 21)) ?
                              sample_n37
1328                        : ( (selector == -23 && (MODE == 16 || MODE == 20)) ?
                              sample_n29
1329                        : ( (selector == -23 && (MODE == 17 || MODE == 19)) ?
                              sample_n24
1330                        : ( (selector == -23 &&  MODE == 18)            ?
                              sample_n19
1331                        : ( (selector == -22 && (MODE == 14 || MODE == 22)) ?
                              sample_n46
1332                        : ( (selector == -22 && (MODE == 15 || MODE == 21)) ?
                              sample_n35
1333                        : ( (selector == -22 && (MODE == 16 || MODE == 20)) ?
                              sample_n28
1334                        : ( (selector == -22 && (MODE == 17 || MODE == 19)) ?
                              sample_n22
1335                        : ( (selector == -22 &&  MODE == 18)            ?
                              sample_n18
1336                        : ( (selector == -21 && (MODE == 13 || MODE == 23)) ?
                              sample_n63
1337                        : ( (selector == -21 && (MODE == 14 || MODE == 22)) ?
                              sample_n43
1338                        : ( (selector == -21 && (MODE == 15 || MODE == 21)) ?
                              sample_n33
1339                        : ( (selector == -21 && (MODE == 16 || MODE == 20)) ?
                              sample_n26
1340                        : ( (selector == -21 && (MODE == 17 || MODE == 19)) ?
                              sample_n21
1341                        : ( (selector == -21 &&  MODE == 18)            ?
                              sample_n17
1342                        : ( (selector == -20 && (MODE == 13 || MODE == 23)) ?
```

```
                    sample_n59
1343                : ( (selector == -20 && (MODE == 14 || MODE == 22)) ?
                    sample_n41
1344                : ( (selector == -20 && (MODE == 15 || MODE == 21)) ?
                    sample_n31
1345                : ( (selector == -20 && (MODE == 16 || MODE == 20)) ?
                    sample_n25
1346                : ( (selector == -20 && (MODE == 17 || MODE == 19)) ?
                    sample_n20
1347                : ( (selector == -20 &&  MODE == 18)            ?
                    sample_n16
1348                : ( (selector == -19 && (MODE == 13 || MODE == 23)) ?
                    sample_n56
1349                : ( (selector == -19 && (MODE == 14 || MODE == 22)) ?
                    sample_n38
1350                : ( (selector == -19 && (MODE == 15 || MODE == 21)) ?
                    sample_n29
1351                : ( (selector == -19 && (MODE == 16 || MODE == 20)) ?
                    sample_n23
1352                : ( (selector == -19 && (MODE == 17 || MODE == 19)) ?
                    sample_n19
1353                : ( (selector == -19 &&  MODE == 18)            ?
                    sample_n15
1354                : ( (selector == -18 && (MODE == 13 || MODE == 23)) ?
                    sample_n52
1355                : ( (selector == -18 && (MODE == 14 || MODE == 22)) ?
                    sample_n36
1356                : ( (selector == -18 && (MODE == 15 || MODE == 21)) ?
                    sample_n27
1357                : ( (selector == -18 && (MODE == 16 || MODE == 20)) ?
                    sample_n22
1358                : ( (selector == -18 && (MODE == 17 || MODE == 19)) ?
                    sample_n17
1359                : ( (selector == -18 &&  MODE == 18)            ?
                    sample_n14
1360                : ( (selector == -17 && (MODE == 13 || MODE == 23)) ?
                    sample_n49
1361                : ( (selector == -17 && (MODE == 14 || MODE == 22)) ?
                    sample_n33
1362                : ( (selector == -17 && (MODE == 15 || MODE == 21)) ?
                    sample_n25
1363                : ( (selector == -17 && (MODE == 16 || MODE == 20)) ?
                    sample_n20
1364                : ( (selector == -17 && (MODE == 17 || MODE == 19)) ?
                    sample_n16
1365                : ( (selector == -17 &&  MODE == 18)            ?
                    sample_n13
1366                : ( (selector == -16 && (MODE == 13 || MODE == 23)) ?
                    sample_n45
1367                : ( (selector == -16 && (MODE == 14 || MODE == 22)) ?
                    sample_n31
1368                : ( (selector == -16 && (MODE == 15 || MODE == 21)) ?
                    sample_n23
1369                : ( (selector == -16 && (MODE == 16 || MODE == 20)) ?
                    sample_n19
1370                : ( (selector == -16 && (MODE == 17 || MODE == 19)) ?
                    sample_n15
```

```
1371                        : ( (selector == -16 &&  MODE == 18)            ?
                              sample_n12
1372                        : ( (selector == -15 && (MODE == 13 || MODE == 23)) ?
                              sample_n42
1373                        : ( (selector == -15 && (MODE == 14 || MODE == 22)) ?
                              sample_n29
1374                        : ( (selector == -15 && (MODE == 15 || MODE == 21)) ?
                              sample_n22
1375                        : ( (selector == -15 && (MODE == 16 || MODE == 20)) ?
                              sample_n17
1376                        : ( (selector == -15 && (MODE == 17 || MODE == 19)) ?
                              sample_n14
1377                        : ( (selector == -15 &&  MODE == 18)            ?
                              sample_n11
1378                        : ( (selector == -14 && (MODE == 13 || MODE == 23)) ?
                              sample_n38
1379                        : ( (selector == -14 && (MODE == 14 || MODE == 22)) ?
                              sample_n26
1380                        : ( (selector == -14 && (MODE == 15 || MODE == 21)) ?
                              sample_n20
1381                        : ( (selector == -14 && (MODE == 16 || MODE == 20)) ?
                              sample_n16
1382                        : ( (selector == -14 && (MODE == 17 || MODE == 19)) ?
                              sample_n13
1383                        : ( (selector == -14 &&  MODE == 18)            ?
                              sample_n10
1384                        : ( (selector == -13 && (MODE == 12 || MODE == 24)) ?
                              sample_n63
1385                        : ( (selector == -13 && (MODE == 13 || MODE == 23)) ?
                              sample_n35
1386                        : ( (selector == -13 && (MODE == 14 || MODE == 22)) ?
                              sample_n24
1387                        : ( (selector == -13 && (MODE == 15 || MODE == 21)) ?
                              sample_n18
1388                        : ( (selector == -13 && (MODE == 16 || MODE == 20)) ?
                              sample_n14
1389                        : ( (selector == -13 && (MODE == 17 || MODE == 19)) ?
                              sample_n11
1390                        : ( (selector == -13 &&  MODE == 18)            ? sample_n9
1391                        : ( (selector == -12 && (MODE == 12 || MODE == 24)) ?
                              sample_n57
1392                        : ( (selector == -12 && (MODE == 13 || MODE == 23)) ?
                              sample_n31
1393                        : ( (selector == -12 && (MODE == 14 || MODE == 22)) ?
                              sample_n21
1394                        : ( (selector == -12 && (MODE == 15 || MODE == 21)) ?
                              sample_n16
1395                        : ( (selector == -12 && (MODE == 16 || MODE == 20)) ?
                              sample_n13
1396                        : ( (selector == -12 && (MODE == 17 || MODE == 19)) ?
                              sample_n10
1397                        : ( (selector == -12 &&  MODE == 18)            ? sample_n8
1398                        : ( (selector == -11 && (MODE == 12 || MODE == 24)) ?
                              sample_n50
1399                        : ( (selector == -11 && (MODE == 13 || MODE == 23)) ?
                              sample_n27
1400                        : ( (selector == -11 && (MODE == 14 || MODE == 22)) ?
```

```
                    sample_n19
1401              : ( (selector == -11 && (MODE == 15 || MODE == 21)) ?
                    sample_n14
1402              : ( (selector == -11 && (MODE == 16 || MODE == 20)) ?
                    sample_n11
1403              : ( (selector == -11 && (MODE == 17 || MODE == 19)) ?
                    sample_n9
1404              : ( (selector == -11 &&  MODE == 18)            ? sample_n7
1405              : ( (selector == -10 && (MODE == 12 || MODE == 24)) ?
                    sample_n44
1406              : ( (selector == -10 && (MODE == 13 || MODE == 23)) ?
                    sample_n24
1407              : ( (selector == -10 && (MODE == 14 || MODE == 22)) ?
                    sample_n16
1408              : ( (selector == -10 && (MODE == 15 || MODE == 21)) ?
                    sample_n12
1409              : ( (selector == -10 && (MODE == 16 || MODE == 20)) ?
                    sample_n10
1410              : ( (selector == -10 && (MODE == 17 || MODE == 19)) ?
                    sample_n8
1411              : ( (selector == -10 &&  MODE == 18)            ? sample_n6
1412              : ( (selector ==  -9 && (MODE == 12 || MODE == 24)) ?
                    sample_n37
1413              : ( (selector ==  -9 && (MODE == 13 || MODE == 23)) ?
                    sample_n20
1414              : ( (selector ==  -9 && (MODE == 14 || MODE == 22)) ?
                    sample_n14
1415              : ( (selector ==  -9 && (MODE == 15 || MODE == 21)) ?
                    sample_n10
1416              : ( (selector ==  -9 && (MODE == 16 || MODE == 20)) ?
                    sample_n8
1417              : ( (selector ==  -9 && (MODE == 17 || MODE == 19)) ?
                    sample_n6
1418              : ( (selector ==  -9 &&  MODE == 18)            ? sample_n5
1419              : ( (selector ==  -8 && (MODE == 12 || MODE == 24)) ?
                    sample_n31
1420              : ( (selector ==  -8 && (MODE == 13 || MODE == 23)) ?
                    sample_n17
1421              : ( (selector ==  -8 && (MODE == 14 || MODE == 22)) ?
                    sample_n11
1422              : ( (selector ==  -8 && (MODE == 15 || MODE == 21)) ?
                    sample_n8
1423              : ( (selector ==  -8 && (MODE == 16 || MODE == 20)) ?
                    sample_n7
1424              : ( (selector ==  -8 && (MODE == 17 || MODE == 19)) ?
                    sample_n5
1425              : ( (selector ==  -8 &&  MODE == 18)            ? sample_n4
1426              : ( (selector ==  -7 && (MODE == 11 || MODE == 25)) ?
                    sample_n63
1427              : ( (selector ==  -7 && (MODE == 12 || MODE == 24)) ?
                    sample_n25
1428              : ( (selector ==  -7 && (MODE == 13 || MODE == 23)) ?
                    sample_n13
1429              : ( (selector ==  -7 && (MODE == 14 || MODE == 22)) ?
                    sample_n9
1430              : ( (selector ==  -7 && (MODE == 15 || MODE == 21)) ?
                    sample_n7
```

```
1431                    : ( (selector ==  -7 && (MODE == 16 || MODE == 20)) ?
                          sample_n5
1432                    : ( (selector ==  -7 && (MODE == 17 || MODE == 19)) ?
                          sample_n4
1433                    : ( (selector ==  -7 &&  MODE == 18)            ? sample_n3
1434                    : ( (selector ==  -6 && (MODE == 11 || MODE == 25)) ?
                          sample_n47
1435                    : ( (selector ==  -6 && (MODE == 12 || MODE == 24)) ?
                          sample_n18
1436                    : ( (selector ==  -6 && (MODE == 13 || MODE == 23)) ?
                          sample_n10
1437                    : ( (selector ==  -6 && (MODE == 14 || MODE == 22)) ?
                          sample_n6
1438                    : ( (selector ==  -6 && (MODE == 15 || MODE == 21)) ?
                          sample_n5
1439                    : ( (selector ==  -6 && (MODE == 16 || MODE == 20)) ?
                          sample_n4
1440                    : ( (selector ==  -6 && (MODE == 17 || MODE == 19)) ?
                          sample_n3
1441                    : ( (selector ==  -6 &&  MODE == 18)            ? sample_n2
1442                    : ( (selector ==  -5 && (MODE == 11 || MODE == 25)) ?
                          sample_n31
1443                    : ( (selector ==  -5 && (MODE == 12 || MODE == 24)) ?
                          sample_n12
1444                    : ( (selector ==  -5 && (MODE == 13 || MODE == 23)) ?
                          sample_n6
1445                    : ( (selector ==  -5 && (MODE == 14 || MODE == 22)) ?
                          sample_n4
1446                    : ( (selector ==  -5 && (MODE == 15 || MODE == 21)) ?
                          sample_n3
1447                    : ( (selector ==  -5 && (MODE == 16 || MODE == 20)) ?
                          sample_n2
1448                    : ( (selector ==  -5 && (MODE == 17 || MODE == 19)) ?
                          sample_n1
1449                    : ( (selector ==  -5 &&  MODE == 18)            ? sample_n1
1450                    : ( (selector ==  -4 && (MODE == 11 || MODE == 25)) ?
                          sample_n15
1451                    : ( (selector ==  -4 && (MODE == 12 || MODE == 24)) ?
                          sample_n5
1452                    : ( (selector ==  -4 && (MODE == 13 || MODE == 23)) ?
                          sample_n3
1453                    : ( (selector ==  -4 && (MODE == 14 || MODE == 22)) ?
                          sample_n1
1454                    : ( (selector ==  -4 && (MODE == 15 || MODE == 21)) ?
                          sample_n1
1455                    : ( (selector ==  -4 && (MODE == 16 || MODE == 20)) ?
                          sample_n1
1456                    : ( (selector ==  -4 && (MODE == 17 || MODE == 19)) ?
                          sample_n0
1457                    : ( (selector ==  -4 &&  MODE == 18)            ? sample_n0
1458                    : (selector ==  -3 ? sample_NN
1459                    : (selector ==  -2 ? sample_0n
1460                    : (selector ==  -1 ? sample_1n
1461                    : (selector ==   0 ? sample_2n
1462                    : (selector ==   1 ? sample_3n
1463                    : (selector ==   2 ? sample_4n
1464                    : (selector ==   3 ? sample_5n
```

```
                        : (selector ==   4 ? sample_6n
                        : (selector ==   5 ? sample_7n
                        : (selector ==   6 ? sample_8n
                        : (selector ==   7 ? sample_9n
                        : (selector ==   8 ? sample_10n
                        : (selector ==   9 ? sample_11n
                        : (selector ==  10 ? sample_12n
                        : (selector ==  11 ? sample_13n
                        : (selector ==  12 ? sample_14n
                        : (selector ==  13 ? sample_15n
                        : (selector ==  14 ? sample_16n
                        : (selector ==  15 ? sample_17n
                        : (selector ==  16 ? sample_18n
                        : (selector ==  17 ? sample_19n
                        : (selector ==  18 ? sample_20n
                        : (selector ==  19 ? sample_21n
                        : (selector ==  20 ? sample_22n
                        : (selector ==  21 ? sample_23n
                        : (selector ==  22 ? sample_24n
                        : (selector ==  23 ? sample_25n
                        : (selector ==  24 ? sample_26n
                        : (selector ==  25 ? sample_27n
                        : (selector ==  26 ? sample_28n
                        : (selector ==  27 ? sample_29n
                        : (selector ==  28 ? sample_30n
                        : (selector ==  29 ? sample_31n
                        : (selector ==  30 ? sample_32n
                        : (selector ==  31 ? sample_33n
                        : (selector ==  32 ? sample_34n
                        : (selector ==  33 ? sample_35n
                        : (selector ==  34 ? sample_36n
                        : (selector ==  35 ? sample_37n
                        : (selector ==  36 ? sample_38n
                        : (selector ==  37 ? sample_39n
                        : (selector ==  38 ? sample_40n
                        : (selector ==  39 ? sample_41n
                        : (selector ==  40 ? sample_42n
                        : (selector ==  41 ? sample_43n
                        : (selector ==  42 ? sample_44n
                        : (selector ==  43 ? sample_45n
                        : (selector ==  44 ? sample_46n
                        : (selector ==  45 ? sample_47n
                        : (selector ==  46 ? sample_48n
                        : (selector ==  47 ? sample_49n
                        : (selector ==  48 ? sample_50n
                        : (selector ==  49 ? sample_51n
                        : (selector ==  50 ? sample_52n
                        : (selector ==  51 ? sample_53n
                        : (selector ==  52 ? sample_54n
                        : (selector ==  53 ? sample_55n
                        : (selector ==  54 ? sample_56n
                        : (selector ==  55 ? sample_57n
                        : (selector ==  56 ? sample_58n
                        : (selector ==  57 ? sample_59n
                        : (selector ==  58 ? sample_60n
                        : (selector ==  59 ? sample_61n
                        : (selector ==  60 ? sample_62n
```

```
1522                        : (selector ==  61 ? sample_63n
1523                        : 0 ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                               ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                               ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                               ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                               ))))))))))))))))));
1524
1525     assign ref4 = MODE < 18 ? ( (selector == -32 && (MODE == 15 || MODE == 21)) ?
              sample_52n
1526                        : ( (selector == -32 && (MODE == 16 || MODE == 20)) ?
                               sample_42n
1527                        : ( (selector == -32 && (MODE == 17 || MODE == 19)) ?
                               sample_33n
1528                        : ( (selector == -32 &&  MODE == 18)            ?
                               sample_27n
1529                        : ( (selector == -31 && (MODE == 15 || MODE == 21)) ?
                               sample_50n
1530                        : ( (selector == -31 && (MODE == 16 || MODE == 20)) ?
                               sample_40n
1531                        : ( (selector == -31 && (MODE == 17 || MODE == 19)) ?
                               sample_32n
1532                        : ( (selector == -31 &&  MODE == 18)            ?
                               sample_26n
1533                        : ( (selector == -30 && (MODE == 14 || MODE == 22)) ?
                               sample_63n
1534                        : ( (selector == -30 && (MODE == 15 || MODE == 21)) ?
                               sample_48n
1535                        : ( (selector == -30 && (MODE == 16 || MODE == 20)) ?
                               sample_39n
1536                        : ( (selector == -30 && (MODE == 17 || MODE == 19)) ?
                               sample_31n
1537                        : ( (selector == -30 &&  MODE == 18)            ?
                               sample_25n
1538                        : ( (selector == -29 && (MODE == 14 || MODE == 22)) ?
                               sample_61n
1539                        : ( (selector == -29 && (MODE == 15 || MODE == 21)) ?
                               sample_46n
1540                        : ( (selector == -29 && (MODE == 16 || MODE == 20)) ?
                               sample_37n
1541                        : ( (selector == -29 && (MODE == 17 || MODE == 19)) ?
                               sample_30n
1542                        : ( (selector == -29 &&  MODE == 18)            ?
                               sample_24n
1543                        : ( (selector == -28 && (MODE == 14 || MODE == 22)) ?
                               sample_58n
1544                        : ( (selector == -28 && (MODE == 15 || MODE == 21)) ?
                               sample_44n
1545                        : ( (selector == -28 && (MODE == 16 || MODE == 20)) ?
                               sample_36n
1546                        : ( (selector == -28 && (MODE == 17 || MODE == 19)) ?
                               sample_29n
1547                        : ( (selector == -28 &&  MODE == 18)            ?
                               sample_23n
1548                        : ( (selector == -27 && (MODE == 14 || MODE == 22)) ?
                               sample_56n
1549                        : ( (selector == -27 && (MODE == 15 || MODE == 21)) ?
                               sample_42n
```

```
1550                        : ( (selector == -27 && (MODE == 16 || MODE == 20)) ?
                              sample_34n
1551                        : ( (selector == -27 && (MODE == 17 || MODE == 19)) ?
                              sample_27n
1552                        : ( (selector == -27 &&  MODE == 18)            ?
                              sample_22n
1553                        : ( (selector == -26 && (MODE == 14 || MODE == 22)) ?
                              sample_53n
1554                        : ( (selector == -26 && (MODE == 15 || MODE == 21)) ?
                              sample_40n
1555                        : ( (selector == -26 && (MODE == 16 || MODE == 20)) ?
                              sample_33n
1556                        : ( (selector == -26 && (MODE == 17 || MODE == 19)) ?
                              sample_26n
1557                        : ( (selector == -26 &&  MODE == 18)            ?
                              sample_21n
1558                        : ( (selector == -25 && (MODE == 14 || MODE == 22)) ?
                              sample_51n
1559                        : ( (selector == -25 && (MODE == 15 || MODE == 21)) ?
                              sample_39n
1560                        : ( (selector == -25 && (MODE == 16 || MODE == 20)) ?
                              sample_31n
1561                        : ( (selector == -25 && (MODE == 17 || MODE == 19)) ?
                              sample_25n
1562                        : ( (selector == -25 &&  MODE == 18)            ?
                              sample_20n
1563                        : ( (selector == -24 && (MODE == 14 || MODE == 22)) ?
                              sample_48n
1564                        : ( (selector == -24 && (MODE == 15 || MODE == 21)) ?
                              sample_37n
1565                        : ( (selector == -24 && (MODE == 16 || MODE == 20)) ?
                              sample_29n
1566                        : ( (selector == -24 && (MODE == 17 || MODE == 19)) ?
                              sample_24n
1567                        : ( (selector == -24 &&  MODE == 18)            ?
                              sample_19n
1568                        : ( (selector == -23 && (MODE == 14 || MODE == 22)) ?
                              sample_46n
1569                        : ( (selector == -23 && (MODE == 15 || MODE == 21)) ?
                              sample_35n
1570                        : ( (selector == -23 && (MODE == 16 || MODE == 20)) ?
                              sample_28n
1571                        : ( (selector == -23 && (MODE == 17 || MODE == 19)) ?
                              sample_22n
1572                        : ( (selector == -23 &&  MODE == 18)            ?
                              sample_18n
1573                        : ( (selector == -22 && (MODE == 13 || MODE == 23)) ?
                              sample_63n
1574                        : ( (selector == -22 && (MODE == 14 || MODE == 22)) ?
                              sample_43n
1575                        : ( (selector == -22 && (MODE == 15 || MODE == 21)) ?
                              sample_33n
1576                        : ( (selector == -22 && (MODE == 16 || MODE == 20)) ?
                              sample_26n
1577                        : ( (selector == -22 && (MODE == 17 || MODE == 19)) ?
                              sample_21n
1578                        : ( (selector == -22 &&  MODE == 18)            ?
```

```
                    sample_17n
1579                : ( (selector == -21 && (MODE == 13 || MODE == 23)) ?
                    sample_59n
1580                : ( (selector == -21 && (MODE == 14 || MODE == 22)) ?
                    sample_41n
1581                : ( (selector == -21 && (MODE == 15 || MODE == 21)) ?
                    sample_31n
1582                : ( (selector == -21 && (MODE == 16 || MODE == 20)) ?
                    sample_25n
1583                : ( (selector == -21 && (MODE == 17 || MODE == 19)) ?
                    sample_20n
1584                : ( (selector == -21 &&  MODE == 18)            ?
                    sample_16n
1585                : ( (selector == -20 && (MODE == 13 || MODE == 23)) ?
                    sample_56n
1586                : ( (selector == -20 && (MODE == 14 || MODE == 22)) ?
                    sample_38n
1587                : ( (selector == -20 && (MODE == 15 || MODE == 21)) ?
                    sample_29n
1588                : ( (selector == -20 && (MODE == 16 || MODE == 20)) ?
                    sample_23n
1589                : ( (selector == -20 && (MODE == 17 || MODE == 19)) ?
                    sample_19n
1590                : ( (selector == -20 &&  MODE == 18)            ?
                    sample_15n
1591                : ( (selector == -19 && (MODE == 13 || MODE == 23)) ?
                    sample_52n
1592                : ( (selector == -19 && (MODE == 14 || MODE == 22)) ?
                    sample_36n
1593                : ( (selector == -19 && (MODE == 15 || MODE == 21)) ?
                    sample_27n
1594                : ( (selector == -19 && (MODE == 16 || MODE == 20)) ?
                    sample_22n
1595                : ( (selector == -19 && (MODE == 17 || MODE == 19)) ?
                    sample_17n
1596                : ( (selector == -19 &&  MODE == 18)            ?
                    sample_14n
1597                : ( (selector == -18 && (MODE == 13 || MODE == 23)) ?
                    sample_49n
1598                : ( (selector == -18 && (MODE == 14 || MODE == 22)) ?
                    sample_33n
1599                : ( (selector == -18 && (MODE == 15 || MODE == 21)) ?
                    sample_25n
1600                : ( (selector == -18 && (MODE == 16 || MODE == 20)) ?
                    sample_20n
1601                : ( (selector == -18 && (MODE == 17 || MODE == 19)) ?
                    sample_16n
1602                : ( (selector == -18 &&  MODE == 18)            ?
                    sample_13n
1603                : ( (selector == -17 && (MODE == 13 || MODE == 23)) ?
                    sample_45n
1604                : ( (selector == -17 && (MODE == 14 || MODE == 22)) ?
                    sample_31n
1605                : ( (selector == -17 && (MODE == 15 || MODE == 21)) ?
                    sample_23n
1606                : ( (selector == -17 && (MODE == 16 || MODE == 20)) ?
                    sample_19n
```

```
1607                        : ( (selector == -17 && (MODE == 17 || MODE == 19)) ?
                              sample_15n
1608                        : ( (selector == -17 &&  MODE == 18)            ?
                              sample_12n
1609                        : ( (selector == -16 && (MODE == 13 || MODE == 23)) ?
                              sample_42n
1610                        : ( (selector == -16 && (MODE == 14 || MODE == 22)) ?
                              sample_29n
1611                        : ( (selector == -16 && (MODE == 15 || MODE == 21)) ?
                              sample_22n
1612                        : ( (selector == -16 && (MODE == 16 || MODE == 20)) ?
                              sample_17n
1613                        : ( (selector == -16 && (MODE == 17 || MODE == 19)) ?
                              sample_14n
1614                        : ( (selector == -16 &&  MODE == 18)            ?
                              sample_11n
1615                        : ( (selector == -15 && (MODE == 13 || MODE == 23)) ?
                              sample_38n
1616                        : ( (selector == -15 && (MODE == 14 || MODE == 22)) ?
                              sample_26n
1617                        : ( (selector == -15 && (MODE == 15 || MODE == 21)) ?
                              sample_20n
1618                        : ( (selector == -15 && (MODE == 16 || MODE == 20)) ?
                              sample_16n
1619                        : ( (selector == -15 && (MODE == 17 || MODE == 19)) ?
                              sample_13n
1620                        : ( (selector == -15 &&  MODE == 18)            ?
                              sample_10n
1621                        : ( (selector == -14 && (MODE == 12 || MODE == 24)) ?
                              sample_63n
1622                        : ( (selector == -14 && (MODE == 13 || MODE == 23)) ?
                              sample_35n
1623                        : ( (selector == -14 && (MODE == 14 || MODE == 22)) ?
                              sample_24n
1624                        : ( (selector == -14 && (MODE == 15 || MODE == 21)) ?
                              sample_18n
1625                        : ( (selector == -14 && (MODE == 16 || MODE == 20)) ?
                              sample_14n
1626                        : ( (selector == -14 && (MODE == 17 || MODE == 19)) ?
                              sample_11n
1627                        : ( (selector == -14 &&  MODE == 18)            ? sample_9n
1628                        : ( (selector == -13 && (MODE == 12 || MODE == 24)) ?
                              sample_57n
1629                        : ( (selector == -13 && (MODE == 13 || MODE == 23)) ?
                              sample_31n
1630                        : ( (selector == -13 && (MODE == 14 || MODE == 22)) ?
                              sample_21n
1631                        : ( (selector == -13 && (MODE == 15 || MODE == 21)) ?
                              sample_16n
1632                        : ( (selector == -13 && (MODE == 16 || MODE == 20)) ?
                              sample_13n
1633                        : ( (selector == -13 && (MODE == 17 || MODE == 19)) ?
                              sample_10n
1634                        : ( (selector == -13 &&  MODE == 18)            ? sample_8n
1635                        : ( (selector == -12 && (MODE == 12 || MODE == 24)) ?
                              sample_50n
1636                        : ( (selector == -12 && (MODE == 13 || MODE == 23)) ?
```

```
                    sample_27n
1637              : ( (selector == -12 && (MODE == 14 || MODE == 22)) ?
                    sample_19n
1638              : ( (selector == -12 && (MODE == 15 || MODE == 21)) ?
                    sample_14n
1639              : ( (selector == -12 && (MODE == 16 || MODE == 20)) ?
                    sample_11n
1640              : ( (selector == -12 && (MODE == 17 || MODE == 19)) ?
                    sample_9n
1641              : ( (selector == -12 &&  MODE == 18)            ? sample_7n
1642              : ( (selector == -11 && (MODE == 12 || MODE == 24)) ?
                    sample_44n
1643              : ( (selector == -11 && (MODE == 13 || MODE == 23)) ?
                    sample_24n
1644              : ( (selector == -11 && (MODE == 14 || MODE == 22)) ?
                    sample_16n
1645              : ( (selector == -11 && (MODE == 15 || MODE == 21)) ?
                    sample_12n
1646              : ( (selector == -11 && (MODE == 16 || MODE == 20)) ?
                    sample_10n
1647              : ( (selector == -11 && (MODE == 17 || MODE == 19)) ?
                    sample_8n
1648              : ( (selector == -11 &&  MODE == 18)            ? sample_6n
1649              : ( (selector == -10 && (MODE == 12 || MODE == 24)) ?
                    sample_37n
1650              : ( (selector == -10 && (MODE == 13 || MODE == 23)) ?
                    sample_20n
1651              : ( (selector == -10 && (MODE == 14 || MODE == 22)) ?
                    sample_14n
1652              : ( (selector == -10 && (MODE == 15 || MODE == 21)) ?
                    sample_10n
1653              : ( (selector == -10 && (MODE == 16 || MODE == 20)) ?
                    sample_8n
1654              : ( (selector == -10 && (MODE == 17 || MODE == 19)) ?
                    sample_6n
1655              : ( (selector == -10 &&  MODE == 18)            ? sample_5n
1656              : ( (selector ==  -9 && (MODE == 12 || MODE == 24)) ?
                    sample_31n
1657              : ( (selector ==  -9 && (MODE == 13 || MODE == 23)) ?
                    sample_17n
1658              : ( (selector ==  -9 && (MODE == 14 || MODE == 22)) ?
                    sample_11n
1659              : ( (selector ==  -9 && (MODE == 15 || MODE == 21)) ?
                    sample_8n
1660              : ( (selector ==  -9 && (MODE == 16 || MODE == 20)) ?
                    sample_7n
1661              : ( (selector ==  -9 && (MODE == 17 || MODE == 19)) ?
                    sample_5n
1662              : ( (selector ==  -9 &&  MODE == 18)            ? sample_4n
1663              : ( (selector ==  -8 && (MODE == 11 || MODE == 25)) ?
                    sample_63n
1664              : ( (selector ==  -8 && (MODE == 12 || MODE == 24)) ?
                    sample_25n
1665              : ( (selector ==  -8 && (MODE == 13 || MODE == 23)) ?
                    sample_13n
1666              : ( (selector ==  -8 && (MODE == 14 || MODE == 22)) ?
                    sample_9n
```

```
1667                        : ( (selector ==  -8 && (MODE == 15 || MODE == 21)) ?
                              sample_7n
1668                        : ( (selector ==  -8 && (MODE == 16 || MODE == 20)) ?
                              sample_5n
1669                        : ( (selector ==  -8 && (MODE == 17 || MODE == 19)) ?
                              sample_4n
1670                        : ( (selector ==  -8 &&  MODE == 18)            ? sample_3n
1671                        : ( (selector ==  -7 && (MODE == 11 || MODE == 25)) ?
                              sample_47n
1672                        : ( (selector ==  -7 && (MODE == 12 || MODE == 24)) ?
                              sample_18n
1673                        : ( (selector ==  -7 && (MODE == 13 || MODE == 23)) ?
                              sample_10n
1674                        : ( (selector ==  -7 && (MODE == 14 || MODE == 22)) ?
                              sample_6n
1675                        : ( (selector ==  -7 && (MODE == 15 || MODE == 21)) ?
                              sample_5n
1676                        : ( (selector ==  -7 && (MODE == 16 || MODE == 20)) ?
                              sample_4n
1677                        : ( (selector ==  -7 && (MODE == 17 || MODE == 19)) ?
                              sample_3n
1678                        : ( (selector ==  -7 &&  MODE == 18)            ? sample_2n
1679                        : ( (selector ==  -6 && (MODE == 11 || MODE == 25)) ?
                              sample_31n
1680                        : ( (selector ==  -6 && (MODE == 12 || MODE == 24)) ?
                              sample_12n
1681                        : ( (selector ==  -6 && (MODE == 13 || MODE == 23)) ?
                              sample_6n
1682                        : ( (selector ==  -6 && (MODE == 14 || MODE == 22)) ?
                              sample_4n
1683                        : ( (selector ==  -6 && (MODE == 15 || MODE == 21)) ?
                              sample_3n
1684                        : ( (selector ==  -6 && (MODE == 16 || MODE == 20)) ?
                              sample_2n
1685                        : ( (selector ==  -6 && (MODE == 17 || MODE == 19)) ?
                              sample_1n
1686                        : ( (selector ==  -6 &&  MODE == 18)            ? sample_1n
1687                        : ( (selector ==  -5 && (MODE == 11 || MODE == 25)) ?
                              sample_15n
1688                        : ( (selector ==  -5 && (MODE == 12 || MODE == 24)) ?
                              sample_5n
1689                        : ( (selector ==  -5 && (MODE == 13 || MODE == 23)) ?
                              sample_3n
1690                        : ( (selector ==  -5 && (MODE == 14 || MODE == 22)) ?
                              sample_1n
1691                        : ( (selector ==  -5 && (MODE == 15 || MODE == 21)) ?
                              sample_1n
1692                        : ( (selector ==  -5 && (MODE == 16 || MODE == 20)) ?
                              sample_1n
1693                        : ( (selector ==  -5 && (MODE == 17 || MODE == 19)) ?
                              sample_0n
1694                        : ( (selector ==  -5 &&  MODE == 18)            ? sample_0n
1695                        : (selector ==  -4 ? sample_NN
1696                        : (selector ==  -3 ? sample_n0
1697                        : (selector ==  -2 ? sample_n1
1698                        : (selector ==  -1 ? sample_n2
1699                        : (selector ==   0 ? sample_n3
```

```
                        : (selector ==   1 ? sample_n4
                        : (selector ==   2 ? sample_n5
                        : (selector ==   3 ? sample_n6
                        : (selector ==   4 ? sample_n7
                        : (selector ==   5 ? sample_n8
                        : (selector ==   6 ? sample_n9
                        : (selector ==   7 ? sample_n10
                        : (selector ==   8 ? sample_n11
                        : (selector ==   9 ? sample_n12
                        : (selector ==  10 ? sample_n13
                        : (selector ==  11 ? sample_n14
                        : (selector ==  12 ? sample_n15
                        : (selector ==  13 ? sample_n16
                        : (selector ==  14 ? sample_n17
                        : (selector ==  15 ? sample_n18
                        : (selector ==  16 ? sample_n19
                        : (selector ==  17 ? sample_n20
                        : (selector ==  18 ? sample_n21
                        : (selector ==  19 ? sample_n22
                        : (selector ==  20 ? sample_n23
                        : (selector ==  21 ? sample_n24
                        : (selector ==  22 ? sample_n25
                        : (selector ==  23 ? sample_n26
                        : (selector ==  24 ? sample_n27
                        : (selector ==  25 ? sample_n28
                        : (selector ==  26 ? sample_n29
                        : (selector ==  27 ? sample_n30
                        : (selector ==  28 ? sample_n31
                        : (selector ==  29 ? sample_n32
                        : (selector ==  30 ? sample_n33
                        : (selector ==  31 ? sample_n34
                        : (selector ==  32 ? sample_n35
                        : (selector ==  33 ? sample_n36
                        : (selector ==  34 ? sample_n37
                        : (selector ==  35 ? sample_n38
                        : (selector ==  36 ? sample_n39
                        : (selector ==  37 ? sample_n40
                        : (selector ==  38 ? sample_n41
                        : (selector ==  39 ? sample_n42
                        : (selector ==  40 ? sample_n43
                        : (selector ==  41 ? sample_n44
                        : (selector ==  42 ? sample_n45
                        : (selector ==  43 ? sample_n46
                        : (selector ==  44 ? sample_n47
                        : (selector ==  45 ? sample_n48
                        : (selector ==  46 ? sample_n49
                        : (selector ==  47 ? sample_n50
                        : (selector ==  48 ? sample_n51
                        : (selector ==  49 ? sample_n52
                        : (selector ==  50 ? sample_n53
                        : (selector ==  51 ? sample_n54
                        : (selector ==  52 ? sample_n55
                        : (selector ==  53 ? sample_n56
                        : (selector ==  54 ? sample_n57
                        : (selector ==  55 ? sample_n58
                        : (selector ==  56 ? sample_n59
                        : (selector ==  57 ? sample_n60
```

```
                        : (selector ==  58 ? sample_n61
                        : (selector ==  59 ? sample_n62
                        : (selector ==  60 ? sample_n63
                        : 0 ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                            ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                            ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                            ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                            ))))))))))))))))
                        : ( (selector == -32 && (MODE == 15 || MODE == 21)) ?
                            sample_n52
                        : ( (selector == -32 && (MODE == 16 || MODE == 20)) ?
                            sample_n42
                        : ( (selector == -32 && (MODE == 17 || MODE == 19)) ?
                            sample_n33
                        : ( (selector == -32 &&  MODE == 18)            ?
                            sample_n27
                        : ( (selector == -31 && (MODE == 15 || MODE == 21)) ?
                            sample_n50
                        : ( (selector == -31 && (MODE == 16 || MODE == 20)) ?
                            sample_n40
                        : ( (selector == -31 && (MODE == 17 || MODE == 19)) ?
                            sample_n32
                        : ( (selector == -31 &&  MODE == 18)            ?
                            sample_n26
                        : ( (selector == -30 && (MODE == 14 || MODE == 22)) ?
                            sample_n63
                        : ( (selector == -30 && (MODE == 15 || MODE == 21)) ?
                            sample_n48
                        : ( (selector == -30 && (MODE == 16 || MODE == 20)) ?
                            sample_n39
                        : ( (selector == -30 && (MODE == 17 || MODE == 19)) ?
                            sample_n31
                        : ( (selector == -30 &&  MODE == 18)            ?
                            sample_n25
                        : ( (selector == -29 && (MODE == 14 || MODE == 22)) ?
                            sample_n61
                        : ( (selector == -29 && (MODE == 15 || MODE == 21)) ?
                            sample_n46
                        : ( (selector == -29 && (MODE == 16 || MODE == 20)) ?
                            sample_n37
                        : ( (selector == -29 && (MODE == 17 || MODE == 19)) ?
                            sample_n30
                        : ( (selector == -29 &&  MODE == 18)            ?
                            sample_n24
                        : ( (selector == -28 && (MODE == 14 || MODE == 22)) ?
                            sample_n58
                        : ( (selector == -28 && (MODE == 15 || MODE == 21)) ?
                            sample_n44
                        : ( (selector == -28 && (MODE == 16 || MODE == 20)) ?
                            sample_n36
                        : ( (selector == -28 && (MODE == 17 || MODE == 19)) ?
                            sample_n29
                        : ( (selector == -28 &&  MODE == 18)            ?
                            sample_n23
                        : ( (selector == -27 && (MODE == 14 || MODE == 22)) ?
                            sample_n56
                        : ( (selector == -27 && (MODE == 15 || MODE == 21)) ?
```

```
                    sample_n42
1786              : ( (selector == -27 && (MODE == 16 || MODE == 20)) ?
                    sample_n34
1787              : ( (selector == -27 && (MODE == 17 || MODE == 19)) ?
                    sample_n27
1788              : ( (selector == -27 &&  MODE == 18)            ?
                    sample_n22
1789              : ( (selector == -26 && (MODE == 14 || MODE == 22)) ?
                    sample_n53
1790              : ( (selector == -26 && (MODE == 15 || MODE == 21)) ?
                    sample_n40
1791              : ( (selector == -26 && (MODE == 16 || MODE == 20)) ?
                    sample_n33
1792              : ( (selector == -26 && (MODE == 17 || MODE == 19)) ?
                    sample_n26
1793              : ( (selector == -26 &&  MODE == 18)            ?
                    sample_n21
1794              : ( (selector == -25 && (MODE == 14 || MODE == 22)) ?
                    sample_n51
1795              : ( (selector == -25 && (MODE == 15 || MODE == 21)) ?
                    sample_n39
1796              : ( (selector == -25 && (MODE == 16 || MODE == 20)) ?
                    sample_n31
1797              : ( (selector == -25 && (MODE == 17 || MODE == 19)) ?
                    sample_n25
1798              : ( (selector == -25 &&  MODE == 18)            ?
                    sample_n20
1799              : ( (selector == -24 && (MODE == 14 || MODE == 22)) ?
                    sample_n48
1800              : ( (selector == -24 && (MODE == 15 || MODE == 21)) ?
                    sample_n37
1801              : ( (selector == -24 && (MODE == 16 || MODE == 20)) ?
                    sample_n29
1802              : ( (selector == -24 && (MODE == 17 || MODE == 19)) ?
                    sample_n24
1803              : ( (selector == -24 &&  MODE == 18)            ?
                    sample_n19
1804              : ( (selector == -23 && (MODE == 14 || MODE == 22)) ?
                    sample_n46
1805              : ( (selector == -23 && (MODE == 15 || MODE == 21)) ?
                    sample_n35
1806              : ( (selector == -23 && (MODE == 16 || MODE == 20)) ?
                    sample_n28
1807              : ( (selector == -23 && (MODE == 17 || MODE == 19)) ?
                    sample_n22
1808              : ( (selector == -23 &&  MODE == 18)            ?
                    sample_n18
1809              : ( (selector == -22 && (MODE == 13 || MODE == 23)) ?
                    sample_n63
1810              : ( (selector == -22 && (MODE == 14 || MODE == 22)) ?
                    sample_n43
1811              : ( (selector == -22 && (MODE == 15 || MODE == 21)) ?
                    sample_n33
1812              : ( (selector == -22 && (MODE == 16 || MODE == 20)) ?
                    sample_n26
1813              : ( (selector == -22 && (MODE == 17 || MODE == 19)) ?
                    sample_n21
```

```
                    : ( (selector == -22 &&  MODE == 18)            ?
                      sample_n17
                    : ( (selector == -21 && (MODE == 13 || MODE == 23)) ?
                      sample_n59
                    : ( (selector == -21 && (MODE == 14 || MODE == 22)) ?
                      sample_n41
                    : ( (selector == -21 && (MODE == 15 || MODE == 21)) ?
                      sample_n31
                    : ( (selector == -21 && (MODE == 16 || MODE == 20)) ?
                      sample_n25
                    : ( (selector == -21 && (MODE == 17 || MODE == 19)) ?
                      sample_n20
                    : ( (selector == -21 &&  MODE == 18)            ?
                      sample_n16
                    : ( (selector == -20 && (MODE == 13 || MODE == 23)) ?
                      sample_n56
                    : ( (selector == -20 && (MODE == 14 || MODE == 22)) ?
                      sample_n38
                    : ( (selector == -20 && (MODE == 15 || MODE == 21)) ?
                      sample_n29
                    : ( (selector == -20 && (MODE == 16 || MODE == 20)) ?
                      sample_n23
                    : ( (selector == -20 && (MODE == 17 || MODE == 19)) ?
                      sample_n19
                    : ( (selector == -20 &&  MODE == 18)            ?
                      sample_n15
                    : ( (selector == -19 && (MODE == 13 || MODE == 23)) ?
                      sample_n52
                    : ( (selector == -19 && (MODE == 14 || MODE == 22)) ?
                      sample_n36
                    : ( (selector == -19 && (MODE == 15 || MODE == 21)) ?
                      sample_n27
                    : ( (selector == -19 && (MODE == 16 || MODE == 20)) ?
                      sample_n22
                    : ( (selector == -19 && (MODE == 17 || MODE == 19)) ?
                      sample_n17
                    : ( (selector == -19 &&  MODE == 18)            ?
                      sample_n14
                    : ( (selector == -18 && (MODE == 13 || MODE == 23)) ?
                      sample_n49
                    : ( (selector == -18 && (MODE == 14 || MODE == 22)) ?
                      sample_n33
                    : ( (selector == -18 && (MODE == 15 || MODE == 21)) ?
                      sample_n25
                    : ( (selector == -18 && (MODE == 16 || MODE == 20)) ?
                      sample_n20
                    : ( (selector == -18 && (MODE == 17 || MODE == 19)) ?
                      sample_n16
                    : ( (selector == -18 &&  MODE == 18)            ?
                      sample_n13
                    : ( (selector == -17 && (MODE == 13 || MODE == 23)) ?
                      sample_n45
                    : ( (selector == -17 && (MODE == 14 || MODE == 22)) ?
                      sample_n31
                    : ( (selector == -17 && (MODE == 15 || MODE == 21)) ?
                      sample_n23
                    : ( (selector == -17 && (MODE == 16 || MODE == 20)) ?
```

```
                        sample_n19
1843                  : ( (selector == -17 && (MODE == 17 || MODE == 19)) ?
                        sample_n15
1844                  : ( (selector == -17 &&  MODE == 18)            ?
                        sample_n12
1845                  : ( (selector == -16 && (MODE == 13 || MODE == 23)) ?
                        sample_n42
1846                  : ( (selector == -16 && (MODE == 14 || MODE == 22)) ?
                        sample_n29
1847                  : ( (selector == -16 && (MODE == 15 || MODE == 21)) ?
                        sample_n22
1848                  : ( (selector == -16 && (MODE == 16 || MODE == 20)) ?
                        sample_n17
1849                  : ( (selector == -16 && (MODE == 17 || MODE == 19)) ?
                        sample_n14
1850                  : ( (selector == -16 &&  MODE == 18)            ?
                        sample_n11
1851                  : ( (selector == -15 && (MODE == 13 || MODE == 23)) ?
                        sample_n38
1852                  : ( (selector == -15 && (MODE == 14 || MODE == 22)) ?
                        sample_n26
1853                  : ( (selector == -15 && (MODE == 15 || MODE == 21)) ?
                        sample_n20
1854                  : ( (selector == -15 && (MODE == 16 || MODE == 20)) ?
                        sample_n16
1855                  : ( (selector == -15 && (MODE == 17 || MODE == 19)) ?
                        sample_n13
1856                  : ( (selector == -15 &&  MODE == 18)            ?
                        sample_n10
1857                  : ( (selector == -14 && (MODE == 12 || MODE == 24)) ?
                        sample_n63
1858                  : ( (selector == -14 && (MODE == 13 || MODE == 23)) ?
                        sample_n35
1859                  : ( (selector == -14 && (MODE == 14 || MODE == 22)) ?
                        sample_n24
1860                  : ( (selector == -14 && (MODE == 15 || MODE == 21)) ?
                        sample_n18
1861                  : ( (selector == -14 && (MODE == 16 || MODE == 20)) ?
                        sample_n14
1862                  : ( (selector == -14 && (MODE == 17 || MODE == 19)) ?
                        sample_n11
1863                  : ( (selector == -14 &&  MODE == 18)            ? sample_n9
1864                  : ( (selector == -13 && (MODE == 12 || MODE == 24)) ?
                        sample_n57
1865                  : ( (selector == -13 && (MODE == 13 || MODE == 23)) ?
                        sample_n31
1866                  : ( (selector == -13 && (MODE == 14 || MODE == 22)) ?
                        sample_n21
1867                  : ( (selector == -13 && (MODE == 15 || MODE == 21)) ?
                        sample_n16
1868                  : ( (selector == -13 && (MODE == 16 || MODE == 20)) ?
                        sample_n13
1869                  : ( (selector == -13 && (MODE == 17 || MODE == 19)) ?
                        sample_n10
1870                  : ( (selector == -13 &&  MODE == 18)            ? sample_n8
1871                  : ( (selector == -12 && (MODE == 12 || MODE == 24)) ?
                        sample_n50
```

```
1872                        : ( (selector == -12 && (MODE == 13 || MODE == 23)) ?
                              sample_n27
1873                        : ( (selector == -12 && (MODE == 14 || MODE == 22)) ?
                              sample_n19
1874                        : ( (selector == -12 && (MODE == 15 || MODE == 21)) ?
                              sample_n14
1875                        : ( (selector == -12 && (MODE == 16 || MODE == 20)) ?
                              sample_n11
1876                        : ( (selector == -12 && (MODE == 17 || MODE == 19)) ?
                              sample_n9
1877                        : ( (selector == -12 &&  MODE == 18)            ? sample_n7
1878                        : ( (selector == -11 && (MODE == 12 || MODE == 24)) ?
                              sample_n44
1879                        : ( (selector == -11 && (MODE == 13 || MODE == 23)) ?
                              sample_n24
1880                        : ( (selector == -11 && (MODE == 14 || MODE == 22)) ?
                              sample_n16
1881                        : ( (selector == -11 && (MODE == 15 || MODE == 21)) ?
                              sample_n12
1882                        : ( (selector == -11 && (MODE == 16 || MODE == 20)) ?
                              sample_n10
1883                        : ( (selector == -11 && (MODE == 17 || MODE == 19)) ?
                              sample_n8
1884                        : ( (selector == -11 &&  MODE == 18)            ? sample_n6
1885                        : ( (selector == -10 && (MODE == 12 || MODE == 24)) ?
                              sample_n37
1886                        : ( (selector == -10 && (MODE == 13 || MODE == 23)) ?
                              sample_n20
1887                        : ( (selector == -10 && (MODE == 14 || MODE == 22)) ?
                              sample_n14
1888                        : ( (selector == -10 && (MODE == 15 || MODE == 21)) ?
                              sample_n10
1889                        : ( (selector == -10 && (MODE == 16 || MODE == 20)) ?
                              sample_n8
1890                        : ( (selector == -10 && (MODE == 17 || MODE == 19)) ?
                              sample_n6
1891                        : ( (selector == -10 &&  MODE == 18)            ? sample_n5
1892                        : ( (selector ==  -9 && (MODE == 12 || MODE == 24)) ?
                              sample_n31
1893                        : ( (selector ==  -9 && (MODE == 13 || MODE == 23)) ?
                              sample_n17
1894                        : ( (selector ==  -9 && (MODE == 14 || MODE == 22)) ?
                              sample_n11
1895                        : ( (selector ==  -9 && (MODE == 15 || MODE == 21)) ?
                              sample_n8
1896                        : ( (selector ==  -9 && (MODE == 16 || MODE == 20)) ?
                              sample_n7
1897                        : ( (selector ==  -9 && (MODE == 17 || MODE == 19)) ?
                              sample_n5
1898                        : ( (selector ==  -9 &&  MODE == 18)            ? sample_n4
1899                        : ( (selector ==  -8 && (MODE == 11 || MODE == 25)) ?
                              sample_n63
1900                        : ( (selector ==  -8 && (MODE == 12 || MODE == 24)) ?
                              sample_n25
1901                        : ( (selector ==  -8 && (MODE == 13 || MODE == 23)) ?
                              sample_n13
1902                        : ( (selector ==  -8 && (MODE == 14 || MODE == 22)) ?
```

```
                        sample_n9
1903                  : ( (selector ==  -8 && (MODE == 15 || MODE == 21)) ?
                        sample_n7
1904                  : ( (selector ==  -8 && (MODE == 16 || MODE == 20)) ?
                        sample_n5
1905                  : ( (selector ==  -8 && (MODE == 17 || MODE == 19)) ?
                        sample_n4
1906                  : ( (selector ==  -8 &&  MODE == 18)            ? sample_n3
1907                  : ( (selector ==  -7 && (MODE == 11 || MODE == 25)) ?
                        sample_n47
1908                  : ( (selector ==  -7 && (MODE == 12 || MODE == 24)) ?
                        sample_n18
1909                  : ( (selector ==  -7 && (MODE == 13 || MODE == 23)) ?
                        sample_n10
1910                  : ( (selector ==  -7 && (MODE == 14 || MODE == 22)) ?
                        sample_n6
1911                  : ( (selector ==  -7 && (MODE == 15 || MODE == 21)) ?
                        sample_n5
1912                  : ( (selector ==  -7 && (MODE == 16 || MODE == 20)) ?
                        sample_n4
1913                  : ( (selector ==  -7 && (MODE == 17 || MODE == 19)) ?
                        sample_n3
1914                  : ( (selector ==  -7 &&  MODE == 18)            ? sample_n2
1915                  : ( (selector ==  -6 && (MODE == 11 || MODE == 25)) ?
                        sample_n31
1916                  : ( (selector ==  -6 && (MODE == 12 || MODE == 24)) ?
                        sample_n12
1917                  : ( (selector ==  -6 && (MODE == 13 || MODE == 23)) ?
                        sample_n6
1918                  : ( (selector ==  -6 && (MODE == 14 || MODE == 22)) ?
                        sample_n4
1919                  : ( (selector ==  -6 && (MODE == 15 || MODE == 21)) ?
                        sample_n3
1920                  : ( (selector ==  -6 && (MODE == 16 || MODE == 20)) ?
                        sample_n2
1921                  : ( (selector ==  -6 && (MODE == 17 || MODE == 19)) ?
                        sample_n1
1922                  : ( (selector ==  -6 &&  MODE == 18)            ? sample_n1
1923                  : ( (selector ==  -5 && (MODE == 11 || MODE == 25)) ?
                        sample_n15
1924                  : ( (selector ==  -5 && (MODE == 12 || MODE == 24)) ?
                        sample_n5
1925                  : ( (selector ==  -5 && (MODE == 13 || MODE == 23)) ?
                        sample_n3
1926                  : ( (selector ==  -5 && (MODE == 14 || MODE == 22)) ?
                        sample_n1
1927                  : ( (selector ==  -5 && (MODE == 15 || MODE == 21)) ?
                        sample_n1
1928                  : ( (selector ==  -5 && (MODE == 16 || MODE == 20)) ?
                        sample_n1
1929                  : ( (selector ==  -5 && (MODE == 17 || MODE == 19)) ?
                        sample_n0
1930                  : ( (selector ==  -5 &&  MODE == 18)            ? sample_n0
1931                  : (selector ==  -4 ? sample_NN
1932                  : (selector ==  -3 ? sample_0n
1933                  : (selector ==  -2 ? sample_1n
1934                  : (selector ==  -1 ? sample_2n
```

```
                        : (selector ==   0 ? sample_3n
                        : (selector ==   1 ? sample_4n
                        : (selector ==   2 ? sample_5n
                        : (selector ==   3 ? sample_6n
                        : (selector ==   4 ? sample_7n
                        : (selector ==   5 ? sample_8n
                        : (selector ==   6 ? sample_9n
                        : (selector ==   7 ? sample_10n
                        : (selector ==   8 ? sample_11n
                        : (selector ==   9 ? sample_12n
                        : (selector ==  10 ? sample_13n
                        : (selector ==  11 ? sample_14n
                        : (selector ==  12 ? sample_15n
                        : (selector ==  13 ? sample_16n
                        : (selector ==  14 ? sample_17n
                        : (selector ==  15 ? sample_18n
                        : (selector ==  16 ? sample_19n
                        : (selector ==  17 ? sample_20n
                        : (selector ==  18 ? sample_21n
                        : (selector ==  19 ? sample_22n
                        : (selector ==  20 ? sample_23n
                        : (selector ==  21 ? sample_24n
                        : (selector ==  22 ? sample_25n
                        : (selector ==  23 ? sample_26n
                        : (selector ==  24 ? sample_27n
                        : (selector ==  25 ? sample_28n
                        : (selector ==  26 ? sample_29n
                        : (selector ==  27 ? sample_30n
                        : (selector ==  28 ? sample_31n
                        : (selector ==  29 ? sample_32n
                        : (selector ==  30 ? sample_33n
                        : (selector ==  31 ? sample_34n
                        : (selector ==  32 ? sample_35n
                        : (selector ==  33 ? sample_36n
                        : (selector ==  34 ? sample_37n
                        : (selector ==  35 ? sample_38n
                        : (selector ==  36 ? sample_39n
                        : (selector ==  37 ? sample_40n
                        : (selector ==  38 ? sample_41n
                        : (selector ==  39 ? sample_42n
                        : (selector ==  40 ? sample_43n
                        : (selector ==  41 ? sample_44n
                        : (selector ==  42 ? sample_45n
                        : (selector ==  43 ? sample_46n
                        : (selector ==  44 ? sample_47n
                        : (selector ==  45 ? sample_48n
                        : (selector ==  46 ? sample_49n
                        : (selector ==  47 ? sample_50n
                        : (selector ==  48 ? sample_51n
                        : (selector ==  49 ? sample_52n
                        : (selector ==  50 ? sample_53n
                        : (selector ==  51 ? sample_54n
                        : (selector ==  52 ? sample_55n
                        : (selector ==  53 ? sample_56n
                        : (selector ==  54 ? sample_57n
                        : (selector ==  55 ? sample_58n
                        : (selector ==  56 ? sample_59n
```

```
                        : (selector ==  57 ? sample_60n
                        : (selector ==  58 ? sample_61n
                        : (selector ==  59 ? sample_62n
                        : (selector ==  60 ? sample_63n
                        : 0 ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                            ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                            ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                            ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                            ))))))))))))))));

    assign ref5 = MODE == 10   ? ( index ==  0 ? sample_0n
                               : ( index ==  1 ? sample_1n
                               : ( index ==  2 ? sample_2n
                               : ( index ==  3 ? sample_3n
                               : ( index ==  4 ? sample_4n
                               : ( index ==  5 ? sample_5n
                               : ( index ==  6 ? sample_6n
                               : ( index ==  7 ? sample_7n
                               : ( index ==  8 ? sample_8n
                               : ( index ==  9 ? sample_9n
                               : ( index == 10 ? sample_10n
                               : ( index == 11 ? sample_11n
                               : ( index == 12 ? sample_12n
                               : ( index == 13 ? sample_13n
                               : ( index == 14 ? sample_14n
                               : ( index == 15 ? sample_15n
                               : ( index == 16 ? sample_16n
                               : ( index == 17 ? sample_17n
                               : ( index == 18 ? sample_18n
                               : ( index == 19 ? sample_19n
                               : ( index == 20 ? sample_20n
                               : ( index == 21 ? sample_21n
                               : ( index == 22 ? sample_22n
                               : ( index == 23 ? sample_23n
                               : ( index == 24 ? sample_24n
                               : ( index == 25 ? sample_25n
                               : ( index == 26 ? sample_26n
                               : ( index == 27 ? sample_27n
                               : ( index == 28 ? sample_28n
                               : ( index == 29 ? sample_29n
                               : ( index == 30 ? sample_30n
                               : ( index == 31 ? sample_31n
                               : 0   ))))))))))))))))))))))))))))))))
                               : ( index ==  0 ? sample_n0
                               : ( index ==  1 ? sample_n1
                               : ( index ==  2 ? sample_n2
                               : ( index ==  3 ? sample_n3
                               : ( index ==  4 ? sample_n4
                               : ( index ==  5 ? sample_n5
                               : ( index ==  6 ? sample_n6
                               : ( index ==  7 ? sample_n7
                               : ( index ==  8 ? sample_n8
                               : ( index ==  9 ? sample_n9
                               : ( index == 10 ? sample_n10
                               : ( index == 11 ? sample_n11
                               : ( index == 12 ? sample_n12
                               : ( index == 13 ? sample_n13
```

```
2045                          : ( index == 14 ? sample_n14
2046                          : ( index == 15 ? sample_n15
2047                          : ( index == 16 ? sample_n16
2048                          : ( index == 17 ? sample_n17
2049                          : ( index == 18 ? sample_n18
2050                          : ( index == 19 ? sample_n19
2051                          : ( index == 20 ? sample_n20
2052                          : ( index == 21 ? sample_n21
2053                          : ( index == 22 ? sample_n22
2054                          : ( index == 23 ? sample_n23
2055                          : ( index == 24 ? sample_n24
2056                          : ( index == 25 ? sample_n25
2057                          : ( index == 26 ? sample_n26
2058                          : ( index == 27 ? sample_n27
2059                          : ( index == 28 ? sample_n28
2060                          : ( index == 29 ? sample_n29
2061                          : ( index == 30 ? sample_n30
2062                          : ( index == 31 ? sample_n31
2063                          : 0 ))))))))))))))))))))))))))))))));
2064
2065 endmodule
```

Listing B.8 – Código fonte da interface do preditor diretamente angular

```
1
2  module angular_directly_top_level
3
4     #(parameter WIDTH = 8, parameter MODE = 10)
5
6     (
7        input clock,
8              reset,
9              start,
10
11       input [WIDTH-1:0] ref0,
12                         ref1,
13                         ref2,
14                         ref3,
15                         ref4,
16                         ref5,
17
18       input[WIDTH-3:0]  N,
19
20       output[WIDTH-2:0] filter_samples,
21
22       output signed [WIDTH-1:0]  samples,
23
24       output   part_done,
25             done,
26
27       output [WIDTH-1:0]   p00,
28                         p01,
29                         p02,
30                         p03,
31                         p10,
32                         p11,
33                         p12,
```

```
34                      p13,
35                      p20,
36                      p21,
37                      p22,
38                      p23,
39                      p30,
40                      p31,
41                      p32,
42                      p33
43      );
44
45      wire    m_condition,
46              i_condition,
47              reset_registers,
48              m_reset,
49              def_enable,
50              refs_enable,
51              shift_enable,
52              i_index_enable,
53              m_index_enable,
54              indexes_enable;
55
56      angular_directly_control angular_control_block(
57         .clock(clock),
58         .reset(reset),
59         .start(start),
60         .m_condition(m_condition),
61         .i_condition(i_condition),
62
63         .reset_registers(reset_registers),
64         .m_reset(m_reset),
65         .def_enable(def_enable),
66         .refs_enable(refs_enable),
67         .shift_enable(shift_enable),
68         .i_index_enable(i_index_enable),
69         .m_index_enable(m_index_enable),
70         .indexes_enable(indexes_enable),
71         .part_done(part_done),
72         .done(done)
73      );
74
75      angular_directly_operative #(WIDTH, MODE) angular_operative_block(
76         .clock(clock),
77         .reset(reset_registers),
78         .m_reset(m_reset),
79         .def_enable(def_enable),
80         .refs_enable(refs_enable),
81         .shift_enable(shift_enable),
82         .i_index_enable(i_index_enable),
83         .m_index_enable(m_index_enable),
84         .indexes_enable(indexes_enable),
85         .ref0(ref0),
86         .ref1(ref1),
87         .ref2(ref2),
88         .ref3(ref3),
89         .ref4(ref4),
90         .ref5(ref5),
```

```
91          .N(N),
92
93          .m_condition(m_condition),
94          .i_condition(i_condition),
95          .filter(filter_samples),
96          .samples(samples),
97          .p00(p00),
98          .p01(p01),
99          .p02(p02),
100         .p03(p03),
101         .p10(p10),
102         .p11(p11),
103         .p12(p12),
104         .p13(p13),
105         .p20(p20),
106         .p21(p21),
107         .p22(p22),
108         .p23(p23),
109         .p30(p30),
110         .p31(p31),
111         .p32(p32),
112         .p33(p33)
113     );
114
115 endmodule
```

Listing B.9 – Código fonte do bloco de controle do preditor angular

```
1
2  module angular_control
3
4      (
5          input     clock,
6                reset,
7                start,
8                m_condition,
9                i_condition,
10
11         output    reset_registers,
12               m_reset,
13               def_enable,
14               refs_enable,
15               shift_enable,
16               i_index_enable,
17               m_index_enable,
18               indexes_enable,
19               displacement_enable,
20               part_done,
21               done
22     );
23
24     reg [3:0] state;
25
26     reg reg_reset_registers,
27         reg_m_reset,
28         reg_def_enable,
29         reg_refs_enable,
```

```
30        reg_shift_enable,
31        reg_i_index_enable,
32        reg_m_index_enable,
33        reg_indexes_enable,
34        reg_displacement_enable,
35        reg_part_done,
36        reg_done;
37
38     parameter    INIT = 0,
39              PRE_LOAD = 1,
40              TIME1 = 2,
41              LOAD = 3,
42              TIME2 = 4,
43              CALC = 5,
44              ITEST = 6,
45              UPDATE = 7,
46              DONE = 8;
47
48     always @(state)
49     begin
50        case (state)
51
52           INIT:
53              begin
54                 reg_reset_registers     = 1'b1;
55                 reg_m_reset          = 1'b0;
56                 reg_def_enable       = 1'b0;
57                 reg_refs_enable         = 1'b0;
58                 reg_shift_enable     = 1'b0;
59                 reg_i_index_enable      = 1'b0;
60                 reg_m_index_enable      = 1'b0;
61                 reg_indexes_enable      = 1'b0;
62                 reg_displacement_enable = 1'b0;
63
64                 reg_part_done        = 1'b0;
65                 reg_done          = 1'b0;
66              end
67
68           PRE_LOAD:
69              begin
70                 reg_reset_registers     = 1'b0;
71                 reg_m_reset          = 1'b0;
72                 reg_def_enable       = 1'b1;
73                 reg_refs_enable         = 1'b0;
74                 reg_shift_enable     = 1'b0;
75                 reg_i_index_enable      = 1'b0;
76                 reg_m_index_enable      = 1'b0;
77                 reg_indexes_enable      = 1'b0;
78                 reg_displacement_enable = 1'b1;
79
80                 reg_part_done        = 1'b0;
81                 reg_done          = 1'b0;
82              end
83
84           TIME1:
85              begin
86                 reg_reset_registers     = 1'b0;
```

```
87                reg_m_reset           = 1'b0;
88                reg_def_enable        = 1'b0;
89                reg_refs_enable          = 1'b0;
90                reg_shift_enable      = 1'b0;
91                reg_i_index_enable       = 1'b0;
92                reg_m_index_enable       = 1'b1;
93                reg_indexes_enable       = 1'b0;
94                reg_displacement_enable = 1'b0;
95
96                reg_part_done         = 1'b0;
97                reg_done           = 1'b0;
98             end
99
100         LOAD:
101            begin
102               reg_reset_registers      = 1'b0;
103               reg_m_reset           = 1'b0;
104               reg_def_enable        = 1'b0;
105               reg_refs_enable          = 1'b1;
106               reg_shift_enable      = 1'b0;
107               reg_i_index_enable       = 1'b0;
108               reg_m_index_enable       = 1'b0;
109               reg_indexes_enable       = 1'b1;
110               reg_displacement_enable = 1'b0;
111
112               reg_part_done         = 1'b0;
113               reg_done           = 1'b0;
114            end
115
116         TIME2:
117            begin
118               reg_reset_registers      = 1'b0;
119               reg_m_reset           = 1'b0;
120               reg_def_enable        = 1'b0;
121               reg_refs_enable          = 1'b0;
122               reg_shift_enable      = 1'b0;
123               reg_i_index_enable       = 1'b0;
124               reg_m_index_enable       = 1'b0;
125               reg_indexes_enable       = 1'b0;
126               reg_displacement_enable = 1'b0;
127
128               reg_part_done         = 1'b0;
129               reg_done           = 1'b0;
130            end
131
132         CALC:
133            begin
134               reg_reset_registers      = 1'b0;
135               reg_m_reset           = 1'b0;
136               reg_def_enable        = 1'b0;
137               reg_refs_enable          = 1'b0;
138               reg_shift_enable      = 1'b1;
139               reg_i_index_enable       = 1'b0;
140               reg_m_index_enable       = 1'b0;
141               reg_indexes_enable       = 1'b0;
142               reg_displacement_enable = 1'b1;
143
```

```
144                reg_part_done         = 1'b0;
145                reg_done           = 1'b0;
146             end
147
148          ITEST:
149             begin
150                reg_reset_registers     = 1'b0;
151                reg_m_reset           = 1'b1;
152                reg_def_enable        = 1'b0;
153                reg_refs_enable          = 1'b0;
154                reg_shift_enable      = 1'b0;
155                reg_i_index_enable       = 1'b1;
156                reg_m_index_enable       = 1'b0;
157                reg_indexes_enable       = 1'b0;
158                reg_displacement_enable = 1'b0;
159
160                reg_part_done         = 1'b0;
161                reg_done           = 1'b0;
162             end
163
164          UPDATE:
165             begin
166                reg_reset_registers     = 1'b0;
167                reg_m_reset           = 1'b0;
168                reg_def_enable        = 1'b0;
169                reg_refs_enable          = 1'b0;
170                reg_shift_enable      = 1'b0;
171                reg_i_index_enable       = 1'b0;
172                reg_m_index_enable       = 1'b0;
173                reg_indexes_enable       = 1'b1;
174                reg_displacement_enable = 1'b0;
175
176                reg_part_done         = 1'b1;
177                reg_done           = 1'b0;
178             end
179
180          DONE:
181             begin
182                reg_reset_registers     = 1'b0;
183                reg_m_reset           = 1'b0;
184                reg_def_enable        = 1'b0;
185                reg_refs_enable          = 1'b0;
186                reg_shift_enable      = 1'b0;
187                reg_i_index_enable       = 1'b0;
188                reg_m_index_enable       = 1'b0;
189                reg_indexes_enable       = 1'b0;
190                reg_displacement_enable = 1'b0;
191
192                reg_part_done         = 1'b1;
193                reg_done           = 1'b1;
194             end
195
196          default:
197             begin
198                reg_reset_registers     = 1'b0;
199                reg_m_reset           = 1'b0;
200                reg_def_enable        = 1'b0;
```

```
201                reg_refs_enable          = 1'b0;
202                reg_shift_enable     = 1'b0;
203                reg_i_index_enable       = 1'b0;
204                reg_m_index_enable       = 1'b0;
205                reg_indexes_enable       = 1'b0;
206                reg_displacement_enable = 1'b0;
207
208                reg_part_done        = 1'b0;
209                reg_done           = 1'b0;
210             end
211          endcase
212       end
213
214       always @(posedge clock)
215       begin
216          if (reset)
217             state <= INIT;
218          else
219          begin
220             case (state)
221
222                INIT:
223                   begin
224                      if (start)
225                         state <= PRE_LOAD;
226                      else
227                         state <= INIT;
228                   end
229
230                PRE_LOAD:
231                   begin
232                      state <= TIME1;
233                   end
234
235                TIME1:
236                   begin
237                      state <= LOAD;
238                   end
239
240                LOAD:
241                   begin
242                      state <= TIME2;
243                   end
244
245                TIME2:
246                   begin
247                      state <= CALC;
248                   end
249
250                CALC:
251                   begin
252                      if (m_condition)
253                      begin
254                         state <= TIME1;
255                      end
256                      else
257                      begin
```

```
258                     state <= ITEST;
259                  end
260               end
261
262            ITEST:
263               begin
264                  if (i_condition)
265                     state <= UPDATE;
266                  else
267                     state <= DONE;
268               end
269
270            UPDATE:
271               begin
272                  state <= PRE_LOAD;
273               end
274
275            DONE:
276               begin
277                  state <= DONE;
278               end
279
280            default:
281               begin
282                  state <= INIT;
283               end
284         endcase
285      end
286   end
287
288   assign reset_registers     = reg_reset_registers;
289   assign m_reset             = reg_m_reset;
290   assign def_enable          = reg_def_enable;
291   assign refs_enable         = reg_refs_enable;
292   assign shift_enable        = reg_shift_enable;
293   assign i_index_enable      = reg_i_index_enable;
294   assign m_index_enable      = reg_m_index_enable;
295   assign indexes_enable      = reg_indexes_enable;
296   assign displacement_enable = reg_displacement_enable;
297   assign part_done           = reg_part_done;
298   assign done                = reg_done;
299
300 endmodule
```

Listing B.10 – Código fonte do bloco operativo do preditor angular

```
1
2  module angular_operative
3
4     #(parameter WIDTH = 8, parameter MODE = 2)
5
6     (
7        input          clock,
8                       reset,
9                       m_reset,
10                      def_enable,
11                      refs_enable,
```

```
12                      shift_enable,
13                      i_index_enable,
14                      m_index_enable,
15                      indexes_enable,
16                      displacement_enable,
17
18      input[WIDTH-1:0]  ref1,
19                      ref2,
20                      ref3,
21                      ref4,
22                      ref5,
23
24      input[WIDTH-3:0]  N,
25
26      output             m_condition,
27                      i_condition,
28
29      output signed [WIDTH-1:0]  samples,
30
31      output[WIDTH-1:0] p00,
32                      p01,
33                      p02,
34                      p03,
35                      p10,
36                      p11,
37                      p12,
38                      p13,
39                      p20,
40                      p21,
41                      p22,
42                      p23,
43                      p30,
44                      p31,
45                      p32,
46                      p33
47  );
48
49  reg reg_signal;
50
51  reg[WIDTH-6:0] reg_m;
52
53  reg[WIDTH-3:0] reg_x_mask,
54                 reg_y_mask,
55                 reg_N;
56
57  reg[WIDTH-2:0] reg_i,
58                 reg_k,
59                 reg_l,
60                 reg_iterations;
61
62  reg[WIDTH-1:0] reg_ref1,
63                 reg_ref2,
64                 reg_ref3,
65                 reg_ref4,
66                 reg_ref5,
67                 reg_angle;
68
```

```
69      reg [WIDTH+4:0]   reg_integer_disp,
70                        reg_fractional_disp;
71
72      wire[WIDTH-6:0] m;
73
74      wire[WIDTH-2:0] i,
75                      k,
76                      l;
77
78      wire[WIDTH-1:0]   pn0,
79                        pn1,
80                        pn2,
81                        pn3;
82
83      wire[WIDTH+4:0]   integer_disp,
84                        fractional_disp;
85
86      angular_shift_buffer #(WIDTH) angular_shift_buffer_block(
87         .clock(clock),
88         .reset(reset),
89         .shift_enable(shift_enable),
90         .pn0(pn0),
91         .pn1(pn1),
92         .pn2(pn2),
93         .pn3(pn3),
94
95         .p00(p00),
96         .p01(p01),
97         .p02(p02),
98         .p03(p03),
99         .p10(p10),
100        .p11(p11),
101        .p12(p12),
102        .p13(p13),
103        .p20(p20),
104        .p21(p21),
105        .p22(p22),
106        .p23(p23),
107        .p30(p30),
108        .p31(p31),
109        .p32(p32),
110        .p33(p33)
111     );
112
113     always @(posedge clock)
114     begin
115
116        if (reset)
117        begin
118           reg_N          <= 6'b000000;
119           reg_i          <= 7'b0000000;
120           reg_k          <= 7'b0000000;
121           reg_l          <= 7'b0000000;
122           reg_m          <= 3'b000;
123           reg_angle         <= 8'b00000000;
124           reg_x_mask        <= 6'b000000;
125           reg_y_mask        <= 6'b000000;
```

```
126           reg_iterations    <= 7'b0000000;
127           reg_integer_disp  <= 13'b0000000000000;
128           reg_fractional_disp  <= 13'b0000000000000;
129           reg_ref1       <= 8'b00000000;
130           reg_ref2       <= 8'b00000000;
131           reg_ref3       <= 8'b00000000;
132           reg_ref4       <= 8'b00000000;
133           reg_ref5       <= 8'b00000000;
134           reg_signal        <= (MODE > 10 && MODE < 26) ? 1'b1 : 1'b0;
135
136           case (MODE)
137              2:  begin reg_angle <= 8'b00100000; end
138              3:  begin reg_angle <= 8'b00011010; end
139              4:  begin reg_angle <= 8'b00010101; end
140              5:  begin reg_angle <= 8'b00010001; end
141              6:  begin reg_angle <= 8'b00001101; end
142              7:  begin reg_angle <= 8'b00001001; end
143              8:  begin reg_angle <= 8'b00000101; end
144              9:  begin reg_angle <= 8'b00000010; end
145              10: begin reg_angle <= 8'b00000000; end
146              11: begin reg_angle <= 8'b00000010; end
147              12: begin reg_angle <= 8'b00000101; end
148              13: begin reg_angle <= 8'b00001001; end
149              14: begin reg_angle <= 8'b00001101; end
150              15: begin reg_angle <= 8'b00010001; end
151              16: begin reg_angle <= 8'b00010101; end
152              17: begin reg_angle <= 8'b00011010; end
153              18: begin reg_angle <= 8'b00100000; end
154              19: begin reg_angle <= 8'b00011010; end
155              20: begin reg_angle <= 8'b00010101; end
156              21: begin reg_angle <= 8'b00010001; end
157              22: begin reg_angle <= 8'b00001101; end
158              23: begin reg_angle <= 8'b00001001; end
159              24: begin reg_angle <= 8'b00000101; end
160              25: begin reg_angle <= 8'b00000010; end
161              26: begin reg_angle <= 8'b00000000; end
162              27: begin reg_angle <= 8'b00000010; end
163              28: begin reg_angle <= 8'b00000101; end
164              29: begin reg_angle <= 8'b00001001; end
165              30: begin reg_angle <= 8'b00001101; end
166              31: begin reg_angle <= 8'b00010001; end
167              32: begin reg_angle <= 8'b00010101; end
168              33: begin reg_angle <= 8'b00011010; end
169              34: begin reg_angle <= 8'b00100000; end
170              default: begin reg_angle <= 8'b00000000; end
171           endcase
172        end
173        else
174        begin
175
176           if (def_enable)
177           begin
178              reg_N <= N;
179
180              case (N)
181                 4: begin
182                       reg_iterations <= 7'b0000001;
```

```
183                     reg_x_mask      <= 6'b000000;
184                     reg_y_mask      <= 6'b000000;
185                  end
186
187               8: begin
188                     reg_iterations <= 7'b0000100;
189                     reg_x_mask      <= 6'b000111;
190                     reg_y_mask      <= 6'b000001;
191                  end
192
193               16:   begin
194                     reg_iterations <= 7'b0010000;
195                     reg_x_mask      <= 6'b001111;
196                     reg_y_mask      <= 6'b000010;
197                  end
198
199               32:   begin
200                     reg_iterations <= 7'b1000000;
201                     reg_x_mask      <= 6'b011111;
202                     reg_y_mask      <= 6'b000011;
203                  end
204               default: begin
205                     reg_iterations <= 7'b0000001;
206                     reg_x_mask      <= 6'b000000;
207                     reg_y_mask      <= 6'b000000;
208                  end
209            endcase
210         end
211
212         if (refs_enable)
213         begin
214            reg_ref1 <= ref1;
215            reg_ref2 <= ref2;
216            reg_ref3 <= ref3;
217            reg_ref4 <= ref4;
218            reg_ref5 <= ref5;
219         end
220
221         if (i_index_enable)
222         begin
223            reg_i <= i;
224         end
225
226         if (m_reset)
227         begin
228            reg_m <= 0;
229         end
230         else
231         begin
232            if (m_index_enable)
233            begin
234               reg_m <= m;
235            end
236         end
237
238         if (indexes_enable)
239         begin
```

```
240             reg_k <= k;
241             reg_l <= l;
242          end
243
244          if (displacement_enable)
245          begin
246             reg_integer_disp  <= integer_disp;
247             reg_fractional_disp  <= fractional_disp;
248          end
249       end
250    end
251
252    assign m          = reg_m + 1;
253    assign i          = reg_i + 1;
254
255    assign k          = reg_m + ((reg_i << 2) & reg_x_mask);
256    assign l          = (reg_i >> reg_y_mask) << 2;
257
258    assign integer_disp     = ((reg_k + 1) * reg_angle) >> 5;
259    assign fractional_disp  = ((reg_k + 1) * reg_angle) & 31;
260
261    assign pn0           = reg_signal   ? ((32 + reg_fractional_disp) * reg_ref1
          - reg_fractional_disp * reg_ref2 + 16) >> 5
262                                 : ((32 - reg_fractional_disp) * reg_ref1 +
                                       reg_fractional_disp * reg_ref2 + 16) >> 5;
263    assign pn1           = reg_signal   ? ((32 + reg_fractional_disp) * reg_ref2
          - reg_fractional_disp * reg_ref3 + 16) >> 5
264                                 : ((32 - reg_fractional_disp) * reg_ref2 +
                                       reg_fractional_disp * reg_ref3 + 16) >> 5;
265    assign pn2           = reg_signal   ? ((32 + reg_fractional_disp) * reg_ref3
          - reg_fractional_disp * reg_ref4 + 16) >> 5
266                                 : ((32 - reg_fractional_disp) * reg_ref3 +
                                       reg_fractional_disp * reg_ref4 + 16) >> 5;
267    assign pn3           = reg_signal   ? ((32 + reg_fractional_disp) * reg_ref4
          - reg_fractional_disp * reg_ref5 + 16) >> 5
268                                 : ((32 - reg_fractional_disp) * reg_ref4 +
                                       reg_fractional_disp * reg_ref5 + 16) >> 5;
269
270    assign i_condition      = reg_i < reg_iterations - 1;
271    assign m_condition      = reg_m < 4;
272
273    assign samples = reg_signal ? reg_l - reg_integer_disp : reg_l +
          reg_integer_disp;
274
275 endmodule
```

Listing B.11 – Código fonte do seletor de amostras do preditor angular

```
1
2  module angular_refs_selector
3
4     #(parameter WIDTH = 8, parameter MODE = 2)
5
6     (
7        input clock,
8
9        input signed [WIDTH-1:0]   selector,
```

```
10
11        input[WIDTH-1:0]  sample_0n,  sample_32n, sample_n0,  sample_n32,
12                          sample_1n,  sample_33n, sample_n1,  sample_n33,
13                          sample_2n,  sample_34n, sample_n2,  sample_n34,
14                          sample_3n,  sample_35n, sample_n3,  sample_n35,
15                          sample_4n,  sample_36n, sample_n4,  sample_n36,
16                          sample_5n,  sample_37n, sample_n5,  sample_n37,
17                          sample_6n,  sample_38n, sample_n6,  sample_n38,
18                          sample_7n,  sample_39n, sample_n7,  sample_n39,
19                          sample_8n,  sample_40n, sample_n8,  sample_n40,
20                          sample_9n,  sample_41n, sample_n9,  sample_n41,
21                          sample_10n, sample_42n, sample_n10, sample_n42,
22                          sample_11n, sample_43n, sample_n11, sample_n43,
23                          sample_12n, sample_44n, sample_n12, sample_n44,
24                          sample_13n, sample_45n, sample_n13, sample_n45,
25                          sample_14n, sample_46n, sample_n14, sample_n46,
26                          sample_15n, sample_47n, sample_n15, sample_n47,
27                          sample_16n, sample_48n, sample_n16, sample_n48,
28                          sample_17n, sample_49n, sample_n17, sample_n49,
29                          sample_18n, sample_50n, sample_n18, sample_n50,
30                          sample_19n, sample_51n, sample_n19, sample_n51,
31                          sample_20n, sample_52n, sample_n20, sample_n52,
32                          sample_21n, sample_53n, sample_n21, sample_n53,
33                          sample_22n, sample_54n, sample_n22, sample_n54,
34                          sample_23n, sample_55n, sample_n23, sample_n55,
35                          sample_24n, sample_56n, sample_n24, sample_n56,
36                          sample_25n, sample_57n, sample_n25, sample_n57,
37                          sample_26n, sample_58n, sample_n26, sample_n58,
38                          sample_27n, sample_59n, sample_n27, sample_n59,
39                          sample_28n, sample_60n, sample_n28, sample_n60,
40                          sample_29n, sample_61n, sample_n29, sample_n61,
41                          sample_30n, sample_62n, sample_n30, sample_n62,
42                          sample_31n, sample_63n, sample_n31, sample_n63,
43
44                          sample_NN,
45
46        output[WIDTH-1:0] ref1,
47                          ref2,
48                          ref3,
49                          ref4,
50                          ref5
51    );
52
53    assign ref1 = MODE < 18 ? ( (selector == -32 && (MODE == 15 || MODE == 21)) ?
          sample_57n
54                          : ( (selector == -32 && (MODE == 16 || MODE == 20)) ?
                                sample_46n
55                          : ( (selector == -32 && (MODE == 17 || MODE == 19)) ?
                                sample_37n
56                          : ( (selector == -32 &&  MODE == 18)            ?
                                sample_30n
57                          : ( (selector == -31 && (MODE == 15 || MODE == 21)) ?
                                sample_55n
58                          : ( (selector == -31 && (MODE == 16 || MODE == 20)) ?
                                sample_45n
59                          : ( (selector == -31 && (MODE == 17 || MODE == 19)) ?
                                sample_36n
```

```
60                      : ( (selector == -31 &&  MODE == 18)            ?
                          sample_29n
61                      : ( (selector == -30 && (MODE == 15 || MODE == 21)) ?
                          sample_54n
62                      : ( (selector == -30 && (MODE == 16 || MODE == 20)) ?
                          sample_43n
63                      : ( (selector == -30 && (MODE == 17 || MODE == 19)) ?
                          sample_35n
64                      : ( (selector == -30 &&  MODE == 18)            ?
                          sample_28n
65                      : ( (selector == -29 && (MODE == 15 || MODE == 21)) ?
                          sample_52n
66                      : ( (selector == -29 && (MODE == 16 || MODE == 20)) ?
                          sample_42n
67                      : ( (selector == -29 && (MODE == 17 || MODE == 19)) ?
                          sample_33n
68                      : ( (selector == -29 &&  MODE == 18)            ?
                          sample_27n
69                      : ( (selector == -28 && (MODE == 15 || MODE == 21)) ?
                          sample_50n
70                      : ( (selector == -28 && (MODE == 16 || MODE == 20)) ?
                          sample_40n
71                      : ( (selector == -28 && (MODE == 17 || MODE == 19)) ?
                          sample_32n
72                      : ( (selector == -28 &&  MODE == 18)            ?
                          sample_26n
73                      : ( (selector == -27 && (MODE == 14 || MODE == 22)) ?
                          sample_63n
74                      : ( (selector == -27 && (MODE == 15 || MODE == 21)) ?
                          sample_48n
75                      : ( (selector == -27 && (MODE == 16 || MODE == 20)) ?
                          sample_39n
76                      : ( (selector == -27 && (MODE == 17 || MODE == 19)) ?
                          sample_31n
77                      : ( (selector == -27 &&  MODE == 18)            ?
                          sample_25n
78                      : ( (selector == -26 && (MODE == 14 || MODE == 22)) ?
                          sample_61n
79                      : ( (selector == -26 && (MODE == 15 || MODE == 21)) ?
                          sample_46n
80                      : ( (selector == -26 && (MODE == 16 || MODE == 20)) ?
                          sample_37n
81                      : ( (selector == -26 && (MODE == 17 || MODE == 19)) ?
                          sample_30n
82                      : ( (selector == -26 &&  MODE == 18)            ?
                          sample_24n
83                      : ( (selector == -25 && (MODE == 14 || MODE == 22)) ?
                          sample_58n
84                      : ( (selector == -25 && (MODE == 15 || MODE == 21)) ?
                          sample_44n
85                      : ( (selector == -25 && (MODE == 16 || MODE == 20)) ?
                          sample_36n
86                      : ( (selector == -25 && (MODE == 17 || MODE == 19)) ?
                          sample_29n
87                      : ( (selector == -25 &&  MODE == 18)            ?
                          sample_23n
88                      : ( (selector == -24 && (MODE == 14 || MODE == 22)) ?
```

```
                    sample_56n
89                : ( (selector == -24 && (MODE == 15 || MODE == 21)) ?
                    sample_42n
90                : ( (selector == -24 && (MODE == 16 || MODE == 20)) ?
                    sample_34n
91                : ( (selector == -24 && (MODE == 17 || MODE == 19)) ?
                    sample_27n
92                : ( (selector == -24 &&  MODE == 18)            ?
                    sample_22n
93                : ( (selector == -23 && (MODE == 14 || MODE == 22)) ?
                    sample_53n
94                : ( (selector == -23 && (MODE == 15 || MODE == 21)) ?
                    sample_40n
95                : ( (selector == -23 && (MODE == 16 || MODE == 20)) ?
                    sample_33n
96                : ( (selector == -23 && (MODE == 17 || MODE == 19)) ?
                    sample_26n
97                : ( (selector == -23 &&  MODE == 18)            ?
                    sample_21n
98                : ( (selector == -22 && (MODE == 14 || MODE == 22)) ?
                    sample_51n
99                : ( (selector == -22 && (MODE == 15 || MODE == 21)) ?
                    sample_39n
100               : ( (selector == -22 && (MODE == 16 || MODE == 20)) ?
                    sample_31n
101               : ( (selector == -22 && (MODE == 17 || MODE == 19)) ?
                    sample_25n
102               : ( (selector == -22 &&  MODE == 18)            ?
                    sample_20n
103               : ( (selector == -21 && (MODE == 14 || MODE == 22)) ?
                    sample_48n
104               : ( (selector == -21 && (MODE == 15 || MODE == 21)) ?
                    sample_37n
105               : ( (selector == -21 && (MODE == 16 || MODE == 20)) ?
                    sample_29n
106               : ( (selector == -21 && (MODE == 17 || MODE == 19)) ?
                    sample_24n
107               : ( (selector == -21 &&  MODE == 18)            ?
                    sample_19n
108               : ( (selector == -20 && (MODE == 14 || MODE == 22)) ?
                    sample_46n
109               : ( (selector == -20 && (MODE == 15 || MODE == 21)) ?
                    sample_35n
110               : ( (selector == -20 && (MODE == 16 || MODE == 20)) ?
                    sample_28n
111               : ( (selector == -20 && (MODE == 17 || MODE == 19)) ?
                    sample_22n
112               : ( (selector == -20 &&  MODE == 18)            ?
                    sample_18n
113               : ( (selector == -19 && (MODE == 13 || MODE == 23)) ?
                    sample_63n
114               : ( (selector == -19 && (MODE == 14 || MODE == 22)) ?
                    sample_43n
115               : ( (selector == -19 && (MODE == 15 || MODE == 21)) ?
                    sample_33n
116               : ( (selector == -19 && (MODE == 16 || MODE == 20)) ?
                    sample_26n
```

```
117                     : ( (selector == -19 && (MODE == 17 || MODE == 19)) ?
                            sample_21n
118                     : ( (selector == -19 &&  MODE == 18)            ?
                            sample_17n
119                     : ( (selector == -18 && (MODE == 13 || MODE == 23)) ?
                            sample_59n
120                     : ( (selector == -18 && (MODE == 14 || MODE == 22)) ?
                            sample_41n
121                     : ( (selector == -18 && (MODE == 15 || MODE == 21)) ?
                            sample_31n
122                     : ( (selector == -18 && (MODE == 16 || MODE == 20)) ?
                            sample_25n
123                     : ( (selector == -18 && (MODE == 17 || MODE == 19)) ?
                            sample_20n
124                     : ( (selector == -18 &&  MODE == 18)            ?
                            sample_16n
125                     : ( (selector == -17 && (MODE == 13 || MODE == 23)) ?
                            sample_56n
126                     : ( (selector == -17 && (MODE == 14 || MODE == 22)) ?
                            sample_38n
127                     : ( (selector == -17 && (MODE == 15 || MODE == 21)) ?
                            sample_29n
128                     : ( (selector == -17 && (MODE == 16 || MODE == 20)) ?
                            sample_23n
129                     : ( (selector == -17 && (MODE == 17 || MODE == 19)) ?
                            sample_19n
130                     : ( (selector == -17 &&  MODE == 18)            ?
                            sample_15n
131                     : ( (selector == -16 && (MODE == 13 || MODE == 23)) ?
                            sample_52n
132                     : ( (selector == -16 && (MODE == 14 || MODE == 22)) ?
                            sample_36n
133                     : ( (selector == -16 && (MODE == 15 || MODE == 21)) ?
                            sample_27n
134                     : ( (selector == -16 && (MODE == 16 || MODE == 20)) ?
                            sample_22n
135                     : ( (selector == -16 && (MODE == 17 || MODE == 19)) ?
                            sample_17n
136                     : ( (selector == -16 &&  MODE == 18)            ?
                            sample_14n
137                     : ( (selector == -15 && (MODE == 13 || MODE == 23)) ?
                            sample_49n
138                     : ( (selector == -15 && (MODE == 14 || MODE == 22)) ?
                            sample_33n
139                     : ( (selector == -15 && (MODE == 15 || MODE == 21)) ?
                            sample_25n
140                     : ( (selector == -15 && (MODE == 16 || MODE == 20)) ?
                            sample_20n
141                     : ( (selector == -15 && (MODE == 17 || MODE == 19)) ?
                            sample_16n
142                     : ( (selector == -15 &&  MODE == 18)            ?
                            sample_13n
143                     : ( (selector == -14 && (MODE == 13 || MODE == 23)) ?
                            sample_45n
144                     : ( (selector == -14 && (MODE == 14 || MODE == 22)) ?
                            sample_31n
145                     : ( (selector == -14 && (MODE == 15 || MODE == 21)) ?
```

```
                    sample_23n
146                 : ( (selector == -14 && (MODE == 16 || MODE == 20)) ?
                    sample_19n
147                 : ( (selector == -14 && (MODE == 17 || MODE == 19)) ?
                    sample_15n
148                 : ( (selector == -14 &&  MODE == 18)            ?
                    sample_12n
149                 : ( (selector == -13 && (MODE == 13 || MODE == 23)) ?
                    sample_42n
150                 : ( (selector == -13 && (MODE == 14 || MODE == 22)) ?
                    sample_29n
151                 : ( (selector == -13 && (MODE == 15 || MODE == 21)) ?
                    sample_22n
152                 : ( (selector == -13 && (MODE == 16 || MODE == 20)) ?
                    sample_17n
153                 : ( (selector == -13 && (MODE == 17 || MODE == 19)) ?
                    sample_14n
154                 : ( (selector == -13 &&  MODE == 18)            ?
                    sample_11n
155                 : ( (selector == -12 && (MODE == 13 || MODE == 23)) ?
                    sample_38n
156                 : ( (selector == -12 && (MODE == 14 || MODE == 22)) ?
                    sample_26n
157                 : ( (selector == -12 && (MODE == 15 || MODE == 21)) ?
                    sample_20n
158                 : ( (selector == -12 && (MODE == 16 || MODE == 20)) ?
                    sample_16n
159                 : ( (selector == -12 && (MODE == 17 || MODE == 19)) ?
                    sample_13n
160                 : ( (selector == -12 &&  MODE == 18)            ?
                    sample_10n
161                 : ( (selector == -11 && (MODE == 12 || MODE == 24)) ?
                    sample_63n
162                 : ( (selector == -11 && (MODE == 13 || MODE == 23)) ?
                    sample_35n
163                 : ( (selector == -11 && (MODE == 14 || MODE == 22)) ?
                    sample_24n
164                 : ( (selector == -11 && (MODE == 15 || MODE == 21)) ?
                    sample_18n
165                 : ( (selector == -11 && (MODE == 16 || MODE == 20)) ?
                    sample_14n
166                 : ( (selector == -11 && (MODE == 17 || MODE == 19)) ?
                    sample_11n
167                 : ( (selector == -11 &&  MODE == 18)            ? sample_9n
168                 : ( (selector == -10 && (MODE == 12 || MODE == 24)) ?
                    sample_57n
169                 : ( (selector == -10 && (MODE == 13 || MODE == 23)) ?
                    sample_31n
170                 : ( (selector == -10 && (MODE == 14 || MODE == 22)) ?
                    sample_21n
171                 : ( (selector == -10 && (MODE == 15 || MODE == 21)) ?
                    sample_16n
172                 : ( (selector == -10 && (MODE == 16 || MODE == 20)) ?
                    sample_13n
173                 : ( (selector == -10 && (MODE == 17 || MODE == 19)) ?
                    sample_10n
174                 : ( (selector == -10 &&  MODE == 18)            ? sample_8n
```

```
175                     : ( (selector ==  -9 && (MODE == 12 || MODE == 24)) ?
                            sample_50n
176                     : ( (selector ==  -9 && (MODE == 13 || MODE == 23)) ?
                            sample_27n
177                     : ( (selector ==  -9 && (MODE == 14 || MODE == 22)) ?
                            sample_19n
178                     : ( (selector ==  -9 && (MODE == 15 || MODE == 21)) ?
                            sample_14n
179                     : ( (selector ==  -9 && (MODE == 16 || MODE == 20)) ?
                            sample_11n
180                     : ( (selector ==  -9 && (MODE == 17 || MODE == 19)) ?
                            sample_9n
181                     : ( (selector ==  -9 &&  MODE == 18)            ? sample_7n
182                     : ( (selector ==  -8 && (MODE == 12 || MODE == 24)) ?
                            sample_44n
183                     : ( (selector ==  -8 && (MODE == 13 || MODE == 23)) ?
                            sample_24n
184                     : ( (selector ==  -8 && (MODE == 14 || MODE == 22)) ?
                            sample_16n
185                     : ( (selector ==  -8 && (MODE == 15 || MODE == 21)) ?
                            sample_12n
186                     : ( (selector ==  -8 && (MODE == 16 || MODE == 20)) ?
                            sample_10n
187                     : ( (selector ==  -8 && (MODE == 17 || MODE == 19)) ?
                            sample_8n
188                     : ( (selector ==  -8 &&  MODE == 18)            ? sample_6n
189                     : ( (selector ==  -7 && (MODE == 12 || MODE == 24)) ?
                            sample_37n
190                     : ( (selector ==  -7 && (MODE == 13 || MODE == 23)) ?
                            sample_20n
191                     : ( (selector ==  -7 && (MODE == 14 || MODE == 22)) ?
                            sample_14n
192                     : ( (selector ==  -7 && (MODE == 15 || MODE == 21)) ?
                            sample_10n
193                     : ( (selector ==  -7 && (MODE == 16 || MODE == 20)) ?
                            sample_8n
194                     : ( (selector ==  -7 && (MODE == 17 || MODE == 19)) ?
                            sample_6n
195                     : ( (selector ==  -7 &&  MODE == 18)            ? sample_5n
196                     : ( (selector ==  -6 && (MODE == 12 || MODE == 24)) ?
                            sample_31n
197                     : ( (selector ==  -6 && (MODE == 13 || MODE == 23)) ?
                            sample_17n
198                     : ( (selector ==  -6 && (MODE == 14 || MODE == 22)) ?
                            sample_11n
199                     : ( (selector ==  -6 && (MODE == 15 || MODE == 21)) ?
                            sample_8n
200                     : ( (selector ==  -6 && (MODE == 16 || MODE == 20)) ?
                            sample_7n
201                     : ( (selector ==  -6 && (MODE == 17 || MODE == 19)) ?
                            sample_5n
202                     : ( (selector ==  -6 &&  MODE == 18)            ? sample_4n
203                     : ( (selector ==  -5 && (MODE == 11 || MODE == 25)) ?
                            sample_63n
204                     : ( (selector ==  -5 && (MODE == 12 || MODE == 24)) ?
                            sample_25n
205                     : ( (selector ==  -5 && (MODE == 13 || MODE == 23)) ?
```

```
                    sample_13n
206                 : ( (selector ==  -5 && (MODE == 14 || MODE == 22)) ?
                    sample_9n
207                 : ( (selector ==  -5 && (MODE == 15 || MODE == 21)) ?
                    sample_7n
208                 : ( (selector ==  -5 && (MODE == 16 || MODE == 20)) ?
                    sample_5n
209                 : ( (selector ==  -5 && (MODE == 17 || MODE == 19)) ?
                    sample_4n
210                 : ( (selector ==  -5 &&  MODE == 18)            ? sample_3n
211                 : ( (selector ==  -4 && (MODE == 11 || MODE == 25)) ?
                    sample_47n
212                 : ( (selector ==  -4 && (MODE == 12 || MODE == 24)) ?
                    sample_18n
213                 : ( (selector ==  -4 && (MODE == 13 || MODE == 23)) ?
                    sample_10n
214                 : ( (selector ==  -4 && (MODE == 14 || MODE == 22)) ?
                    sample_6n
215                 : ( (selector ==  -4 && (MODE == 15 || MODE == 21)) ?
                    sample_5n
216                 : ( (selector ==  -4 && (MODE == 16 || MODE == 20)) ?
                    sample_4n
217                 : ( (selector ==  -4 && (MODE == 17 || MODE == 19)) ?
                    sample_3n
218                 : ( (selector ==  -4 &&  MODE == 18)            ? sample_2n
219                 : ( (selector ==  -3 && (MODE == 11 || MODE == 25)) ?
                    sample_31n
220                 : ( (selector ==  -3 && (MODE == 12 || MODE == 24)) ?
                    sample_12n
221                 : ( (selector ==  -3 && (MODE == 13 || MODE == 23)) ?
                    sample_6n
222                 : ( (selector ==  -3 && (MODE == 14 || MODE == 22)) ?
                    sample_4n
223                 : ( (selector ==  -3 && (MODE == 15 || MODE == 21)) ?
                    sample_3n
224                 : ( (selector ==  -3 && (MODE == 16 || MODE == 20)) ?
                    sample_2n
225                 : ( (selector ==  -3 && (MODE == 17 || MODE == 19)) ?
                    sample_1n
226                 : ( (selector ==  -3 &&  MODE == 18)            ? sample_1n
227                 : ( (selector ==  -2 && (MODE == 11 || MODE == 25)) ?
                    sample_15n
228                 : ( (selector ==  -2 && (MODE == 12 || MODE == 24)) ?
                    sample_5n
229                 : ( (selector ==  -2 && (MODE == 13 || MODE == 23)) ?
                    sample_3n
230                 : ( (selector ==  -2 && (MODE == 14 || MODE == 22)) ?
                    sample_1n
231                 : ( (selector ==  -2 && (MODE == 15 || MODE == 21)) ?
                    sample_1n
232                 : ( (selector ==  -2 && (MODE == 16 || MODE == 20)) ?
                    sample_1n
233                 : ( (selector ==  -2 && (MODE == 17 || MODE == 19)) ?
                    sample_0n
234                 : ( (selector ==  -2 &&  MODE == 18)            ? sample_0n
235                 : (selector ==  -1 ? sample_NN
236                 : (selector ==   0 ? sample_n0
```

```
237                         : (selector ==   1 ? sample_n1
238                         : (selector ==   2 ? sample_n2
239                         : (selector ==   3 ? sample_n3
240                         : (selector ==   4 ? sample_n4
241                         : (selector ==   5 ? sample_n5
242                         : (selector ==   6 ? sample_n6
243                         : (selector ==   7 ? sample_n7
244                         : (selector ==   8 ? sample_n8
245                         : (selector ==   9 ? sample_n9
246                         : (selector ==  10 ? sample_n10
247                         : (selector ==  11 ? sample_n11
248                         : (selector ==  12 ? sample_n12
249                         : (selector ==  13 ? sample_n13
250                         : (selector ==  14 ? sample_n14
251                         : (selector ==  15 ? sample_n15
252                         : (selector ==  16 ? sample_n16
253                         : (selector ==  17 ? sample_n17
254                         : (selector ==  18 ? sample_n18
255                         : (selector ==  19 ? sample_n19
256                         : (selector ==  20 ? sample_n20
257                         : (selector ==  21 ? sample_n21
258                         : (selector ==  22 ? sample_n22
259                         : (selector ==  23 ? sample_n23
260                         : (selector ==  24 ? sample_n24
261                         : (selector ==  25 ? sample_n25
262                         : (selector ==  26 ? sample_n26
263                         : (selector ==  27 ? sample_n27
264                         : (selector ==  28 ? sample_n28
265                         : (selector ==  29 ? sample_n29
266                         : (selector ==  30 ? sample_n30
267                         : (selector ==  31 ? sample_n31
268                         : (selector ==  32 ? sample_n32
269                         : (selector ==  33 ? sample_n33
270                         : (selector ==  34 ? sample_n34
271                         : (selector ==  35 ? sample_n35
272                         : (selector ==  36 ? sample_n36
273                         : (selector ==  37 ? sample_n37
274                         : (selector ==  38 ? sample_n38
275                         : (selector ==  39 ? sample_n39
276                         : (selector ==  40 ? sample_n40
277                         : (selector ==  41 ? sample_n41
278                         : (selector ==  42 ? sample_n42
279                         : (selector ==  43 ? sample_n43
280                         : (selector ==  44 ? sample_n44
281                         : (selector ==  45 ? sample_n45
282                         : (selector ==  46 ? sample_n46
283                         : (selector ==  47 ? sample_n47
284                         : (selector ==  48 ? sample_n48
285                         : (selector ==  49 ? sample_n49
286                         : (selector ==  50 ? sample_n50
287                         : (selector ==  51 ? sample_n51
288                         : (selector ==  52 ? sample_n52
289                         : (selector ==  53 ? sample_n53
290                         : (selector ==  54 ? sample_n54
291                         : (selector ==  55 ? sample_n55
292                         : (selector ==  56 ? sample_n56
293                         : (selector ==  57 ? sample_n57
```

```
                    : (selector ==  58 ? sample_n58
                    : (selector ==  59 ? sample_n59
                    : (selector ==  60 ? sample_n60
                    : (selector ==  61 ? sample_n61
                    : (selector ==  62 ? sample_n62
                    : (selector ==  63 ? sample_n63
                    : 0 )))))))))))))))))))))))))))))))))))))))))))))))))))))))
                      )))))))))))))))))))))))))))))))))))))))))))))))))))))))
                      )))))))))))))))))))))))))))))))))))))))))))))))))))))))
                      )))))))))))))))))))))))))))))))))))))))))))))))))))))))
                      )))))))))))))))))))))))))))
                    : ( (selector == -32 && (MODE == 15 || MODE == 21)) ?
                      sample_57n
                    : ( (selector == -32 && (MODE == 16 || MODE == 20)) ?
                      sample_n46
                    : ( (selector == -32 && (MODE == 17 || MODE == 19)) ?
                      sample_n37
                    : ( (selector == -32 &&  MODE == 18)            ?
                      sample_n30
                    : ( (selector == -31 && (MODE == 15 || MODE == 21)) ?
                      sample_n55
                    : ( (selector == -31 && (MODE == 16 || MODE == 20)) ?
                      sample_n45
                    : ( (selector == -31 && (MODE == 17 || MODE == 19)) ?
                      sample_n36
                    : ( (selector == -31 &&  MODE == 18)            ?
                      sample_n29
                    : ( (selector == -30 && (MODE == 15 || MODE == 21)) ?
                      sample_n54
                    : ( (selector == -30 && (MODE == 16 || MODE == 20)) ?
                      sample_n43
                    : ( (selector == -30 && (MODE == 17 || MODE == 19)) ?
                      sample_n35
                    : ( (selector == -30 &&  MODE == 18)            ?
                      sample_n28
                    : ( (selector == -29 && (MODE == 15 || MODE == 21)) ?
                      sample_n52
                    : ( (selector == -29 && (MODE == 16 || MODE == 20)) ?
                      sample_n42
                    : ( (selector == -29 && (MODE == 17 || MODE == 19)) ?
                      sample_n33
                    : ( (selector == -29 &&  MODE == 18)            ?
                      sample_n27
                    : ( (selector == -28 && (MODE == 15 || MODE == 21)) ?
                      sample_n50
                    : ( (selector == -28 && (MODE == 16 || MODE == 20)) ?
                      sample_n40
                    : ( (selector == -28 && (MODE == 17 || MODE == 19)) ?
                      sample_n32
                    : ( (selector == -28 &&  MODE == 18)            ?
                      sample_n26
                    : ( (selector == -27 && (MODE == 14 || MODE == 22)) ?
                      sample_n63
                    : ( (selector == -27 && (MODE == 15 || MODE == 21)) ?
                      sample_n48
                    : ( (selector == -27 && (MODE == 16 || MODE == 20)) ?
                      sample_n39
```

```
324                         : ( (selector == -27 && (MODE == 17 || MODE == 19)) ?
                              sample_n31
325                         : ( (selector == -27 &&  MODE == 18)            ?
                              sample_n25
326                         : ( (selector == -26 && (MODE == 14 || MODE == 22)) ?
                              sample_n61
327                         : ( (selector == -26 && (MODE == 15 || MODE == 21)) ?
                              sample_n46
328                         : ( (selector == -26 && (MODE == 16 || MODE == 20)) ?
                              sample_n37
329                         : ( (selector == -26 && (MODE == 17 || MODE == 19)) ?
                              sample_n30
330                         : ( (selector == -26 &&  MODE == 18)            ?
                              sample_n24
331                         : ( (selector == -25 && (MODE == 14 || MODE == 22)) ?
                              sample_n58
332                         : ( (selector == -25 && (MODE == 15 || MODE == 21)) ?
                              sample_n44
333                         : ( (selector == -25 && (MODE == 16 || MODE == 20)) ?
                              sample_n36
334                         : ( (selector == -25 && (MODE == 17 || MODE == 19)) ?
                              sample_n29
335                         : ( (selector == -25 &&  MODE == 18)            ?
                              sample_n23
336                         : ( (selector == -24 && (MODE == 14 || MODE == 22)) ?
                              sample_n56
337                         : ( (selector == -24 && (MODE == 15 || MODE == 21)) ?
                              sample_n42
338                         : ( (selector == -24 && (MODE == 16 || MODE == 20)) ?
                              sample_n34
339                         : ( (selector == -24 && (MODE == 17 || MODE == 19)) ?
                              sample_n27
340                         : ( (selector == -24 &&  MODE == 18)            ?
                              sample_n22
341                         : ( (selector == -23 && (MODE == 14 || MODE == 22)) ?
                              sample_n53
342                         : ( (selector == -23 && (MODE == 15 || MODE == 21)) ?
                              sample_n40
343                         : ( (selector == -23 && (MODE == 16 || MODE == 20)) ?
                              sample_n33
344                         : ( (selector == -23 && (MODE == 17 || MODE == 19)) ?
                              sample_n26
345                         : ( (selector == -23 &&  MODE == 18)            ?
                              sample_n21
346                         : ( (selector == -22 && (MODE == 14 || MODE == 22)) ?
                              sample_n51
347                         : ( (selector == -22 && (MODE == 15 || MODE == 21)) ?
                              sample_n39
348                         : ( (selector == -22 && (MODE == 16 || MODE == 20)) ?
                              sample_n31
349                         : ( (selector == -22 && (MODE == 17 || MODE == 19)) ?
                              sample_n25
350                         : ( (selector == -22 &&  MODE == 18)            ?
                              sample_n20
351                         : ( (selector == -21 && (MODE == 14 || MODE == 22)) ?
                              sample_n48
352                         : ( (selector == -21 && (MODE == 15 || MODE == 21)) ?
```

```
                    sample_n37
353                 : ( (selector == -21 && (MODE == 16 || MODE == 20)) ?
                    sample_n29
354                 : ( (selector == -21 && (MODE == 17 || MODE == 19)) ?
                    sample_n24
355                 : ( (selector == -21 &&  MODE == 18)            ?
                    sample_n19
356                 : ( (selector == -20 && (MODE == 14 || MODE == 22)) ?
                    sample_n46
357                 : ( (selector == -20 && (MODE == 15 || MODE == 21)) ?
                    sample_n35
358                 : ( (selector == -20 && (MODE == 16 || MODE == 20)) ?
                    sample_n28
359                 : ( (selector == -20 && (MODE == 17 || MODE == 19)) ?
                    sample_n22
360                 : ( (selector == -20 &&  MODE == 18)            ?
                    sample_n18
361                 : ( (selector == -19 && (MODE == 13 || MODE == 23)) ?
                    sample_n63
362                 : ( (selector == -19 && (MODE == 14 || MODE == 22)) ?
                    sample_n43
363                 : ( (selector == -19 && (MODE == 15 || MODE == 21)) ?
                    sample_n33
364                 : ( (selector == -19 && (MODE == 16 || MODE == 20)) ?
                    sample_n26
365                 : ( (selector == -19 && (MODE == 17 || MODE == 19)) ?
                    sample_n21
366                 : ( (selector == -19 &&  MODE == 18)            ?
                    sample_n17
367                 : ( (selector == -18 && (MODE == 13 || MODE == 23)) ?
                    sample_n59
368                 : ( (selector == -18 && (MODE == 14 || MODE == 22)) ?
                    sample_n41
369                 : ( (selector == -18 && (MODE == 15 || MODE == 21)) ?
                    sample_n31
370                 : ( (selector == -18 && (MODE == 16 || MODE == 20)) ?
                    sample_n25
371                 : ( (selector == -18 && (MODE == 17 || MODE == 19)) ?
                    sample_n20
372                 : ( (selector == -18 &&  MODE == 18)            ?
                    sample_n16
373                 : ( (selector == -17 && (MODE == 13 || MODE == 23)) ?
                    sample_n56
374                 : ( (selector == -17 && (MODE == 14 || MODE == 22)) ?
                    sample_n38
375                 : ( (selector == -17 && (MODE == 15 || MODE == 21)) ?
                    sample_n29
376                 : ( (selector == -17 && (MODE == 16 || MODE == 20)) ?
                    sample_n23
377                 : ( (selector == -17 && (MODE == 17 || MODE == 19)) ?
                    sample_n19
378                 : ( (selector == -17 &&  MODE == 18)            ?
                    sample_n15
379                 : ( (selector == -16 && (MODE == 13 || MODE == 23)) ?
                    sample_n52
380                 : ( (selector == -16 && (MODE == 14 || MODE == 22)) ?
                    sample_n36
```

```
381                         : ( (selector == -16 && (MODE == 15 || MODE == 21)) ?
                              sample_n27
382                         : ( (selector == -16 && (MODE == 16 || MODE == 20)) ?
                              sample_n22
383                         : ( (selector == -16 && (MODE == 17 || MODE == 19)) ?
                              sample_n17
384                         : ( (selector == -16 &&  MODE == 18)            ?
                              sample_n14
385                         : ( (selector == -15 && (MODE == 13 || MODE == 23)) ?
                              sample_n49
386                         : ( (selector == -15 && (MODE == 14 || MODE == 22)) ?
                              sample_n33
387                         : ( (selector == -15 && (MODE == 15 || MODE == 21)) ?
                              sample_n25
388                         : ( (selector == -15 && (MODE == 16 || MODE == 20)) ?
                              sample_n20
389                         : ( (selector == -15 && (MODE == 17 || MODE == 19)) ?
                              sample_n16
390                         : ( (selector == -15 &&  MODE == 18)            ?
                              sample_n13
391                         : ( (selector == -14 && (MODE == 13 || MODE == 23)) ?
                              sample_n45
392                         : ( (selector == -14 && (MODE == 14 || MODE == 22)) ?
                              sample_n31
393                         : ( (selector == -14 && (MODE == 15 || MODE == 21)) ?
                              sample_n23
394                         : ( (selector == -14 && (MODE == 16 || MODE == 20)) ?
                              sample_n19
395                         : ( (selector == -14 && (MODE == 17 || MODE == 19)) ?
                              sample_n15
396                         : ( (selector == -14 &&  MODE == 18)            ?
                              sample_n12
397                         : ( (selector == -13 && (MODE == 13 || MODE == 23)) ?
                              sample_n42
398                         : ( (selector == -13 && (MODE == 14 || MODE == 22)) ?
                              sample_n29
399                         : ( (selector == -13 && (MODE == 15 || MODE == 21)) ?
                              sample_n22
400                         : ( (selector == -13 && (MODE == 16 || MODE == 20)) ?
                              sample_n17
401                         : ( (selector == -13 && (MODE == 17 || MODE == 19)) ?
                              sample_n14
402                         : ( (selector == -13 &&  MODE == 18)            ?
                              sample_n11
403                         : ( (selector == -12 && (MODE == 13 || MODE == 23)) ?
                              sample_n38
404                         : ( (selector == -12 && (MODE == 14 || MODE == 22)) ?
                              sample_n26
405                         : ( (selector == -12 && (MODE == 15 || MODE == 21)) ?
                              sample_n20
406                         : ( (selector == -12 && (MODE == 16 || MODE == 20)) ?
                              sample_n16
407                         : ( (selector == -12 && (MODE == 17 || MODE == 19)) ?
                              sample_n13
408                         : ( (selector == -12 &&  MODE == 18)            ?
                              sample_n10
409                         : ( (selector == -11 && (MODE == 12 || MODE == 24)) ?
```

```
                    sample_n63
410                 : ( (selector == -11 && (MODE == 13 || MODE == 23)) ?
                    sample_n35
411                 : ( (selector == -11 && (MODE == 14 || MODE == 22)) ?
                    sample_n24
412                 : ( (selector == -11 && (MODE == 15 || MODE == 21)) ?
                    sample_n18
413                 : ( (selector == -11 && (MODE == 16 || MODE == 20)) ?
                    sample_n14
414                 : ( (selector == -11 && (MODE == 17 || MODE == 19)) ?
                    sample_n11
415                 : ( (selector == -11 &&  MODE == 18)            ? sample_n9
416                 : ( (selector == -10 && (MODE == 12 || MODE == 24)) ?
                    sample_n57
417                 : ( (selector == -10 && (MODE == 13 || MODE == 23)) ?
                    sample_n31
418                 : ( (selector == -10 && (MODE == 14 || MODE == 22)) ?
                    sample_n21
419                 : ( (selector == -10 && (MODE == 15 || MODE == 21)) ?
                    sample_n16
420                 : ( (selector == -10 && (MODE == 16 || MODE == 20)) ?
                    sample_n13
421                 : ( (selector == -10 && (MODE == 17 || MODE == 19)) ?
                    sample_n10
422                 : ( (selector == -10 &&  MODE == 18)            ? sample_n8
423                 : ( (selector ==  -9 && (MODE == 12 || MODE == 24)) ?
                    sample_n50
424                 : ( (selector ==  -9 && (MODE == 13 || MODE == 23)) ?
                    sample_n27
425                 : ( (selector ==  -9 && (MODE == 14 || MODE == 22)) ?
                    sample_n19
426                 : ( (selector ==  -9 && (MODE == 15 || MODE == 21)) ?
                    sample_n14
427                 : ( (selector ==  -9 && (MODE == 16 || MODE == 20)) ?
                    sample_n11
428                 : ( (selector ==  -9 && (MODE == 17 || MODE == 19)) ?
                    sample_n9
429                 : ( (selector ==  -9 &&  MODE == 18)            ? sample_n7
430                 : ( (selector ==  -8 && (MODE == 12 || MODE == 24)) ?
                    sample_n44
431                 : ( (selector ==  -8 && (MODE == 13 || MODE == 23)) ?
                    sample_n24
432                 : ( (selector ==  -8 && (MODE == 14 || MODE == 22)) ?
                    sample_n16
433                 : ( (selector ==  -8 && (MODE == 15 || MODE == 21)) ?
                    sample_n12
434                 : ( (selector ==  -8 && (MODE == 16 || MODE == 20)) ?
                    sample_n10
435                 : ( (selector ==  -8 && (MODE == 17 || MODE == 19)) ?
                    sample_n8
436                 : ( (selector ==  -8 &&  MODE == 18)            ? sample_n6
437                 : ( (selector ==  -7 && (MODE == 12 || MODE == 24)) ?
                    sample_n37
438                 : ( (selector ==  -7 && (MODE == 13 || MODE == 23)) ?
                    sample_n20
439                 : ( (selector ==  -7 && (MODE == 14 || MODE == 22)) ?
                    sample_n14
```

```
440                     : ( (selector ==  -7 && (MODE == 15 || MODE == 21)) ?
                            sample_n10
441                     : ( (selector ==  -7 && (MODE == 16 || MODE == 20)) ?
                            sample_n8
442                     : ( (selector ==  -7 && (MODE == 17 || MODE == 19)) ?
                            sample_n6
443                     : ( (selector ==  -7 &&  MODE == 18)            ? sample_n5
444                     : ( (selector ==  -6 && (MODE == 12 || MODE == 24)) ?
                            sample_n31
445                     : ( (selector ==  -6 && (MODE == 13 || MODE == 23)) ?
                            sample_n17
446                     : ( (selector ==  -6 && (MODE == 14 || MODE == 22)) ?
                            sample_n11
447                     : ( (selector ==  -6 && (MODE == 15 || MODE == 21)) ?
                            sample_n8
448                     : ( (selector ==  -6 && (MODE == 16 || MODE == 20)) ?
                            sample_n7
449                     : ( (selector ==  -6 && (MODE == 17 || MODE == 19)) ?
                            sample_n5
450                     : ( (selector ==  -6 &&  MODE == 18)            ? sample_n4
451                     : ( (selector ==  -5 && (MODE == 11 || MODE == 25)) ?
                            sample_n63
452                     : ( (selector ==  -5 && (MODE == 12 || MODE == 24)) ?
                            sample_n25
453                     : ( (selector ==  -5 && (MODE == 13 || MODE == 23)) ?
                            sample_n13
454                     : ( (selector ==  -5 && (MODE == 14 || MODE == 22)) ?
                            sample_n9
455                     : ( (selector ==  -5 && (MODE == 15 || MODE == 21)) ?
                            sample_n7
456                     : ( (selector ==  -5 && (MODE == 16 || MODE == 20)) ?
                            sample_n5
457                     : ( (selector ==  -5 && (MODE == 17 || MODE == 19)) ?
                            sample_n4
458                     : ( (selector ==  -5 &&  MODE == 18)            ? sample_n3
459                     : ( (selector ==  -4 && (MODE == 11 || MODE == 25)) ?
                            sample_n47
460                     : ( (selector ==  -4 && (MODE == 12 || MODE == 24)) ?
                            sample_n18
461                     : ( (selector ==  -4 && (MODE == 13 || MODE == 23)) ?
                            sample_n10
462                     : ( (selector ==  -4 && (MODE == 14 || MODE == 22)) ?
                            sample_n6
463                     : ( (selector ==  -4 && (MODE == 15 || MODE == 21)) ?
                            sample_n5
464                     : ( (selector ==  -4 && (MODE == 16 || MODE == 20)) ?
                            sample_n4
465                     : ( (selector ==  -4 && (MODE == 17 || MODE == 19)) ?
                            sample_n3
466                     : ( (selector ==  -4 &&  MODE == 18)            ? sample_n2
467                     : ( (selector ==  -3 && (MODE == 11 || MODE == 25)) ?
                            sample_n31
468                     : ( (selector ==  -3 && (MODE == 12 || MODE == 24)) ?
                            sample_n12
469                     : ( (selector ==  -3 && (MODE == 13 || MODE == 23)) ?
                            sample_n6
470                     : ( (selector ==  -3 && (MODE == 14 || MODE == 22)) ?
```

```
                        sample_n4
471                   : ( (selector ==  -3 && (MODE == 15 || MODE == 21)) ?
                        sample_n3
472                   : ( (selector ==  -3 && (MODE == 16 || MODE == 20)) ?
                        sample_n2
473                   : ( (selector ==  -3 && (MODE == 17 || MODE == 19)) ?
                        sample_n1
474                   : ( (selector ==  -3 &&  MODE == 18)            ? sample_n1
475                   : ( (selector ==  -2 && (MODE == 11 || MODE == 25)) ?
                        sample_n15
476                   : ( (selector ==  -2 && (MODE == 12 || MODE == 24)) ?
                        sample_n5
477                   : ( (selector ==  -2 && (MODE == 13 || MODE == 23)) ?
                        sample_n3
478                   : ( (selector ==  -2 && (MODE == 14 || MODE == 22)) ?
                        sample_n1
479                   : ( (selector ==  -2 && (MODE == 15 || MODE == 21)) ?
                        sample_n1
480                   : ( (selector ==  -2 && (MODE == 16 || MODE == 20)) ?
                        sample_n1
481                   : ( (selector ==  -2 && (MODE == 17 || MODE == 19)) ?
                        sample_n0
482                   : ( (selector ==  -2 &&  MODE == 18)            ? sample_n0
483                   : (selector ==  -1 ? sample_NN
484                   : (selector ==   0 ? sample_0n
485                   : (selector ==   1 ? sample_1n
486                   : (selector ==   2 ? sample_2n
487                   : (selector ==   3 ? sample_3n
488                   : (selector ==   4 ? sample_4n
489                   : (selector ==   5 ? sample_5n
490                   : (selector ==   6 ? sample_6n
491                   : (selector ==   7 ? sample_7n
492                   : (selector ==   8 ? sample_8n
493                   : (selector ==   9 ? sample_9n
494                   : (selector ==  10 ? sample_10n
495                   : (selector ==  11 ? sample_11n
496                   : (selector ==  12 ? sample_12n
497                   : (selector ==  13 ? sample_13n
498                   : (selector ==  14 ? sample_14n
499                   : (selector ==  15 ? sample_15n
500                   : (selector ==  16 ? sample_16n
501                   : (selector ==  17 ? sample_17n
502                   : (selector ==  18 ? sample_18n
503                   : (selector ==  19 ? sample_19n
504                   : (selector ==  20 ? sample_20n
505                   : (selector ==  21 ? sample_21n
506                   : (selector ==  22 ? sample_22n
507                   : (selector ==  23 ? sample_23n
508                   : (selector ==  24 ? sample_24n
509                   : (selector ==  25 ? sample_25n
510                   : (selector ==  26 ? sample_26n
511                   : (selector ==  27 ? sample_27n
512                   : (selector ==  28 ? sample_28n
513                   : (selector ==  29 ? sample_29n
514                   : (selector ==  30 ? sample_30n
515                   : (selector ==  31 ? sample_31n
516                   : (selector ==  32 ? sample_32n
```

```
                      : (selector ==  33 ? sample_33n
                      : (selector ==  34 ? sample_34n
                      : (selector ==  35 ? sample_35n
                      : (selector ==  36 ? sample_36n
                      : (selector ==  37 ? sample_37n
                      : (selector ==  38 ? sample_38n
                      : (selector ==  39 ? sample_39n
                      : (selector ==  40 ? sample_40n
                      : (selector ==  41 ? sample_41n
                      : (selector ==  42 ? sample_42n
                      : (selector ==  43 ? sample_43n
                      : (selector ==  44 ? sample_44n
                      : (selector ==  45 ? sample_45n
                      : (selector ==  46 ? sample_46n
                      : (selector ==  47 ? sample_47n
                      : (selector ==  48 ? sample_48n
                      : (selector ==  49 ? sample_49n
                      : (selector ==  50 ? sample_50n
                      : (selector ==  51 ? sample_51n
                      : (selector ==  52 ? sample_52n
                      : (selector ==  53 ? sample_53n
                      : (selector ==  54 ? sample_54n
                      : (selector ==  55 ? sample_55n
                      : (selector ==  56 ? sample_56n
                      : (selector ==  57 ? sample_57n
                      : (selector ==  58 ? sample_58n
                      : (selector ==  59 ? sample_59n
                      : (selector ==  60 ? sample_60n
                      : (selector ==  61 ? sample_61n
                      : (selector ==  62 ? sample_62n
                      : (selector ==  63 ? sample_63n
                      : 0 ))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                          ))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                          ))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                          ))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                          ))))))))))))))))))))))));

    assign ref2 = MODE < 18 ? ( (selector == -32 && (MODE == 15 || MODE == 21)) ?
        sample_55n
                      : ( (selector == -32 && (MODE == 16 || MODE == 20)) ?
                          sample_45n
                      : ( (selector == -32 && (MODE == 17 || MODE == 19)) ?
                          sample_36n
                      : ( (selector == -32 &&  MODE == 18)            ?
                          sample_29n
                      : ( (selector == -31 && (MODE == 15 || MODE == 21)) ?
                          sample_54n
                      : ( (selector == -31 && (MODE == 16 || MODE == 20)) ?
                          sample_43n
                      : ( (selector == -31 && (MODE == 17 || MODE == 19)) ?
                          sample_35n
                      : ( (selector == -31 &&  MODE == 18)            ?
                          sample_28n
                      : ( (selector == -30 && (MODE == 15 || MODE == 21)) ?
                          sample_52n
                      : ( (selector == -30 && (MODE == 16 || MODE == 20)) ?
                          sample_42n
```

```
560                         : ( (selector == -30 && (MODE == 17 || MODE == 19)) ?
                              sample_33n
561                         : ( (selector == -30 &&  MODE == 18)            ?
                              sample_27n
562                         : ( (selector == -29 && (MODE == 15 || MODE == 21)) ?
                              sample_50n
563                         : ( (selector == -29 && (MODE == 16 || MODE == 20)) ?
                              sample_40n
564                         : ( (selector == -29 && (MODE == 17 || MODE == 19)) ?
                              sample_32n
565                         : ( (selector == -29 &&  MODE == 18)            ?
                              sample_26n
566                         : ( (selector == -28 && (MODE == 14 || MODE == 22)) ?
                              sample_63n
567                         : ( (selector == -28 && (MODE == 15 || MODE == 21)) ?
                              sample_48n
568                         : ( (selector == -28 && (MODE == 16 || MODE == 20)) ?
                              sample_39n
569                         : ( (selector == -28 && (MODE == 17 || MODE == 19)) ?
                              sample_31n
570                         : ( (selector == -28 &&  MODE == 18)            ?
                              sample_25n
571                         : ( (selector == -27 && (MODE == 14 || MODE == 22)) ?
                              sample_61n
572                         : ( (selector == -27 && (MODE == 15 || MODE == 21)) ?
                              sample_46n
573                         : ( (selector == -27 && (MODE == 16 || MODE == 20)) ?
                              sample_37n
574                         : ( (selector == -27 && (MODE == 17 || MODE == 19)) ?
                              sample_30n
575                         : ( (selector == -27 &&  MODE == 18)            ?
                              sample_24n
576                         : ( (selector == -26 && (MODE == 14 || MODE == 22)) ?
                              sample_58n
577                         : ( (selector == -26 && (MODE == 15 || MODE == 21)) ?
                              sample_44n
578                         : ( (selector == -26 && (MODE == 16 || MODE == 20)) ?
                              sample_36n
579                         : ( (selector == -26 && (MODE == 17 || MODE == 19)) ?
                              sample_29n
580                         : ( (selector == -26 &&  MODE == 18)            ?
                              sample_23n
581                         : ( (selector == -25 && (MODE == 14 || MODE == 22)) ?
                              sample_56n
582                         : ( (selector == -25 && (MODE == 15 || MODE == 21)) ?
                              sample_42n
583                         : ( (selector == -25 && (MODE == 16 || MODE == 20)) ?
                              sample_34n
584                         : ( (selector == -25 && (MODE == 17 || MODE == 19)) ?
                              sample_27n
585                         : ( (selector == -25 &&  MODE == 18)            ?
                              sample_22n
586                         : ( (selector == -24 && (MODE == 14 || MODE == 22)) ?
                              sample_53n
587                         : ( (selector == -24 && (MODE == 15 || MODE == 21)) ?
                              sample_40n
588                         : ( (selector == -24 && (MODE == 16 || MODE == 20)) ?
```

```
                    sample_33n
589                 : ( (selector == -24 && (MODE == 17 || MODE == 19)) ?
                    sample_26n
590                 : ( (selector == -24 &&  MODE == 18)            ?
                    sample_21n
591                 : ( (selector == -23 && (MODE == 14 || MODE == 22)) ?
                    sample_51n
592                 : ( (selector == -23 && (MODE == 15 || MODE == 21)) ?
                    sample_39n
593                 : ( (selector == -23 && (MODE == 16 || MODE == 20)) ?
                    sample_31n
594                 : ( (selector == -23 && (MODE == 17 || MODE == 19)) ?
                    sample_25n
595                 : ( (selector == -23 &&  MODE == 18)            ?
                    sample_20n
596                 : ( (selector == -22 && (MODE == 14 || MODE == 22)) ?
                    sample_48n
597                 : ( (selector == -22 && (MODE == 15 || MODE == 21)) ?
                    sample_37n
598                 : ( (selector == -22 && (MODE == 16 || MODE == 20)) ?
                    sample_29n
599                 : ( (selector == -22 && (MODE == 17 || MODE == 19)) ?
                    sample_24n
600                 : ( (selector == -22 &&  MODE == 18)            ?
                    sample_19n
601                 : ( (selector == -21 && (MODE == 14 || MODE == 22)) ?
                    sample_46n
602                 : ( (selector == -21 && (MODE == 15 || MODE == 21)) ?
                    sample_35n
603                 : ( (selector == -21 && (MODE == 16 || MODE == 20)) ?
                    sample_28n
604                 : ( (selector == -21 && (MODE == 17 || MODE == 19)) ?
                    sample_22n
605                 : ( (selector == -21 &&  MODE == 18)            ?
                    sample_18n
606                 : ( (selector == -20 && (MODE == 13 || MODE == 23)) ?
                    sample_63n
607                 : ( (selector == -20 && (MODE == 14 || MODE == 22)) ?
                    sample_43n
608                 : ( (selector == -20 && (MODE == 15 || MODE == 21)) ?
                    sample_33n
609                 : ( (selector == -20 && (MODE == 16 || MODE == 20)) ?
                    sample_26n
610                 : ( (selector == -20 && (MODE == 17 || MODE == 19)) ?
                    sample_21n
611                 : ( (selector == -20 &&  MODE == 18)            ?
                    sample_17n
612                 : ( (selector == -19 && (MODE == 13 || MODE == 23)) ?
                    sample_59n
613                 : ( (selector == -19 && (MODE == 14 || MODE == 22)) ?
                    sample_41n
614                 : ( (selector == -19 && (MODE == 15 || MODE == 21)) ?
                    sample_31n
615                 : ( (selector == -19 && (MODE == 16 || MODE == 20)) ?
                    sample_25n
616                 : ( (selector == -19 && (MODE == 17 || MODE == 19)) ?
                    sample_20n
```

```
617                         : ( (selector == -19 &&  MODE == 18)            ?
                              sample_16n
618                         : ( (selector == -18 && (MODE == 13 || MODE == 23)) ?
                              sample_56n
619                         : ( (selector == -18 && (MODE == 14 || MODE == 22)) ?
                              sample_38n
620                         : ( (selector == -18 && (MODE == 15 || MODE == 21)) ?
                              sample_29n
621                         : ( (selector == -18 && (MODE == 16 || MODE == 20)) ?
                              sample_23n
622                         : ( (selector == -18 && (MODE == 17 || MODE == 19)) ?
                              sample_19n
623                         : ( (selector == -18 &&  MODE == 18)            ?
                              sample_15n
624                         : ( (selector == -17 && (MODE == 13 || MODE == 23)) ?
                              sample_52n
625                         : ( (selector == -17 && (MODE == 14 || MODE == 22)) ?
                              sample_36n
626                         : ( (selector == -17 && (MODE == 15 || MODE == 21)) ?
                              sample_27n
627                         : ( (selector == -17 && (MODE == 16 || MODE == 20)) ?
                              sample_22n
628                         : ( (selector == -17 && (MODE == 17 || MODE == 19)) ?
                              sample_17n
629                         : ( (selector == -17 &&  MODE == 18)            ?
                              sample_14n
630                         : ( (selector == -16 && (MODE == 13 || MODE == 23)) ?
                              sample_49n
631                         : ( (selector == -16 && (MODE == 14 || MODE == 22)) ?
                              sample_33n
632                         : ( (selector == -16 && (MODE == 15 || MODE == 21)) ?
                              sample_25n
633                         : ( (selector == -16 && (MODE == 16 || MODE == 20)) ?
                              sample_20n
634                         : ( (selector == -16 && (MODE == 17 || MODE == 19)) ?
                              sample_16n
635                         : ( (selector == -16 &&  MODE == 18)            ?
                              sample_13n
636                         : ( (selector == -15 && (MODE == 13 || MODE == 23)) ?
                              sample_45n
637                         : ( (selector == -15 && (MODE == 14 || MODE == 22)) ?
                              sample_31n
638                         : ( (selector == -15 && (MODE == 15 || MODE == 21)) ?
                              sample_23n
639                         : ( (selector == -15 && (MODE == 16 || MODE == 20)) ?
                              sample_19n
640                         : ( (selector == -15 && (MODE == 17 || MODE == 19)) ?
                              sample_15n
641                         : ( (selector == -15 &&  MODE == 18)            ?
                              sample_12n
642                         : ( (selector == -14 && (MODE == 13 || MODE == 23)) ?
                              sample_42n
643                         : ( (selector == -14 && (MODE == 14 || MODE == 22)) ?
                              sample_29n
644                         : ( (selector == -14 && (MODE == 15 || MODE == 21)) ?
                              sample_22n
645                         : ( (selector == -14 && (MODE == 16 || MODE == 20)) ?
```

```
                        sample_17n
646                   : ( (selector == -14 && (MODE == 17 || MODE == 19)) ?
                        sample_14n
647                   : ( (selector == -14 &&  MODE == 18)            ?
                        sample_11n
648                   : ( (selector == -13 && (MODE == 13 || MODE == 23)) ?
                        sample_38n
649                   : ( (selector == -13 && (MODE == 14 || MODE == 22)) ?
                        sample_26n
650                   : ( (selector == -13 && (MODE == 15 || MODE == 21)) ?
                        sample_20n
651                   : ( (selector == -13 && (MODE == 16 || MODE == 20)) ?
                        sample_16n
652                   : ( (selector == -13 && (MODE == 17 || MODE == 19)) ?
                        sample_13n
653                   : ( (selector == -13 &&  MODE == 18)            ?
                        sample_10n
654                   : ( (selector == -12 && (MODE == 12 || MODE == 24)) ?
                        sample_63n
655                   : ( (selector == -12 && (MODE == 13 || MODE == 23)) ?
                        sample_35n
656                   : ( (selector == -12 && (MODE == 14 || MODE == 22)) ?
                        sample_24n
657                   : ( (selector == -12 && (MODE == 15 || MODE == 21)) ?
                        sample_18n
658                   : ( (selector == -12 && (MODE == 16 || MODE == 20)) ?
                        sample_14n
659                   : ( (selector == -12 && (MODE == 17 || MODE == 19)) ?
                        sample_11n
660                   : ( (selector == -12 &&  MODE == 18)            ? sample_9n
661                   : ( (selector == -11 && (MODE == 12 || MODE == 24)) ?
                        sample_57n
662                   : ( (selector == -11 && (MODE == 13 || MODE == 23)) ?
                        sample_31n
663                   : ( (selector == -11 && (MODE == 14 || MODE == 22)) ?
                        sample_21n
664                   : ( (selector == -11 && (MODE == 15 || MODE == 21)) ?
                        sample_16n
665                   : ( (selector == -11 && (MODE == 16 || MODE == 20)) ?
                        sample_13n
666                   : ( (selector == -11 && (MODE == 17 || MODE == 19)) ?
                        sample_10n
667                   : ( (selector == -11 &&  MODE == 18)            ? sample_8n
668                   : ( (selector == -10 && (MODE == 12 || MODE == 24)) ?
                        sample_50n
669                   : ( (selector == -10 && (MODE == 13 || MODE == 23)) ?
                        sample_27n
670                   : ( (selector == -10 && (MODE == 14 || MODE == 22)) ?
                        sample_19n
671                   : ( (selector == -10 && (MODE == 15 || MODE == 21)) ?
                        sample_14n
672                   : ( (selector == -10 && (MODE == 16 || MODE == 20)) ?
                        sample_11n
673                   : ( (selector == -10 && (MODE == 17 || MODE == 19)) ?
                        sample_9n
674                   : ( (selector == -10 &&  MODE == 18)            ? sample_7n
675                   : ( (selector ==  -9 && (MODE == 12 || MODE == 24)) ?
```

```
                    sample_44n
676                 : ( (selector ==  -9 && (MODE == 13 || MODE == 23)) ?
                    sample_24n
677                 : ( (selector ==  -9 && (MODE == 14 || MODE == 22)) ?
                    sample_16n
678                 : ( (selector ==  -9 && (MODE == 15 || MODE == 21)) ?
                    sample_12n
679                 : ( (selector ==  -9 && (MODE == 16 || MODE == 20)) ?
                    sample_10n
680                 : ( (selector ==  -9 && (MODE == 17 || MODE == 19)) ?
                    sample_8n
681                 : ( (selector ==  -9 &&  MODE == 18)            ? sample_6n
682                 : ( (selector ==  -8 && (MODE == 12 || MODE == 24)) ?
                    sample_37n
683                 : ( (selector ==  -8 && (MODE == 13 || MODE == 23)) ?
                    sample_20n
684                 : ( (selector ==  -8 && (MODE == 14 || MODE == 22)) ?
                    sample_14n
685                 : ( (selector ==  -8 && (MODE == 15 || MODE == 21)) ?
                    sample_10n
686                 : ( (selector ==  -8 && (MODE == 16 || MODE == 20)) ?
                    sample_8n
687                 : ( (selector ==  -8 && (MODE == 17 || MODE == 19)) ?
                    sample_6n
688                 : ( (selector ==  -8 &&  MODE == 18)            ? sample_5n
689                 : ( (selector ==  -7 && (MODE == 12 || MODE == 24)) ?
                    sample_31n
690                 : ( (selector ==  -7 && (MODE == 13 || MODE == 23)) ?
                    sample_17n
691                 : ( (selector ==  -7 && (MODE == 14 || MODE == 22)) ?
                    sample_11n
692                 : ( (selector ==  -7 && (MODE == 15 || MODE == 21)) ?
                    sample_8n
693                 : ( (selector ==  -7 && (MODE == 16 || MODE == 20)) ?
                    sample_7n
694                 : ( (selector ==  -7 && (MODE == 17 || MODE == 19)) ?
                    sample_5n
695                 : ( (selector ==  -7 &&  MODE == 18)            ? sample_4n
696                 : ( (selector ==  -6 && (MODE == 11 || MODE == 25)) ?
                    sample_63n
697                 : ( (selector ==  -6 && (MODE == 12 || MODE == 24)) ?
                    sample_25n
698                 : ( (selector ==  -6 && (MODE == 13 || MODE == 23)) ?
                    sample_13n
699                 : ( (selector ==  -6 && (MODE == 14 || MODE == 22)) ?
                    sample_9n
700                 : ( (selector ==  -6 && (MODE == 15 || MODE == 21)) ?
                    sample_7n
701                 : ( (selector ==  -6 && (MODE == 16 || MODE == 20)) ?
                    sample_5n
702                 : ( (selector ==  -6 && (MODE == 17 || MODE == 19)) ?
                    sample_4n
703                 : ( (selector ==  -6 &&  MODE == 18)            ? sample_3n
704                 : ( (selector ==  -5 && (MODE == 11 || MODE == 25)) ?
                    sample_47n
705                 : ( (selector ==  -5 && (MODE == 12 || MODE == 24)) ?
                    sample_18n
```

```
706                     : ( (selector ==  -5 && (MODE == 13 || MODE == 23)) ?
                            sample_10n
707                     : ( (selector ==  -5 && (MODE == 14 || MODE == 22)) ?
                            sample_6n
708                     : ( (selector ==  -5 && (MODE == 15 || MODE == 21)) ?
                            sample_5n
709                     : ( (selector ==  -5 && (MODE == 16 || MODE == 20)) ?
                            sample_4n
710                     : ( (selector ==  -5 && (MODE == 17 || MODE == 19)) ?
                            sample_3n
711                     : ( (selector ==  -5 &&  MODE == 18)            ? sample_2n
712                     : ( (selector ==  -4 && (MODE == 11 || MODE == 25)) ?
                            sample_31n
713                     : ( (selector ==  -4 && (MODE == 12 || MODE == 24)) ?
                            sample_12n
714                     : ( (selector ==  -4 && (MODE == 13 || MODE == 23)) ?
                            sample_6n
715                     : ( (selector ==  -4 && (MODE == 14 || MODE == 22)) ?
                            sample_4n
716                     : ( (selector ==  -4 && (MODE == 15 || MODE == 21)) ?
                            sample_3n
717                     : ( (selector ==  -4 && (MODE == 16 || MODE == 20)) ?
                            sample_2n
718                     : ( (selector ==  -4 && (MODE == 17 || MODE == 19)) ?
                            sample_1n
719                     : ( (selector ==  -4 &&  MODE == 18)            ? sample_1n
720                     : ( (selector ==  -3 && (MODE == 11 || MODE == 25)) ?
                            sample_15n
721                     : ( (selector ==  -3 && (MODE == 12 || MODE == 24)) ?
                            sample_5n
722                     : ( (selector ==  -3 && (MODE == 13 || MODE == 23)) ?
                            sample_3n
723                     : ( (selector ==  -3 && (MODE == 14 || MODE == 22)) ?
                            sample_1n
724                     : ( (selector ==  -3 && (MODE == 15 || MODE == 21)) ?
                            sample_1n
725                     : ( (selector ==  -3 && (MODE == 16 || MODE == 20)) ?
                            sample_1n
726                     : ( (selector ==  -3 && (MODE == 17 || MODE == 19)) ?
                            sample_0n
727                     : ( (selector ==  -3 &&  MODE == 18)            ? sample_0n
728                     : (selector ==  -2 ? sample_NN
729                     : (selector ==  -1 ? sample_n0
730                     : (selector ==   0 ? sample_n1
731                     : (selector ==   1 ? sample_n2
732                     : (selector ==   2 ? sample_n3
733                     : (selector ==   3 ? sample_n4
734                     : (selector ==   4 ? sample_n5
735                     : (selector ==   5 ? sample_n6
736                     : (selector ==   6 ? sample_n7
737                     : (selector ==   7 ? sample_n8
738                     : (selector ==   8 ? sample_n9
739                     : (selector ==   9 ? sample_n10
740                     : (selector ==  10 ? sample_n11
741                     : (selector ==  11 ? sample_n12
742                     : (selector ==  12 ? sample_n13
743                     : (selector ==  13 ? sample_n14
```

```
744                     : (selector ==  14 ? sample_n15
745                     : (selector ==  15 ? sample_n16
746                     : (selector ==  16 ? sample_n17
747                     : (selector ==  17 ? sample_n18
748                     : (selector ==  18 ? sample_n19
749                     : (selector ==  19 ? sample_n20
750                     : (selector ==  20 ? sample_n21
751                     : (selector ==  21 ? sample_n22
752                     : (selector ==  22 ? sample_n23
753                     : (selector ==  23 ? sample_n24
754                     : (selector ==  24 ? sample_n25
755                     : (selector ==  25 ? sample_n26
756                     : (selector ==  26 ? sample_n27
757                     : (selector ==  27 ? sample_n28
758                     : (selector ==  28 ? sample_n29
759                     : (selector ==  29 ? sample_n30
760                     : (selector ==  30 ? sample_n31
761                     : (selector ==  31 ? sample_n32
762                     : (selector ==  32 ? sample_n33
763                     : (selector ==  33 ? sample_n34
764                     : (selector ==  34 ? sample_n35
765                     : (selector ==  35 ? sample_n36
766                     : (selector ==  36 ? sample_n37
767                     : (selector ==  37 ? sample_n38
768                     : (selector ==  38 ? sample_n39
769                     : (selector ==  39 ? sample_n40
770                     : (selector ==  40 ? sample_n41
771                     : (selector ==  41 ? sample_n42
772                     : (selector ==  42 ? sample_n43
773                     : (selector ==  43 ? sample_n44
774                     : (selector ==  44 ? sample_n45
775                     : (selector ==  45 ? sample_n46
776                     : (selector ==  46 ? sample_n47
777                     : (selector ==  47 ? sample_n48
778                     : (selector ==  48 ? sample_n49
779                     : (selector ==  49 ? sample_n50
780                     : (selector ==  50 ? sample_n51
781                     : (selector ==  51 ? sample_n52
782                     : (selector ==  52 ? sample_n53
783                     : (selector ==  53 ? sample_n54
784                     : (selector ==  54 ? sample_n55
785                     : (selector ==  55 ? sample_n56
786                     : (selector ==  56 ? sample_n57
787                     : (selector ==  57 ? sample_n58
788                     : (selector ==  58 ? sample_n59
789                     : (selector ==  59 ? sample_n60
790                     : (selector ==  60 ? sample_n61
791                     : (selector ==  61 ? sample_n62
792                     : (selector ==  62 ? sample_n63
793                     : 0 ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                        ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                        ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                        ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                        ))))))))))))))))))))
794                     : ( (selector == -32 && (MODE == 15 || MODE == 21)) ?
                          sample_n55
795                     : ( (selector == -32 && (MODE == 16 || MODE == 20)) ?
```

```
                    sample_n45
796                 : ( (selector == -32 && (MODE == 17 || MODE == 19)) ?
                    sample_n36
797                 : ( (selector == -32 &&  MODE == 18)            ?
                    sample_n29
798                 : ( (selector == -31 && (MODE == 15 || MODE == 21)) ?
                    sample_n54
799                 : ( (selector == -31 && (MODE == 16 || MODE == 20)) ?
                    sample_n43
800                 : ( (selector == -31 && (MODE == 17 || MODE == 19)) ?
                    sample_n35
801                 : ( (selector == -31 &&  MODE == 18)            ?
                    sample_n28
802                 : ( (selector == -30 && (MODE == 15 || MODE == 21)) ?
                    sample_n52
803                 : ( (selector == -30 && (MODE == 16 || MODE == 20)) ?
                    sample_n42
804                 : ( (selector == -30 && (MODE == 17 || MODE == 19)) ?
                    sample_n33
805                 : ( (selector == -30 &&  MODE == 18)            ?
                    sample_n27
806                 : ( (selector == -29 && (MODE == 15 || MODE == 21)) ?
                    sample_n50
807                 : ( (selector == -29 && (MODE == 16 || MODE == 20)) ?
                    sample_n40
808                 : ( (selector == -29 && (MODE == 17 || MODE == 19)) ?
                    sample_n32
809                 : ( (selector == -29 &&  MODE == 18)            ?
                    sample_n26
810                 : ( (selector == -28 && (MODE == 14 || MODE == 22)) ?
                    sample_n63
811                 : ( (selector == -28 && (MODE == 15 || MODE == 21)) ?
                    sample_n48
812                 : ( (selector == -28 && (MODE == 16 || MODE == 20)) ?
                    sample_n39
813                 : ( (selector == -28 && (MODE == 17 || MODE == 19)) ?
                    sample_n31
814                 : ( (selector == -28 &&  MODE == 18)            ?
                    sample_n25
815                 : ( (selector == -27 && (MODE == 14 || MODE == 22)) ?
                    sample_n61
816                 : ( (selector == -27 && (MODE == 15 || MODE == 21)) ?
                    sample_n46
817                 : ( (selector == -27 && (MODE == 16 || MODE == 20)) ?
                    sample_n37
818                 : ( (selector == -27 && (MODE == 17 || MODE == 19)) ?
                    sample_n30
819                 : ( (selector == -27 &&  MODE == 18)            ?
                    sample_n24
820                 : ( (selector == -26 && (MODE == 14 || MODE == 22)) ?
                    sample_n58
821                 : ( (selector == -26 && (MODE == 15 || MODE == 21)) ?
                    sample_n44
822                 : ( (selector == -26 && (MODE == 16 || MODE == 20)) ?
                    sample_n36
823                 : ( (selector == -26 && (MODE == 17 || MODE == 19)) ?
                    sample_n29
```

```
824                         : ( (selector == -26 &&  MODE == 18)            ?
                              sample_n23
825                         : ( (selector == -25 && (MODE == 14 || MODE == 22)) ?
                              sample_n56
826                         : ( (selector == -25 && (MODE == 15 || MODE == 21)) ?
                              sample_n42
827                         : ( (selector == -25 && (MODE == 16 || MODE == 20)) ?
                              sample_n34
828                         : ( (selector == -25 && (MODE == 17 || MODE == 19)) ?
                              sample_n27
829                         : ( (selector == -25 &&  MODE == 18)            ?
                              sample_n22
830                         : ( (selector == -24 && (MODE == 14 || MODE == 22)) ?
                              sample_n53
831                         : ( (selector == -24 && (MODE == 15 || MODE == 21)) ?
                              sample_n40
832                         : ( (selector == -24 && (MODE == 16 || MODE == 20)) ?
                              sample_n33
833                         : ( (selector == -24 && (MODE == 17 || MODE == 19)) ?
                              sample_n26
834                         : ( (selector == -24 &&  MODE == 18)            ?
                              sample_n21
835                         : ( (selector == -23 && (MODE == 14 || MODE == 22)) ?
                              sample_n51
836                         : ( (selector == -23 && (MODE == 15 || MODE == 21)) ?
                              sample_n39
837                         : ( (selector == -23 && (MODE == 16 || MODE == 20)) ?
                              sample_n31
838                         : ( (selector == -23 && (MODE == 17 || MODE == 19)) ?
                              sample_n25
839                         : ( (selector == -23 &&  MODE == 18)            ?
                              sample_n20
840                         : ( (selector == -22 && (MODE == 14 || MODE == 22)) ?
                              sample_n48
841                         : ( (selector == -22 && (MODE == 15 || MODE == 21)) ?
                              sample_n37
842                         : ( (selector == -22 && (MODE == 16 || MODE == 20)) ?
                              sample_n29
843                         : ( (selector == -22 && (MODE == 17 || MODE == 19)) ?
                              sample_n24
844                         : ( (selector == -22 &&  MODE == 18)            ?
                              sample_n19
845                         : ( (selector == -21 && (MODE == 14 || MODE == 22)) ?
                              sample_n46
846                         : ( (selector == -21 && (MODE == 15 || MODE == 21)) ?
                              sample_n35
847                         : ( (selector == -21 && (MODE == 16 || MODE == 20)) ?
                              sample_n28
848                         : ( (selector == -21 && (MODE == 17 || MODE == 19)) ?
                              sample_n22
849                         : ( (selector == -21 &&  MODE == 18)            ?
                              sample_n18
850                         : ( (selector == -20 && (MODE == 13 || MODE == 23)) ?
                              sample_n63
851                         : ( (selector == -20 && (MODE == 14 || MODE == 22)) ?
                              sample_n43
852                         : ( (selector == -20 && (MODE == 15 || MODE == 21)) ?
```

```
                    sample_n33
853                 : ( (selector == -20 && (MODE == 16 || MODE == 20)) ?
                    sample_n26
854                 : ( (selector == -20 && (MODE == 17 || MODE == 19)) ?
                    sample_n21
855                 : ( (selector == -20 &&  MODE == 18)            ?
                    sample_n17
856                 : ( (selector == -19 && (MODE == 13 || MODE == 23)) ?
                    sample_n59
857                 : ( (selector == -19 && (MODE == 14 || MODE == 22)) ?
                    sample_n41
858                 : ( (selector == -19 && (MODE == 15 || MODE == 21)) ?
                    sample_n31
859                 : ( (selector == -19 && (MODE == 16 || MODE == 20)) ?
                    sample_n25
860                 : ( (selector == -19 && (MODE == 17 || MODE == 19)) ?
                    sample_n20
861                 : ( (selector == -19 &&  MODE == 18)            ?
                    sample_n16
862                 : ( (selector == -18 && (MODE == 13 || MODE == 23)) ?
                    sample_n56
863                 : ( (selector == -18 && (MODE == 14 || MODE == 22)) ?
                    sample_n38
864                 : ( (selector == -18 && (MODE == 15 || MODE == 21)) ?
                    sample_n29
865                 : ( (selector == -18 && (MODE == 16 || MODE == 20)) ?
                    sample_n23
866                 : ( (selector == -18 && (MODE == 17 || MODE == 19)) ?
                    sample_n19
867                 : ( (selector == -18 &&  MODE == 18)            ?
                    sample_n15
868                 : ( (selector == -17 && (MODE == 13 || MODE == 23)) ?
                    sample_n52
869                 : ( (selector == -17 && (MODE == 14 || MODE == 22)) ?
                    sample_n36
870                 : ( (selector == -17 && (MODE == 15 || MODE == 21)) ?
                    sample_n27
871                 : ( (selector == -17 && (MODE == 16 || MODE == 20)) ?
                    sample_n22
872                 : ( (selector == -17 && (MODE == 17 || MODE == 19)) ?
                    sample_n17
873                 : ( (selector == -17 &&  MODE == 18)            ?
                    sample_n14
874                 : ( (selector == -16 && (MODE == 13 || MODE == 23)) ?
                    sample_n49
875                 : ( (selector == -16 && (MODE == 14 || MODE == 22)) ?
                    sample_n33
876                 : ( (selector == -16 && (MODE == 15 || MODE == 21)) ?
                    sample_n25
877                 : ( (selector == -16 && (MODE == 16 || MODE == 20)) ?
                    sample_n20
878                 : ( (selector == -16 && (MODE == 17 || MODE == 19)) ?
                    sample_n16
879                 : ( (selector == -16 &&  MODE == 18)            ?
                    sample_n13
880                 : ( (selector == -15 && (MODE == 13 || MODE == 23)) ?
                    sample_n45
```

```
881                         : ( (selector == -15 && (MODE == 14 || MODE == 22)) ?
                              sample_n31
882                         : ( (selector == -15 && (MODE == 15 || MODE == 21)) ?
                              sample_n23
883                         : ( (selector == -15 && (MODE == 16 || MODE == 20)) ?
                              sample_n19
884                         : ( (selector == -15 && (MODE == 17 || MODE == 19)) ?
                              sample_n15
885                         : ( (selector == -15 &&  MODE == 18)            ?
                              sample_n12
886                         : ( (selector == -14 && (MODE == 13 || MODE == 23)) ?
                              sample_n42
887                         : ( (selector == -14 && (MODE == 14 || MODE == 22)) ?
                              sample_n29
888                         : ( (selector == -14 && (MODE == 15 || MODE == 21)) ?
                              sample_n22
889                         : ( (selector == -14 && (MODE == 16 || MODE == 20)) ?
                              sample_n17
890                         : ( (selector == -14 && (MODE == 17 || MODE == 19)) ?
                              sample_n14
891                         : ( (selector == -14 &&  MODE == 18)            ?
                              sample_n11
892                         : ( (selector == -13 && (MODE == 13 || MODE == 23)) ?
                              sample_n38
893                         : ( (selector == -13 && (MODE == 14 || MODE == 22)) ?
                              sample_n26
894                         : ( (selector == -13 && (MODE == 15 || MODE == 21)) ?
                              sample_n20
895                         : ( (selector == -13 && (MODE == 16 || MODE == 20)) ?
                              sample_n16
896                         : ( (selector == -13 && (MODE == 17 || MODE == 19)) ?
                              sample_n13
897                         : ( (selector == -13 &&  MODE == 18)            ?
                              sample_n10
898                         : ( (selector == -12 && (MODE == 12 || MODE == 24)) ?
                              sample_n63
899                         : ( (selector == -12 && (MODE == 13 || MODE == 23)) ?
                              sample_n35
900                         : ( (selector == -12 && (MODE == 14 || MODE == 22)) ?
                              sample_n24
901                         : ( (selector == -12 && (MODE == 15 || MODE == 21)) ?
                              sample_n18
902                         : ( (selector == -12 && (MODE == 16 || MODE == 20)) ?
                              sample_n14
903                         : ( (selector == -12 && (MODE == 17 || MODE == 19)) ?
                              sample_n11
904                         : ( (selector == -12 &&  MODE == 18)            ? sample_n9
905                         : ( (selector == -11 && (MODE == 12 || MODE == 24)) ?
                              sample_n57
906                         : ( (selector == -11 && (MODE == 13 || MODE == 23)) ?
                              sample_n31
907                         : ( (selector == -11 && (MODE == 14 || MODE == 22)) ?
                              sample_n21
908                         : ( (selector == -11 && (MODE == 15 || MODE == 21)) ?
                              sample_n16
909                         : ( (selector == -11 && (MODE == 16 || MODE == 20)) ?
                              sample_n13
```

```
910                         : ( (selector == -11 && (MODE == 17 || MODE == 19)) ?
                              sample_n10
911                         : ( (selector == -11 &&  MODE == 18)            ? sample_n8
912                         : ( (selector == -10 && (MODE == 12 || MODE == 24)) ?
                              sample_n50
913                         : ( (selector == -10 && (MODE == 13 || MODE == 23)) ?
                              sample_n27
914                         : ( (selector == -10 && (MODE == 14 || MODE == 22)) ?
                              sample_n19
915                         : ( (selector == -10 && (MODE == 15 || MODE == 21)) ?
                              sample_n14
916                         : ( (selector == -10 && (MODE == 16 || MODE == 20)) ?
                              sample_n11
917                         : ( (selector == -10 && (MODE == 17 || MODE == 19)) ?
                              sample_n9
918                         : ( (selector == -10 &&  MODE == 18)            ? sample_n7
919                         : ( (selector ==  -9 && (MODE == 12 || MODE == 24)) ?
                              sample_n44
920                         : ( (selector ==  -9 && (MODE == 13 || MODE == 23)) ?
                              sample_n24
921                         : ( (selector ==  -9 && (MODE == 14 || MODE == 22)) ?
                              sample_n16
922                         : ( (selector ==  -9 && (MODE == 15 || MODE == 21)) ?
                              sample_n12
923                         : ( (selector ==  -9 && (MODE == 16 || MODE == 20)) ?
                              sample_n10
924                         : ( (selector ==  -9 && (MODE == 17 || MODE == 19)) ?
                              sample_n8
925                         : ( (selector ==  -9 &&  MODE == 18)            ? sample_n6
926                         : ( (selector ==  -8 && (MODE == 12 || MODE == 24)) ?
                              sample_n37
927                         : ( (selector ==  -8 && (MODE == 13 || MODE == 23)) ?
                              sample_n20
928                         : ( (selector ==  -8 && (MODE == 14 || MODE == 22)) ?
                              sample_n14
929                         : ( (selector ==  -8 && (MODE == 15 || MODE == 21)) ?
                              sample_n10
930                         : ( (selector ==  -8 && (MODE == 16 || MODE == 20)) ?
                              sample_n8
931                         : ( (selector ==  -8 && (MODE == 17 || MODE == 19)) ?
                              sample_n6
932                         : ( (selector ==  -8 &&  MODE == 18)            ? sample_n5
933                         : ( (selector ==  -7 && (MODE == 12 || MODE == 24)) ?
                              sample_n31
934                         : ( (selector ==  -7 && (MODE == 13 || MODE == 23)) ?
                              sample_n17
935                         : ( (selector ==  -7 && (MODE == 14 || MODE == 22)) ?
                              sample_n11
936                         : ( (selector ==  -7 && (MODE == 15 || MODE == 21)) ?
                              sample_n8
937                         : ( (selector ==  -7 && (MODE == 16 || MODE == 20)) ?
                              sample_n7
938                         : ( (selector ==  -7 && (MODE == 17 || MODE == 19)) ?
                              sample_n5
939                         : ( (selector ==  -7 &&  MODE == 18)            ? sample_n4
940                         : ( (selector ==  -6 && (MODE == 11 || MODE == 25)) ?
                              sample_n63
```

```
941                         : ( (selector ==  -6 && (MODE == 12 || MODE == 24)) ?
                              sample_n25
942                         : ( (selector ==  -6 && (MODE == 13 || MODE == 23)) ?
                              sample_n13
943                         : ( (selector ==  -6 && (MODE == 14 || MODE == 22)) ?
                              sample_n9
944                         : ( (selector ==  -6 && (MODE == 15 || MODE == 21)) ?
                              sample_n7
945                         : ( (selector ==  -6 && (MODE == 16 || MODE == 20)) ?
                              sample_n5
946                         : ( (selector ==  -6 && (MODE == 17 || MODE == 19)) ?
                              sample_n4
947                         : ( (selector ==  -6 &&  MODE == 18)            ? sample_n3
948                         : ( (selector ==  -5 && (MODE == 11 || MODE == 25)) ?
                              sample_n47
949                         : ( (selector ==  -5 && (MODE == 12 || MODE == 24)) ?
                              sample_n18
950                         : ( (selector ==  -5 && (MODE == 13 || MODE == 23)) ?
                              sample_n10
951                         : ( (selector ==  -5 && (MODE == 14 || MODE == 22)) ?
                              sample_n6
952                         : ( (selector ==  -5 && (MODE == 15 || MODE == 21)) ?
                              sample_n5
953                         : ( (selector ==  -5 && (MODE == 16 || MODE == 20)) ?
                              sample_n4
954                         : ( (selector ==  -5 && (MODE == 17 || MODE == 19)) ?
                              sample_n3
955                         : ( (selector ==  -5 &&  MODE == 18)            ? sample_n2
956                         : ( (selector ==  -4 && (MODE == 11 || MODE == 25)) ?
                              sample_n31
957                         : ( (selector ==  -4 && (MODE == 12 || MODE == 24)) ?
                              sample_n12
958                         : ( (selector ==  -4 && (MODE == 13 || MODE == 23)) ?
                              sample_n6
959                         : ( (selector ==  -4 && (MODE == 14 || MODE == 22)) ?
                              sample_n4
960                         : ( (selector ==  -4 && (MODE == 15 || MODE == 21)) ?
                              sample_n3
961                         : ( (selector ==  -4 && (MODE == 16 || MODE == 20)) ?
                              sample_n2
962                         : ( (selector ==  -4 && (MODE == 17 || MODE == 19)) ?
                              sample_n1
963                         : ( (selector ==  -4 &&  MODE == 18)            ? sample_n1
964                         : ( (selector ==  -3 && (MODE == 11 || MODE == 25)) ?
                              sample_n15
965                         : ( (selector ==  -3 && (MODE == 12 || MODE == 24)) ?
                              sample_n5
966                         : ( (selector ==  -3 && (MODE == 13 || MODE == 23)) ?
                              sample_n3
967                         : ( (selector ==  -3 && (MODE == 14 || MODE == 22)) ?
                              sample_n1
968                         : ( (selector ==  -3 && (MODE == 15 || MODE == 21)) ?
                              sample_n1
969                         : ( (selector ==  -3 && (MODE == 16 || MODE == 20)) ?
                              sample_n1
970                         : ( (selector ==  -3 && (MODE == 17 || MODE == 19)) ?
                              sample_n0
```

```
971                         : ( (selector ==  -3 &&  MODE == 18)            ? sample_n0
972                         : (selector ==  -2 ? sample_NN
973                         : (selector ==  -1 ? sample_0n
974                         : (selector ==   0 ? sample_1n
975                         : (selector ==   1 ? sample_2n
976                         : (selector ==   2 ? sample_3n
977                         : (selector ==   3 ? sample_4n
978                         : (selector ==   4 ? sample_5n
979                         : (selector ==   5 ? sample_6n
980                         : (selector ==   6 ? sample_7n
981                         : (selector ==   7 ? sample_8n
982                         : (selector ==   8 ? sample_9n
983                         : (selector ==   9 ? sample_10n
984                         : (selector ==  10 ? sample_11n
985                         : (selector ==  11 ? sample_12n
986                         : (selector ==  12 ? sample_13n
987                         : (selector ==  13 ? sample_14n
988                         : (selector ==  14 ? sample_15n
989                         : (selector ==  15 ? sample_16n
990                         : (selector ==  16 ? sample_17n
991                         : (selector ==  17 ? sample_18n
992                         : (selector ==  18 ? sample_19n
993                         : (selector ==  19 ? sample_20n
994                         : (selector ==  20 ? sample_21n
995                         : (selector ==  21 ? sample_22n
996                         : (selector ==  22 ? sample_23n
997                         : (selector ==  23 ? sample_24n
998                         : (selector ==  24 ? sample_25n
999                         : (selector ==  25 ? sample_26n
1000                        : (selector ==  26 ? sample_27n
1001                        : (selector ==  27 ? sample_28n
1002                        : (selector ==  28 ? sample_29n
1003                        : (selector ==  29 ? sample_30n
1004                        : (selector ==  30 ? sample_31n
1005                        : (selector ==  31 ? sample_32n
1006                        : (selector ==  32 ? sample_33n
1007                        : (selector ==  33 ? sample_34n
1008                        : (selector ==  34 ? sample_35n
1009                        : (selector ==  35 ? sample_36n
1010                        : (selector ==  36 ? sample_37n
1011                        : (selector ==  37 ? sample_38n
1012                        : (selector ==  38 ? sample_39n
1013                        : (selector ==  39 ? sample_40n
1014                        : (selector ==  40 ? sample_41n
1015                        : (selector ==  41 ? sample_42n
1016                        : (selector ==  42 ? sample_43n
1017                        : (selector ==  43 ? sample_44n
1018                        : (selector ==  44 ? sample_45n
1019                        : (selector ==  45 ? sample_46n
1020                        : (selector ==  46 ? sample_47n
1021                        : (selector ==  47 ? sample_48n
1022                        : (selector ==  48 ? sample_49n
1023                        : (selector ==  49 ? sample_50n
1024                        : (selector ==  50 ? sample_51n
1025                        : (selector ==  51 ? sample_52n
1026                        : (selector ==  52 ? sample_53n
1027                        : (selector ==  53 ? sample_54n
```

```
                      : (selector ==  54 ? sample_55n
                      : (selector ==  55 ? sample_56n
                      : (selector ==  56 ? sample_57n
                      : (selector ==  57 ? sample_58n
                      : (selector ==  58 ? sample_59n
                      : (selector ==  59 ? sample_60n
                      : (selector ==  60 ? sample_61n
                      : (selector ==  61 ? sample_62n
                      : (selector ==  62 ? sample_63n
                      : 0 ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                          ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                          ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                          ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                          ))))))))))))))))))))));

    assign ref3 = MODE < 18 ? ( (selector == -32 && (MODE == 15 || MODE == 21)) ?
         sample_54n
                      : ( (selector == -32 && (MODE == 16 || MODE == 20)) ?
                          sample_43n
                      : ( (selector == -32 && (MODE == 17 || MODE == 19)) ?
                          sample_35n
                      : ( (selector == -32 &&  MODE == 18)            ?
                          sample_28n
                      : ( (selector == -31 && (MODE == 15 || MODE == 21)) ?
                          sample_52n
                      : ( (selector == -31 && (MODE == 16 || MODE == 20)) ?
                          sample_42n
                      : ( (selector == -31 && (MODE == 17 || MODE == 19)) ?
                          sample_33n
                      : ( (selector == -31 &&  MODE == 18)            ?
                          sample_27n
                      : ( (selector == -30 && (MODE == 15 || MODE == 21)) ?
                          sample_50n
                      : ( (selector == -30 && (MODE == 16 || MODE == 20)) ?
                          sample_40n
                      : ( (selector == -30 && (MODE == 17 || MODE == 19)) ?
                          sample_32n
                      : ( (selector == -30 &&  MODE == 18)            ?
                          sample_26n
                      : ( (selector == -29 && (MODE == 14 || MODE == 22)) ?
                          sample_63n
                      : ( (selector == -29 && (MODE == 15 || MODE == 21)) ?
                          sample_48n
                      : ( (selector == -29 && (MODE == 16 || MODE == 20)) ?
                          sample_39n
                      : ( (selector == -29 && (MODE == 17 || MODE == 19)) ?
                          sample_31n
                      : ( (selector == -29 &&  MODE == 18)            ?
                          sample_25n
                      : ( (selector == -28 && (MODE == 14 || MODE == 22)) ?
                          sample_61n
                      : ( (selector == -28 && (MODE == 15 || MODE == 21)) ?
                          sample_46n
                      : ( (selector == -28 && (MODE == 16 || MODE == 20)) ?
                          sample_37n
                      : ( (selector == -28 && (MODE == 17 || MODE == 19)) ?
                          sample_30n
```

```
1060                    : ( (selector == -28 &&  MODE == 18)            ?
                          sample_24n
1061                    : ( (selector == -27 && (MODE == 14 || MODE == 22)) ?
                          sample_58n
1062                    : ( (selector == -27 && (MODE == 15 || MODE == 21)) ?
                          sample_44n
1063                    : ( (selector == -27 && (MODE == 16 || MODE == 20)) ?
                          sample_36n
1064                    : ( (selector == -27 && (MODE == 17 || MODE == 19)) ?
                          sample_29n
1065                    : ( (selector == -27 &&  MODE == 18)            ?
                          sample_23n
1066                    : ( (selector == -26 && (MODE == 14 || MODE == 22)) ?
                          sample_56n
1067                    : ( (selector == -26 && (MODE == 15 || MODE == 21)) ?
                          sample_42n
1068                    : ( (selector == -26 && (MODE == 16 || MODE == 20)) ?
                          sample_34n
1069                    : ( (selector == -26 && (MODE == 17 || MODE == 19)) ?
                          sample_27n
1070                    : ( (selector == -26 &&  MODE == 18)            ?
                          sample_22n
1071                    : ( (selector == -25 && (MODE == 14 || MODE == 22)) ?
                          sample_53n
1072                    : ( (selector == -25 && (MODE == 15 || MODE == 21)) ?
                          sample_40n
1073                    : ( (selector == -25 && (MODE == 16 || MODE == 20)) ?
                          sample_33n
1074                    : ( (selector == -25 && (MODE == 17 || MODE == 19)) ?
                          sample_26n
1075                    : ( (selector == -25 &&  MODE == 18)            ?
                          sample_21n
1076                    : ( (selector == -24 && (MODE == 14 || MODE == 22)) ?
                          sample_51n
1077                    : ( (selector == -24 && (MODE == 15 || MODE == 21)) ?
                          sample_39n
1078                    : ( (selector == -24 && (MODE == 16 || MODE == 20)) ?
                          sample_31n
1079                    : ( (selector == -24 && (MODE == 17 || MODE == 19)) ?
                          sample_25n
1080                    : ( (selector == -24 &&  MODE == 18)            ?
                          sample_20n
1081                    : ( (selector == -23 && (MODE == 14 || MODE == 22)) ?
                          sample_48n
1082                    : ( (selector == -23 && (MODE == 15 || MODE == 21)) ?
                          sample_37n
1083                    : ( (selector == -23 && (MODE == 16 || MODE == 20)) ?
                          sample_29n
1084                    : ( (selector == -23 && (MODE == 17 || MODE == 19)) ?
                          sample_24n
1085                    : ( (selector == -23 &&  MODE == 18)            ?
                          sample_19n
1086                    : ( (selector == -22 && (MODE == 14 || MODE == 22)) ?
                          sample_46n
1087                    : ( (selector == -22 && (MODE == 15 || MODE == 21)) ?
                          sample_35n
1088                    : ( (selector == -22 && (MODE == 16 || MODE == 20)) ?
```

```
                    sample_28n
1089                : ( (selector == -22 && (MODE == 17 || MODE == 19)) ?
                    sample_22n
1090                : ( (selector == -22 &&  MODE == 18)            ?
                    sample_18n
1091                : ( (selector == -21 && (MODE == 13 || MODE == 23)) ?
                    sample_63n
1092                : ( (selector == -21 && (MODE == 14 || MODE == 22)) ?
                    sample_43n
1093                : ( (selector == -21 && (MODE == 15 || MODE == 21)) ?
                    sample_33n
1094                : ( (selector == -21 && (MODE == 16 || MODE == 20)) ?
                    sample_26n
1095                : ( (selector == -21 && (MODE == 17 || MODE == 19)) ?
                    sample_21n
1096                : ( (selector == -21 &&  MODE == 18)            ?
                    sample_17n
1097                : ( (selector == -20 && (MODE == 13 || MODE == 23)) ?
                    sample_59n
1098                : ( (selector == -20 && (MODE == 14 || MODE == 22)) ?
                    sample_41n
1099                : ( (selector == -20 && (MODE == 15 || MODE == 21)) ?
                    sample_31n
1100                : ( (selector == -20 && (MODE == 16 || MODE == 20)) ?
                    sample_25n
1101                : ( (selector == -20 && (MODE == 17 || MODE == 19)) ?
                    sample_20n
1102                : ( (selector == -20 &&  MODE == 18)            ?
                    sample_16n
1103                : ( (selector == -19 && (MODE == 13 || MODE == 23)) ?
                    sample_56n
1104                : ( (selector == -19 && (MODE == 14 || MODE == 22)) ?
                    sample_38n
1105                : ( (selector == -19 && (MODE == 15 || MODE == 21)) ?
                    sample_29n
1106                : ( (selector == -19 && (MODE == 16 || MODE == 20)) ?
                    sample_23n
1107                : ( (selector == -19 && (MODE == 17 || MODE == 19)) ?
                    sample_19n
1108                : ( (selector == -19 &&  MODE == 18)            ?
                    sample_15n
1109                : ( (selector == -18 && (MODE == 13 || MODE == 23)) ?
                    sample_52n
1110                : ( (selector == -18 && (MODE == 14 || MODE == 22)) ?
                    sample_36n
1111                : ( (selector == -18 && (MODE == 15 || MODE == 21)) ?
                    sample_27n
1112                : ( (selector == -18 && (MODE == 16 || MODE == 20)) ?
                    sample_22n
1113                : ( (selector == -18 && (MODE == 17 || MODE == 19)) ?
                    sample_17n
1114                : ( (selector == -18 &&  MODE == 18)            ?
                    sample_14n
1115                : ( (selector == -17 && (MODE == 13 || MODE == 23)) ?
                    sample_49n
1116                : ( (selector == -17 && (MODE == 14 || MODE == 22)) ?
                    sample_33n
```

```
1117                        : ( (selector == -17 && (MODE == 15 || MODE == 21)) ?
                              sample_25n
1118                        : ( (selector == -17 && (MODE == 16 || MODE == 20)) ?
                              sample_20n
1119                        : ( (selector == -17 && (MODE == 17 || MODE == 19)) ?
                              sample_16n
1120                        : ( (selector == -17 &&  MODE == 18)            ?
                              sample_13n
1121                        : ( (selector == -16 && (MODE == 13 || MODE == 23)) ?
                              sample_45n
1122                        : ( (selector == -16 && (MODE == 14 || MODE == 22)) ?
                              sample_31n
1123                        : ( (selector == -16 && (MODE == 15 || MODE == 21)) ?
                              sample_23n
1124                        : ( (selector == -16 && (MODE == 16 || MODE == 20)) ?
                              sample_19n
1125                        : ( (selector == -16 && (MODE == 17 || MODE == 19)) ?
                              sample_15n
1126                        : ( (selector == -16 &&  MODE == 18)            ?
                              sample_12n
1127                        : ( (selector == -15 && (MODE == 13 || MODE == 23)) ?
                              sample_42n
1128                        : ( (selector == -15 && (MODE == 14 || MODE == 22)) ?
                              sample_29n
1129                        : ( (selector == -15 && (MODE == 15 || MODE == 21)) ?
                              sample_22n
1130                        : ( (selector == -15 && (MODE == 16 || MODE == 20)) ?
                              sample_17n
1131                        : ( (selector == -15 && (MODE == 17 || MODE == 19)) ?
                              sample_14n
1132                        : ( (selector == -15 &&  MODE == 18)            ?
                              sample_11n
1133                        : ( (selector == -14 && (MODE == 13 || MODE == 23)) ?
                              sample_38n
1134                        : ( (selector == -14 && (MODE == 14 || MODE == 22)) ?
                              sample_26n
1135                        : ( (selector == -14 && (MODE == 15 || MODE == 21)) ?
                              sample_20n
1136                        : ( (selector == -14 && (MODE == 16 || MODE == 20)) ?
                              sample_16n
1137                        : ( (selector == -14 && (MODE == 17 || MODE == 19)) ?
                              sample_13n
1138                        : ( (selector == -14 &&  MODE == 18)            ?
                              sample_10n
1139                        : ( (selector == -13 && (MODE == 12 || MODE == 24)) ?
                              sample_63n
1140                        : ( (selector == -13 && (MODE == 13 || MODE == 23)) ?
                              sample_35n
1141                        : ( (selector == -13 && (MODE == 14 || MODE == 22)) ?
                              sample_24n
1142                        : ( (selector == -13 && (MODE == 15 || MODE == 21)) ?
                              sample_18n
1143                        : ( (selector == -13 && (MODE == 16 || MODE == 20)) ?
                              sample_14n
1144                        : ( (selector == -13 && (MODE == 17 || MODE == 19)) ?
                              sample_11n
1145                        : ( (selector == -13 &&  MODE == 18)            ? sample_9n
```

```
1146                        : ( (selector == -12 && (MODE == 12 || MODE == 24)) ?
                              sample_57n
1147                        : ( (selector == -12 && (MODE == 13 || MODE == 23)) ?
                              sample_31n
1148                        : ( (selector == -12 && (MODE == 14 || MODE == 22)) ?
                              sample_21n
1149                        : ( (selector == -12 && (MODE == 15 || MODE == 21)) ?
                              sample_16n
1150                        : ( (selector == -12 && (MODE == 16 || MODE == 20)) ?
                              sample_13n
1151                        : ( (selector == -12 && (MODE == 17 || MODE == 19)) ?
                              sample_10n
1152                        : ( (selector == -12 &&  MODE == 18)            ? sample_8n
1153                        : ( (selector == -11 && (MODE == 12 || MODE == 24)) ?
                              sample_50n
1154                        : ( (selector == -11 && (MODE == 13 || MODE == 23)) ?
                              sample_27n
1155                        : ( (selector == -11 && (MODE == 14 || MODE == 22)) ?
                              sample_19n
1156                        : ( (selector == -11 && (MODE == 15 || MODE == 21)) ?
                              sample_14n
1157                        : ( (selector == -11 && (MODE == 16 || MODE == 20)) ?
                              sample_11n
1158                        : ( (selector == -11 && (MODE == 17 || MODE == 19)) ?
                              sample_9n
1159                        : ( (selector == -11 &&  MODE == 18)            ? sample_7n
1160                        : ( (selector == -10 && (MODE == 12 || MODE == 24)) ?
                              sample_44n
1161                        : ( (selector == -10 && (MODE == 13 || MODE == 23)) ?
                              sample_24n
1162                        : ( (selector == -10 && (MODE == 14 || MODE == 22)) ?
                              sample_16n
1163                        : ( (selector == -10 && (MODE == 15 || MODE == 21)) ?
                              sample_12n
1164                        : ( (selector == -10 && (MODE == 16 || MODE == 20)) ?
                              sample_10n
1165                        : ( (selector == -10 && (MODE == 17 || MODE == 19)) ?
                              sample_8n
1166                        : ( (selector == -10 &&  MODE == 18)            ? sample_6n
1167                        : ( (selector ==  -9 && (MODE == 12 || MODE == 24)) ?
                              sample_37n
1168                        : ( (selector ==  -9 && (MODE == 13 || MODE == 23)) ?
                              sample_20n
1169                        : ( (selector ==  -9 && (MODE == 14 || MODE == 22)) ?
                              sample_14n
1170                        : ( (selector ==  -9 && (MODE == 15 || MODE == 21)) ?
                              sample_10n
1171                        : ( (selector ==  -9 && (MODE == 16 || MODE == 20)) ?
                              sample_8n
1172                        : ( (selector ==  -9 && (MODE == 17 || MODE == 19)) ?
                              sample_6n
1173                        : ( (selector ==  -9 &&  MODE == 18)            ? sample_5n
1174                        : ( (selector ==  -8 && (MODE == 12 || MODE == 24)) ?
                              sample_31n
1175                        : ( (selector ==  -8 && (MODE == 13 || MODE == 23)) ?
                              sample_17n
1176                        : ( (selector ==  -8 && (MODE == 14 || MODE == 22)) ?
```

```
                    sample_11n
1177                : ( (selector ==  -8 && (MODE == 15 || MODE == 21)) ?
                    sample_8n
1178                : ( (selector ==  -8 && (MODE == 16 || MODE == 20)) ?
                    sample_7n
1179                : ( (selector ==  -8 && (MODE == 17 || MODE == 19)) ?
                    sample_5n
1180                : ( (selector ==  -8 &&  MODE == 18)            ? sample_4n
1181                : ( (selector ==  -7 && (MODE == 11 || MODE == 25)) ?
                    sample_63n
1182                : ( (selector ==  -7 && (MODE == 12 || MODE == 24)) ?
                    sample_25n
1183                : ( (selector ==  -7 && (MODE == 13 || MODE == 23)) ?
                    sample_13n
1184                : ( (selector ==  -7 && (MODE == 14 || MODE == 22)) ?
                    sample_9n
1185                : ( (selector ==  -7 && (MODE == 15 || MODE == 21)) ?
                    sample_7n
1186                : ( (selector ==  -7 && (MODE == 16 || MODE == 20)) ?
                    sample_5n
1187                : ( (selector ==  -7 && (MODE == 17 || MODE == 19)) ?
                    sample_4n
1188                : ( (selector ==  -7 &&  MODE == 18)            ? sample_3n
1189                : ( (selector ==  -6 && (MODE == 11 || MODE == 25)) ?
                    sample_47n
1190                : ( (selector ==  -6 && (MODE == 12 || MODE == 24)) ?
                    sample_18n
1191                : ( (selector ==  -6 && (MODE == 13 || MODE == 23)) ?
                    sample_10n
1192                : ( (selector ==  -6 && (MODE == 14 || MODE == 22)) ?
                    sample_6n
1193                : ( (selector ==  -6 && (MODE == 15 || MODE == 21)) ?
                    sample_5n
1194                : ( (selector ==  -6 && (MODE == 16 || MODE == 20)) ?
                    sample_4n
1195                : ( (selector ==  -6 && (MODE == 17 || MODE == 19)) ?
                    sample_3n
1196                : ( (selector ==  -6 &&  MODE == 18)            ? sample_2n
1197                : ( (selector ==  -5 && (MODE == 11 || MODE == 25)) ?
                    sample_31n
1198                : ( (selector ==  -5 && (MODE == 12 || MODE == 24)) ?
                    sample_12n
1199                : ( (selector ==  -5 && (MODE == 13 || MODE == 23)) ?
                    sample_6n
1200                : ( (selector ==  -5 && (MODE == 14 || MODE == 22)) ?
                    sample_4n
1201                : ( (selector ==  -5 && (MODE == 15 || MODE == 21)) ?
                    sample_3n
1202                : ( (selector ==  -5 && (MODE == 16 || MODE == 20)) ?
                    sample_2n
1203                : ( (selector ==  -5 && (MODE == 17 || MODE == 19)) ?
                    sample_1n
1204                : ( (selector ==  -5 &&  MODE == 18)            ? sample_1n
1205                : ( (selector ==  -4 && (MODE == 11 || MODE == 25)) ?
                    sample_15n
1206                : ( (selector ==  -4 && (MODE == 12 || MODE == 24)) ?
                    sample_5n
```

```
1207                        : ( (selector ==  -4 && (MODE == 13 || MODE == 23)) ?
                              sample_3n
1208                        : ( (selector ==  -4 && (MODE == 14 || MODE == 22)) ?
                              sample_1n
1209                        : ( (selector ==  -4 && (MODE == 15 || MODE == 21)) ?
                              sample_1n
1210                        : ( (selector ==  -4 && (MODE == 16 || MODE == 20)) ?
                              sample_1n
1211                        : ( (selector ==  -4 && (MODE == 17 || MODE == 19)) ?
                              sample_0n
1212                        : ( (selector ==  -4 &&  MODE == 18)            ? sample_0n
1213                        : (selector ==  -3 ? sample_NN
1214                        : (selector ==  -2 ? sample_n0
1215                        : (selector ==  -1 ? sample_n1
1216                        : (selector ==   0 ? sample_n2
1217                        : (selector ==   1 ? sample_n3
1218                        : (selector ==   2 ? sample_n4
1219                        : (selector ==   3 ? sample_n5
1220                        : (selector ==   4 ? sample_n6
1221                        : (selector ==   5 ? sample_n7
1222                        : (selector ==   6 ? sample_n8
1223                        : (selector ==   7 ? sample_n9
1224                        : (selector ==   8 ? sample_n10
1225                        : (selector ==   9 ? sample_n11
1226                        : (selector ==  10 ? sample_n12
1227                        : (selector ==  11 ? sample_n13
1228                        : (selector ==  12 ? sample_n14
1229                        : (selector ==  13 ? sample_n15
1230                        : (selector ==  14 ? sample_n16
1231                        : (selector ==  15 ? sample_n17
1232                        : (selector ==  16 ? sample_n18
1233                        : (selector ==  17 ? sample_n19
1234                        : (selector ==  18 ? sample_n20
1235                        : (selector ==  19 ? sample_n21
1236                        : (selector ==  20 ? sample_n22
1237                        : (selector ==  21 ? sample_n23
1238                        : (selector ==  22 ? sample_n24
1239                        : (selector ==  23 ? sample_n25
1240                        : (selector ==  24 ? sample_n26
1241                        : (selector ==  25 ? sample_n27
1242                        : (selector ==  26 ? sample_n28
1243                        : (selector ==  27 ? sample_n29
1244                        : (selector ==  28 ? sample_n30
1245                        : (selector ==  29 ? sample_n31
1246                        : (selector ==  30 ? sample_n32
1247                        : (selector ==  31 ? sample_n33
1248                        : (selector ==  32 ? sample_n34
1249                        : (selector ==  33 ? sample_n35
1250                        : (selector ==  34 ? sample_n36
1251                        : (selector ==  35 ? sample_n37
1252                        : (selector ==  36 ? sample_n38
1253                        : (selector ==  37 ? sample_n39
1254                        : (selector ==  38 ? sample_n40
1255                        : (selector ==  39 ? sample_n41
1256                        : (selector ==  40 ? sample_n42
1257                        : (selector ==  41 ? sample_n43
1258                        : (selector ==  42 ? sample_n44
```

```
: (selector ==  43 ? sample_n45
: (selector ==  44 ? sample_n46
: (selector ==  45 ? sample_n47
: (selector ==  46 ? sample_n48
: (selector ==  47 ? sample_n49
: (selector ==  48 ? sample_n50
: (selector ==  49 ? sample_n51
: (selector ==  50 ? sample_n52
: (selector ==  51 ? sample_n53
: (selector ==  52 ? sample_n54
: (selector ==  53 ? sample_n55
: (selector ==  54 ? sample_n56
: (selector ==  55 ? sample_n57
: (selector ==  56 ? sample_n58
: (selector ==  57 ? sample_n59
: (selector ==  58 ? sample_n60
: (selector ==  59 ? sample_n61
: (selector ==  60 ? sample_n62
: (selector ==  61 ? sample_n63
: 0 )))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
: ( (selector == -32 && (MODE == 15 || MODE == 21)) ? sample_n54
: ( (selector == -32 && (MODE == 16 || MODE == 20)) ? sample_n43
: ( (selector == -32 && (MODE == 17 || MODE == 19)) ? sample_n35
: ( (selector == -32 &&  MODE == 18)             ? sample_n28
: ( (selector == -31 && (MODE == 15 || MODE == 21)) ? sample_n52
: ( (selector == -31 && (MODE == 16 || MODE == 20)) ? sample_n42
: ( (selector == -31 && (MODE == 17 || MODE == 19)) ? sample_n33
: ( (selector == -31 &&  MODE == 18)             ? sample_n27
: ( (selector == -30 && (MODE == 15 || MODE == 21)) ? sample_n50
: ( (selector == -30 && (MODE == 16 || MODE == 20)) ? sample_n40
: ( (selector == -30 && (MODE == 17 || MODE == 19)) ? sample_n32
: ( (selector == -30 &&  MODE == 18)             ? sample_n26
: ( (selector == -29 && (MODE == 14 || MODE == 22)) ? sample_n63
: ( (selector == -29 && (MODE == 15 || MODE == 21)) ? sample_n48
: ( (selector == -29 && (MODE == 16 || MODE == 20)) ? sample_n39
: ( (selector == -29 && (MODE == 17 || MODE == 19)) ? sample_n31
: ( (selector == -29 &&  MODE == 18)             ?
```

```
                    sample_n25
1296              : ( (selector == -28 && (MODE == 14 || MODE == 22)) ?
                    sample_n61
1297              : ( (selector == -28 && (MODE == 15 || MODE == 21)) ?
                    sample_n46
1298              : ( (selector == -28 && (MODE == 16 || MODE == 20)) ?
                    sample_n37
1299              : ( (selector == -28 && (MODE == 17 || MODE == 19)) ?
                    sample_n30
1300              : ( (selector == -28 &&  MODE == 18)            ?
                    sample_n24
1301              : ( (selector == -27 && (MODE == 14 || MODE == 22)) ?
                    sample_n58
1302              : ( (selector == -27 && (MODE == 15 || MODE == 21)) ?
                    sample_n44
1303              : ( (selector == -27 && (MODE == 16 || MODE == 20)) ?
                    sample_n36
1304              : ( (selector == -27 && (MODE == 17 || MODE == 19)) ?
                    sample_n29
1305              : ( (selector == -27 &&  MODE == 18)            ?
                    sample_n23
1306              : ( (selector == -26 && (MODE == 14 || MODE == 22)) ?
                    sample_n56
1307              : ( (selector == -26 && (MODE == 15 || MODE == 21)) ?
                    sample_n42
1308              : ( (selector == -26 && (MODE == 16 || MODE == 20)) ?
                    sample_n34
1309              : ( (selector == -26 && (MODE == 17 || MODE == 19)) ?
                    sample_n27
1310              : ( (selector == -26 &&  MODE == 18)            ?
                    sample_n22
1311              : ( (selector == -25 && (MODE == 14 || MODE == 22)) ?
                    sample_n53
1312              : ( (selector == -25 && (MODE == 15 || MODE == 21)) ?
                    sample_n40
1313              : ( (selector == -25 && (MODE == 16 || MODE == 20)) ?
                    sample_n33
1314              : ( (selector == -25 && (MODE == 17 || MODE == 19)) ?
                    sample_n26
1315              : ( (selector == -25 &&  MODE == 18)            ?
                    sample_n21
1316              : ( (selector == -24 && (MODE == 14 || MODE == 22)) ?
                    sample_n51
1317              : ( (selector == -24 && (MODE == 15 || MODE == 21)) ?
                    sample_n39
1318              : ( (selector == -24 && (MODE == 16 || MODE == 20)) ?
                    sample_n31
1319              : ( (selector == -24 && (MODE == 17 || MODE == 19)) ?
                    sample_n25
1320              : ( (selector == -24 &&  MODE == 18)            ?
                    sample_n20
1321              : ( (selector == -23 && (MODE == 14 || MODE == 22)) ?
                    sample_n48
1322              : ( (selector == -23 && (MODE == 15 || MODE == 21)) ?
                    sample_n37
1323              : ( (selector == -23 && (MODE == 16 || MODE == 20)) ?
                    sample_n29
```

```
1324                        : ( (selector == -23 && (MODE == 17 || MODE == 19)) ?
                              sample_n24
1325                        : ( (selector == -23 &&  MODE == 18)            ?
                              sample_n19
1326                        : ( (selector == -22 && (MODE == 14 || MODE == 22)) ?
                              sample_n46
1327                        : ( (selector == -22 && (MODE == 15 || MODE == 21)) ?
                              sample_n35
1328                        : ( (selector == -22 && (MODE == 16 || MODE == 20)) ?
                              sample_n28
1329                        : ( (selector == -22 && (MODE == 17 || MODE == 19)) ?
                              sample_n22
1330                        : ( (selector == -22 &&  MODE == 18)            ?
                              sample_n18
1331                        : ( (selector == -21 && (MODE == 13 || MODE == 23)) ?
                              sample_n63
1332                        : ( (selector == -21 && (MODE == 14 || MODE == 22)) ?
                              sample_n43
1333                        : ( (selector == -21 && (MODE == 15 || MODE == 21)) ?
                              sample_n33
1334                        : ( (selector == -21 && (MODE == 16 || MODE == 20)) ?
                              sample_n26
1335                        : ( (selector == -21 && (MODE == 17 || MODE == 19)) ?
                              sample_n21
1336                        : ( (selector == -21 &&  MODE == 18)            ?
                              sample_n17
1337                        : ( (selector == -20 && (MODE == 13 || MODE == 23)) ?
                              sample_n59
1338                        : ( (selector == -20 && (MODE == 14 || MODE == 22)) ?
                              sample_n41
1339                        : ( (selector == -20 && (MODE == 15 || MODE == 21)) ?
                              sample_n31
1340                        : ( (selector == -20 && (MODE == 16 || MODE == 20)) ?
                              sample_n25
1341                        : ( (selector == -20 && (MODE == 17 || MODE == 19)) ?
                              sample_n20
1342                        : ( (selector == -20 &&  MODE == 18)            ?
                              sample_n16
1343                        : ( (selector == -19 && (MODE == 13 || MODE == 23)) ?
                              sample_n56
1344                        : ( (selector == -19 && (MODE == 14 || MODE == 22)) ?
                              sample_n38
1345                        : ( (selector == -19 && (MODE == 15 || MODE == 21)) ?
                              sample_n29
1346                        : ( (selector == -19 && (MODE == 16 || MODE == 20)) ?
                              sample_n23
1347                        : ( (selector == -19 && (MODE == 17 || MODE == 19)) ?
                              sample_n19
1348                        : ( (selector == -19 &&  MODE == 18)            ?
                              sample_n15
1349                        : ( (selector == -18 && (MODE == 13 || MODE == 23)) ?
                              sample_n52
1350                        : ( (selector == -18 && (MODE == 14 || MODE == 22)) ?
                              sample_n36
1351                        : ( (selector == -18 && (MODE == 15 || MODE == 21)) ?
                              sample_n27
1352                        : ( (selector == -18 && (MODE == 16 || MODE == 20)) ?
```

```
                    sample_n22
1353              : ( (selector == -18 && (MODE == 17 || MODE == 19)) ?
                    sample_n17
1354              : ( (selector == -18 &&  MODE == 18)            ?
                    sample_n14
1355              : ( (selector == -17 && (MODE == 13 || MODE == 23)) ?
                    sample_n49
1356              : ( (selector == -17 && (MODE == 14 || MODE == 22)) ?
                    sample_n33
1357              : ( (selector == -17 && (MODE == 15 || MODE == 21)) ?
                    sample_n25
1358              : ( (selector == -17 && (MODE == 16 || MODE == 20)) ?
                    sample_n20
1359              : ( (selector == -17 && (MODE == 17 || MODE == 19)) ?
                    sample_n16
1360              : ( (selector == -17 &&  MODE == 18)            ?
                    sample_n13
1361              : ( (selector == -16 && (MODE == 13 || MODE == 23)) ?
                    sample_n45
1362              : ( (selector == -16 && (MODE == 14 || MODE == 22)) ?
                    sample_n31
1363              : ( (selector == -16 && (MODE == 15 || MODE == 21)) ?
                    sample_n23
1364              : ( (selector == -16 && (MODE == 16 || MODE == 20)) ?
                    sample_n19
1365              : ( (selector == -16 && (MODE == 17 || MODE == 19)) ?
                    sample_n15
1366              : ( (selector == -16 &&  MODE == 18)            ?
                    sample_n12
1367              : ( (selector == -15 && (MODE == 13 || MODE == 23)) ?
                    sample_n42
1368              : ( (selector == -15 && (MODE == 14 || MODE == 22)) ?
                    sample_n29
1369              : ( (selector == -15 && (MODE == 15 || MODE == 21)) ?
                    sample_n22
1370              : ( (selector == -15 && (MODE == 16 || MODE == 20)) ?
                    sample_n17
1371              : ( (selector == -15 && (MODE == 17 || MODE == 19)) ?
                    sample_n14
1372              : ( (selector == -15 &&  MODE == 18)            ?
                    sample_n11
1373              : ( (selector == -14 && (MODE == 13 || MODE == 23)) ?
                    sample_n38
1374              : ( (selector == -14 && (MODE == 14 || MODE == 22)) ?
                    sample_n26
1375              : ( (selector == -14 && (MODE == 15 || MODE == 21)) ?
                    sample_n20
1376              : ( (selector == -14 && (MODE == 16 || MODE == 20)) ?
                    sample_n16
1377              : ( (selector == -14 && (MODE == 17 || MODE == 19)) ?
                    sample_n13
1378              : ( (selector == -14 &&  MODE == 18)            ?
                    sample_n10
1379              : ( (selector == -13 && (MODE == 12 || MODE == 24)) ?
                    sample_n63
1380              : ( (selector == -13 && (MODE == 13 || MODE == 23)) ?
                    sample_n35
```

```
1381                        : ( (selector == -13 && (MODE == 14 || MODE == 22)) ?
                              sample_n24
1382                        : ( (selector == -13 && (MODE == 15 || MODE == 21)) ?
                              sample_n18
1383                        : ( (selector == -13 && (MODE == 16 || MODE == 20)) ?
                              sample_n14
1384                        : ( (selector == -13 && (MODE == 17 || MODE == 19)) ?
                              sample_n11
1385                        : ( (selector == -13 &&  MODE == 18)            ? sample_n9
1386                        : ( (selector == -12 && (MODE == 12 || MODE == 24)) ?
                              sample_n57
1387                        : ( (selector == -12 && (MODE == 13 || MODE == 23)) ?
                              sample_n31
1388                        : ( (selector == -12 && (MODE == 14 || MODE == 22)) ?
                              sample_n21
1389                        : ( (selector == -12 && (MODE == 15 || MODE == 21)) ?
                              sample_n16
1390                        : ( (selector == -12 && (MODE == 16 || MODE == 20)) ?
                              sample_n13
1391                        : ( (selector == -12 && (MODE == 17 || MODE == 19)) ?
                              sample_n10
1392                        : ( (selector == -12 &&  MODE == 18)            ? sample_n8
1393                        : ( (selector == -11 && (MODE == 12 || MODE == 24)) ?
                              sample_n50
1394                        : ( (selector == -11 && (MODE == 13 || MODE == 23)) ?
                              sample_n27
1395                        : ( (selector == -11 && (MODE == 14 || MODE == 22)) ?
                              sample_n19
1396                        : ( (selector == -11 && (MODE == 15 || MODE == 21)) ?
                              sample_n14
1397                        : ( (selector == -11 && (MODE == 16 || MODE == 20)) ?
                              sample_n11
1398                        : ( (selector == -11 && (MODE == 17 || MODE == 19)) ?
                              sample_n9
1399                        : ( (selector == -11 &&  MODE == 18)            ? sample_n7
1400                        : ( (selector == -10 && (MODE == 12 || MODE == 24)) ?
                              sample_n44
1401                        : ( (selector == -10 && (MODE == 13 || MODE == 23)) ?
                              sample_n24
1402                        : ( (selector == -10 && (MODE == 14 || MODE == 22)) ?
                              sample_n16
1403                        : ( (selector == -10 && (MODE == 15 || MODE == 21)) ?
                              sample_n12
1404                        : ( (selector == -10 && (MODE == 16 || MODE == 20)) ?
                              sample_n10
1405                        : ( (selector == -10 && (MODE == 17 || MODE == 19)) ?
                              sample_n8
1406                        : ( (selector == -10 &&  MODE == 18)            ? sample_n6
1407                        : ( (selector ==  -9 && (MODE == 12 || MODE == 24)) ?
                              sample_n37
1408                        : ( (selector ==  -9 && (MODE == 13 || MODE == 23)) ?
                              sample_n20
1409                        : ( (selector ==  -9 && (MODE == 14 || MODE == 22)) ?
                              sample_n14
1410                        : ( (selector ==  -9 && (MODE == 15 || MODE == 21)) ?
                              sample_n10
1411                        : ( (selector ==  -9 && (MODE == 16 || MODE == 20)) ?
```

```
                    sample_n8
1412              : ( (selector ==  -9 && (MODE == 17 || MODE == 19)) ?
                    sample_n6
1413              : ( (selector ==  -9 &&  MODE == 18)            ? sample_n5
1414              : ( (selector ==  -8 && (MODE == 12 || MODE == 24)) ?
                    sample_n31
1415              : ( (selector ==  -8 && (MODE == 13 || MODE == 23)) ?
                    sample_n17
1416              : ( (selector ==  -8 && (MODE == 14 || MODE == 22)) ?
                    sample_n11
1417              : ( (selector ==  -8 && (MODE == 15 || MODE == 21)) ?
                    sample_n8
1418              : ( (selector ==  -8 && (MODE == 16 || MODE == 20)) ?
                    sample_n7
1419              : ( (selector ==  -8 && (MODE == 17 || MODE == 19)) ?
                    sample_n5
1420              : ( (selector ==  -8 &&  MODE == 18)            ? sample_n4
1421              : ( (selector ==  -7 && (MODE == 11 || MODE == 25)) ?
                    sample_n63
1422              : ( (selector ==  -7 && (MODE == 12 || MODE == 24)) ?
                    sample_n25
1423              : ( (selector ==  -7 && (MODE == 13 || MODE == 23)) ?
                    sample_n13
1424              : ( (selector ==  -7 && (MODE == 14 || MODE == 22)) ?
                    sample_n9
1425              : ( (selector ==  -7 && (MODE == 15 || MODE == 21)) ?
                    sample_n7
1426              : ( (selector ==  -7 && (MODE == 16 || MODE == 20)) ?
                    sample_n5
1427              : ( (selector ==  -7 && (MODE == 17 || MODE == 19)) ?
                    sample_n4
1428              : ( (selector ==  -7 &&  MODE == 18)            ? sample_n3
1429              : ( (selector ==  -6 && (MODE == 11 || MODE == 25)) ?
                    sample_n47
1430              : ( (selector ==  -6 && (MODE == 12 || MODE == 24)) ?
                    sample_n18
1431              : ( (selector ==  -6 && (MODE == 13 || MODE == 23)) ?
                    sample_n10
1432              : ( (selector ==  -6 && (MODE == 14 || MODE == 22)) ?
                    sample_n6
1433              : ( (selector ==  -6 && (MODE == 15 || MODE == 21)) ?
                    sample_n5
1434              : ( (selector ==  -6 && (MODE == 16 || MODE == 20)) ?
                    sample_n4
1435              : ( (selector ==  -6 && (MODE == 17 || MODE == 19)) ?
                    sample_n3
1436              : ( (selector ==  -6 &&  MODE == 18)            ? sample_n2
1437              : ( (selector ==  -5 && (MODE == 11 || MODE == 25)) ?
                    sample_n31
1438              : ( (selector ==  -5 && (MODE == 12 || MODE == 24)) ?
                    sample_n12
1439              : ( (selector ==  -5 && (MODE == 13 || MODE == 23)) ?
                    sample_n6
1440              : ( (selector ==  -5 && (MODE == 14 || MODE == 22)) ?
                    sample_n4
1441              : ( (selector ==  -5 && (MODE == 15 || MODE == 21)) ?
                    sample_n3
```

```
1442                        : ( (selector ==  -5 && (MODE == 16 || MODE == 20)) ?
                              sample_n2
1443                        : ( (selector ==  -5 && (MODE == 17 || MODE == 19)) ?
                              sample_n1
1444                        : ( (selector ==  -5 &&  MODE == 18)            ? sample_n1
1445                        : ( (selector ==  -4 && (MODE == 11 || MODE == 25)) ?
                              sample_n15
1446                        : ( (selector ==  -4 && (MODE == 12 || MODE == 24)) ?
                              sample_n5
1447                        : ( (selector ==  -4 && (MODE == 13 || MODE == 23)) ?
                              sample_n3
1448                        : ( (selector ==  -4 && (MODE == 14 || MODE == 22)) ?
                              sample_n1
1449                        : ( (selector ==  -4 && (MODE == 15 || MODE == 21)) ?
                              sample_n1
1450                        : ( (selector ==  -4 && (MODE == 16 || MODE == 20)) ?
                              sample_n1
1451                        : ( (selector ==  -4 && (MODE == 17 || MODE == 19)) ?
                              sample_n0
1452                        : ( (selector ==  -4 &&  MODE == 18)            ? sample_n0
1453                        : (selector ==  -3 ? sample_NN
1454                        : (selector ==  -2 ? sample_0n
1455                        : (selector ==  -1 ? sample_1n
1456                        : (selector ==   0 ? sample_2n
1457                        : (selector ==   1 ? sample_3n
1458                        : (selector ==   2 ? sample_4n
1459                        : (selector ==   3 ? sample_5n
1460                        : (selector ==   4 ? sample_6n
1461                        : (selector ==   5 ? sample_7n
1462                        : (selector ==   6 ? sample_8n
1463                        : (selector ==   7 ? sample_9n
1464                        : (selector ==   8 ? sample_10n
1465                        : (selector ==   9 ? sample_11n
1466                        : (selector ==  10 ? sample_12n
1467                        : (selector ==  11 ? sample_13n
1468                        : (selector ==  12 ? sample_14n
1469                        : (selector ==  13 ? sample_15n
1470                        : (selector ==  14 ? sample_16n
1471                        : (selector ==  15 ? sample_17n
1472                        : (selector ==  16 ? sample_18n
1473                        : (selector ==  17 ? sample_19n
1474                        : (selector ==  18 ? sample_20n
1475                        : (selector ==  19 ? sample_21n
1476                        : (selector ==  20 ? sample_22n
1477                        : (selector ==  21 ? sample_23n
1478                        : (selector ==  22 ? sample_24n
1479                        : (selector ==  23 ? sample_25n
1480                        : (selector ==  24 ? sample_26n
1481                        : (selector ==  25 ? sample_27n
1482                        : (selector ==  26 ? sample_28n
1483                        : (selector ==  27 ? sample_29n
1484                        : (selector ==  28 ? sample_30n
1485                        : (selector ==  29 ? sample_31n
1486                        : (selector ==  30 ? sample_32n
1487                        : (selector ==  31 ? sample_33n
1488                        : (selector ==  32 ? sample_34n
1489                        : (selector ==  33 ? sample_35n
```

```
                      : (selector ==  34 ? sample_36n
                      : (selector ==  35 ? sample_37n
                      : (selector ==  36 ? sample_38n
                      : (selector ==  37 ? sample_39n
                      : (selector ==  38 ? sample_40n
                      : (selector ==  39 ? sample_41n
                      : (selector ==  40 ? sample_42n
                      : (selector ==  41 ? sample_43n
                      : (selector ==  42 ? sample_44n
                      : (selector ==  43 ? sample_45n
                      : (selector ==  44 ? sample_46n
                      : (selector ==  45 ? sample_47n
                      : (selector ==  46 ? sample_48n
                      : (selector ==  47 ? sample_49n
                      : (selector ==  48 ? sample_50n
                      : (selector ==  49 ? sample_51n
                      : (selector ==  50 ? sample_52n
                      : (selector ==  51 ? sample_53n
                      : (selector ==  52 ? sample_54n
                      : (selector ==  53 ? sample_55n
                      : (selector ==  54 ? sample_56n
                      : (selector ==  55 ? sample_57n
                      : (selector ==  56 ? sample_58n
                      : (selector ==  57 ? sample_59n
                      : (selector ==  58 ? sample_60n
                      : (selector ==  59 ? sample_61n
                      : (selector ==  60 ? sample_62n
                      : (selector ==  61 ? sample_63n
                      : 0 ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                         ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                         ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                         ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                         ))))))))))))))))))));

    assign ref4 = MODE < 18 ? ( (selector == -32 && (MODE == 15 || MODE == 21)) ?
         sample_52n
                      : ( (selector == -32 && (MODE == 16 || MODE == 20)) ?
                         sample_42n
                      : ( (selector == -32 && (MODE == 17 || MODE == 19)) ?
                         sample_33n
                      : ( (selector == -32 &&  MODE == 18)            ?
                         sample_27n
                      : ( (selector == -31 && (MODE == 15 || MODE == 21)) ?
                         sample_50n
                      : ( (selector == -31 && (MODE == 16 || MODE == 20)) ?
                         sample_40n
                      : ( (selector == -31 && (MODE == 17 || MODE == 19)) ?
                         sample_32n
                      : ( (selector == -31 &&  MODE == 18)            ?
                         sample_26n
                      : ( (selector == -30 && (MODE == 14 || MODE == 22)) ?
                         sample_63n
                      : ( (selector == -30 && (MODE == 15 || MODE == 21)) ?
                         sample_48n
                      : ( (selector == -30 && (MODE == 16 || MODE == 20)) ?
                         sample_39n
                      : ( (selector == -30 && (MODE == 17 || MODE == 19)) ?
```

```
                    sample_31n
1532              : ( (selector == -30 &&  MODE == 18)            ?
                    sample_25n
1533              : ( (selector == -29 && (MODE == 14 || MODE == 22)) ?
                    sample_61n
1534              : ( (selector == -29 && (MODE == 15 || MODE == 21)) ?
                    sample_46n
1535              : ( (selector == -29 && (MODE == 16 || MODE == 20)) ?
                    sample_37n
1536              : ( (selector == -29 && (MODE == 17 || MODE == 19)) ?
                    sample_30n
1537              : ( (selector == -29 &&  MODE == 18)            ?
                    sample_24n
1538              : ( (selector == -28 && (MODE == 14 || MODE == 22)) ?
                    sample_58n
1539              : ( (selector == -28 && (MODE == 15 || MODE == 21)) ?
                    sample_44n
1540              : ( (selector == -28 && (MODE == 16 || MODE == 20)) ?
                    sample_36n
1541              : ( (selector == -28 && (MODE == 17 || MODE == 19)) ?
                    sample_29n
1542              : ( (selector == -28 &&  MODE == 18)            ?
                    sample_23n
1543              : ( (selector == -27 && (MODE == 14 || MODE == 22)) ?
                    sample_56n
1544              : ( (selector == -27 && (MODE == 15 || MODE == 21)) ?
                    sample_42n
1545              : ( (selector == -27 && (MODE == 16 || MODE == 20)) ?
                    sample_34n
1546              : ( (selector == -27 && (MODE == 17 || MODE == 19)) ?
                    sample_27n
1547              : ( (selector == -27 &&  MODE == 18)            ?
                    sample_22n
1548              : ( (selector == -26 && (MODE == 14 || MODE == 22)) ?
                    sample_53n
1549              : ( (selector == -26 && (MODE == 15 || MODE == 21)) ?
                    sample_40n
1550              : ( (selector == -26 && (MODE == 16 || MODE == 20)) ?
                    sample_33n
1551              : ( (selector == -26 && (MODE == 17 || MODE == 19)) ?
                    sample_26n
1552              : ( (selector == -26 &&  MODE == 18)            ?
                    sample_21n
1553              : ( (selector == -25 && (MODE == 14 || MODE == 22)) ?
                    sample_51n
1554              : ( (selector == -25 && (MODE == 15 || MODE == 21)) ?
                    sample_39n
1555              : ( (selector == -25 && (MODE == 16 || MODE == 20)) ?
                    sample_31n
1556              : ( (selector == -25 && (MODE == 17 || MODE == 19)) ?
                    sample_25n
1557              : ( (selector == -25 &&  MODE == 18)            ?
                    sample_20n
1558              : ( (selector == -24 && (MODE == 14 || MODE == 22)) ?
                    sample_48n
1559              : ( (selector == -24 && (MODE == 15 || MODE == 21)) ?
                    sample_37n
```

```
                        : ( (selector == -24 && (MODE == 16 || MODE == 20)) ?
                          sample_29n
                        : ( (selector == -24 && (MODE == 17 || MODE == 19)) ?
                          sample_24n
                        : ( (selector == -24 &&  MODE == 18)            ?
                          sample_19n
                        : ( (selector == -23 && (MODE == 14 || MODE == 22)) ?
                          sample_46n
                        : ( (selector == -23 && (MODE == 15 || MODE == 21)) ?
                          sample_35n
                        : ( (selector == -23 && (MODE == 16 || MODE == 20)) ?
                          sample_28n
                        : ( (selector == -23 && (MODE == 17 || MODE == 19)) ?
                          sample_22n
                        : ( (selector == -23 &&  MODE == 18)            ?
                          sample_18n
                        : ( (selector == -22 && (MODE == 13 || MODE == 23)) ?
                          sample_63n
                        : ( (selector == -22 && (MODE == 14 || MODE == 22)) ?
                          sample_43n
                        : ( (selector == -22 && (MODE == 15 || MODE == 21)) ?
                          sample_33n
                        : ( (selector == -22 && (MODE == 16 || MODE == 20)) ?
                          sample_26n
                        : ( (selector == -22 && (MODE == 17 || MODE == 19)) ?
                          sample_21n
                        : ( (selector == -22 &&  MODE == 18)            ?
                          sample_17n
                        : ( (selector == -21 && (MODE == 13 || MODE == 23)) ?
                          sample_59n
                        : ( (selector == -21 && (MODE == 14 || MODE == 22)) ?
                          sample_41n
                        : ( (selector == -21 && (MODE == 15 || MODE == 21)) ?
                          sample_31n
                        : ( (selector == -21 && (MODE == 16 || MODE == 20)) ?
                          sample_25n
                        : ( (selector == -21 && (MODE == 17 || MODE == 19)) ?
                          sample_20n
                        : ( (selector == -21 &&  MODE == 18)            ?
                          sample_16n
                        : ( (selector == -20 && (MODE == 13 || MODE == 23)) ?
                          sample_56n
                        : ( (selector == -20 && (MODE == 14 || MODE == 22)) ?
                          sample_38n
                        : ( (selector == -20 && (MODE == 15 || MODE == 21)) ?
                          sample_29n
                        : ( (selector == -20 && (MODE == 16 || MODE == 20)) ?
                          sample_23n
                        : ( (selector == -20 && (MODE == 17 || MODE == 19)) ?
                          sample_19n
                        : ( (selector == -20 &&  MODE == 18)            ?
                          sample_15n
                        : ( (selector == -19 && (MODE == 13 || MODE == 23)) ?
                          sample_52n
                        : ( (selector == -19 && (MODE == 14 || MODE == 22)) ?
                          sample_36n
                        : ( (selector == -19 && (MODE == 15 || MODE == 21)) ?
```

```
                    sample_27n
1589                : ( (selector == -19 && (MODE == 16 || MODE == 20)) ?
                    sample_22n
1590                : ( (selector == -19 && (MODE == 17 || MODE == 19)) ?
                    sample_17n
1591                : ( (selector == -19 &&  MODE == 18)            ?
                    sample_14n
1592                : ( (selector == -18 && (MODE == 13 || MODE == 23)) ?
                    sample_49n
1593                : ( (selector == -18 && (MODE == 14 || MODE == 22)) ?
                    sample_33n
1594                : ( (selector == -18 && (MODE == 15 || MODE == 21)) ?
                    sample_25n
1595                : ( (selector == -18 && (MODE == 16 || MODE == 20)) ?
                    sample_20n
1596                : ( (selector == -18 && (MODE == 17 || MODE == 19)) ?
                    sample_16n
1597                : ( (selector == -18 &&  MODE == 18)            ?
                    sample_13n
1598                : ( (selector == -17 && (MODE == 13 || MODE == 23)) ?
                    sample_45n
1599                : ( (selector == -17 && (MODE == 14 || MODE == 22)) ?
                    sample_31n
1600                : ( (selector == -17 && (MODE == 15 || MODE == 21)) ?
                    sample_23n
1601                : ( (selector == -17 && (MODE == 16 || MODE == 20)) ?
                    sample_19n
1602                : ( (selector == -17 && (MODE == 17 || MODE == 19)) ?
                    sample_15n
1603                : ( (selector == -17 &&  MODE == 18)            ?
                    sample_12n
1604                : ( (selector == -16 && (MODE == 13 || MODE == 23)) ?
                    sample_42n
1605                : ( (selector == -16 && (MODE == 14 || MODE == 22)) ?
                    sample_29n
1606                : ( (selector == -16 && (MODE == 15 || MODE == 21)) ?
                    sample_22n
1607                : ( (selector == -16 && (MODE == 16 || MODE == 20)) ?
                    sample_17n
1608                : ( (selector == -16 && (MODE == 17 || MODE == 19)) ?
                    sample_14n
1609                : ( (selector == -16 &&  MODE == 18)            ?
                    sample_11n
1610                : ( (selector == -15 && (MODE == 13 || MODE == 23)) ?
                    sample_38n
1611                : ( (selector == -15 && (MODE == 14 || MODE == 22)) ?
                    sample_26n
1612                : ( (selector == -15 && (MODE == 15 || MODE == 21)) ?
                    sample_20n
1613                : ( (selector == -15 && (MODE == 16 || MODE == 20)) ?
                    sample_16n
1614                : ( (selector == -15 && (MODE == 17 || MODE == 19)) ?
                    sample_13n
1615                : ( (selector == -15 &&  MODE == 18)            ?
                    sample_10n
1616                : ( (selector == -14 && (MODE == 12 || MODE == 24)) ?
                    sample_63n
```

```
1617                        : ( (selector == -14 && (MODE == 13 || MODE == 23)) ?
                              sample_35n
1618                        : ( (selector == -14 && (MODE == 14 || MODE == 22)) ?
                              sample_24n
1619                        : ( (selector == -14 && (MODE == 15 || MODE == 21)) ?
                              sample_18n
1620                        : ( (selector == -14 && (MODE == 16 || MODE == 20)) ?
                              sample_14n
1621                        : ( (selector == -14 && (MODE == 17 || MODE == 19)) ?
                              sample_11n
1622                        : ( (selector == -14 &&  MODE == 18)            ? sample_9n
1623                        : ( (selector == -13 && (MODE == 12 || MODE == 24)) ?
                              sample_57n
1624                        : ( (selector == -13 && (MODE == 13 || MODE == 23)) ?
                              sample_31n
1625                        : ( (selector == -13 && (MODE == 14 || MODE == 22)) ?
                              sample_21n
1626                        : ( (selector == -13 && (MODE == 15 || MODE == 21)) ?
                              sample_16n
1627                        : ( (selector == -13 && (MODE == 16 || MODE == 20)) ?
                              sample_13n
1628                        : ( (selector == -13 && (MODE == 17 || MODE == 19)) ?
                              sample_10n
1629                        : ( (selector == -13 &&  MODE == 18)            ? sample_8n
1630                        : ( (selector == -12 && (MODE == 12 || MODE == 24)) ?
                              sample_50n
1631                        : ( (selector == -12 && (MODE == 13 || MODE == 23)) ?
                              sample_27n
1632                        : ( (selector == -12 && (MODE == 14 || MODE == 22)) ?
                              sample_19n
1633                        : ( (selector == -12 && (MODE == 15 || MODE == 21)) ?
                              sample_14n
1634                        : ( (selector == -12 && (MODE == 16 || MODE == 20)) ?
                              sample_11n
1635                        : ( (selector == -12 && (MODE == 17 || MODE == 19)) ?
                              sample_9n
1636                        : ( (selector == -12 &&  MODE == 18)            ? sample_7n
1637                        : ( (selector == -11 && (MODE == 12 || MODE == 24)) ?
                              sample_44n
1638                        : ( (selector == -11 && (MODE == 13 || MODE == 23)) ?
                              sample_24n
1639                        : ( (selector == -11 && (MODE == 14 || MODE == 22)) ?
                              sample_16n
1640                        : ( (selector == -11 && (MODE == 15 || MODE == 21)) ?
                              sample_12n
1641                        : ( (selector == -11 && (MODE == 16 || MODE == 20)) ?
                              sample_10n
1642                        : ( (selector == -11 && (MODE == 17 || MODE == 19)) ?
                              sample_8n
1643                        : ( (selector == -11 &&  MODE == 18)            ? sample_6n
1644                        : ( (selector == -10 && (MODE == 12 || MODE == 24)) ?
                              sample_37n
1645                        : ( (selector == -10 && (MODE == 13 || MODE == 23)) ?
                              sample_20n
1646                        : ( (selector == -10 && (MODE == 14 || MODE == 22)) ?
                              sample_14n
1647                        : ( (selector == -10 && (MODE == 15 || MODE == 21)) ?
```

```
                    sample_10n
1648              : ( (selector == -10 && (MODE == 16 || MODE == 20)) ?
                    sample_8n
1649              : ( (selector == -10 && (MODE == 17 || MODE == 19)) ?
                    sample_6n
1650              : ( (selector == -10 &&  MODE == 18)            ? sample_5n
1651              : ( (selector ==  -9 && (MODE == 12 || MODE == 24)) ?
                    sample_31n
1652              : ( (selector ==  -9 && (MODE == 13 || MODE == 23)) ?
                    sample_17n
1653              : ( (selector ==  -9 && (MODE == 14 || MODE == 22)) ?
                    sample_11n
1654              : ( (selector ==  -9 && (MODE == 15 || MODE == 21)) ?
                    sample_8n
1655              : ( (selector ==  -9 && (MODE == 16 || MODE == 20)) ?
                    sample_7n
1656              : ( (selector ==  -9 && (MODE == 17 || MODE == 19)) ?
                    sample_5n
1657              : ( (selector ==  -9 &&  MODE == 18)            ? sample_4n
1658              : ( (selector ==  -8 && (MODE == 11 || MODE == 25)) ?
                    sample_63n
1659              : ( (selector ==  -8 && (MODE == 12 || MODE == 24)) ?
                    sample_25n
1660              : ( (selector ==  -8 && (MODE == 13 || MODE == 23)) ?
                    sample_13n
1661              : ( (selector ==  -8 && (MODE == 14 || MODE == 22)) ?
                    sample_9n
1662              : ( (selector ==  -8 && (MODE == 15 || MODE == 21)) ?
                    sample_7n
1663              : ( (selector ==  -8 && (MODE == 16 || MODE == 20)) ?
                    sample_5n
1664              : ( (selector ==  -8 && (MODE == 17 || MODE == 19)) ?
                    sample_4n
1665              : ( (selector ==  -8 &&  MODE == 18)            ? sample_3n
1666              : ( (selector ==  -7 && (MODE == 11 || MODE == 25)) ?
                    sample_47n
1667              : ( (selector ==  -7 && (MODE == 12 || MODE == 24)) ?
                    sample_18n
1668              : ( (selector ==  -7 && (MODE == 13 || MODE == 23)) ?
                    sample_10n
1669              : ( (selector ==  -7 && (MODE == 14 || MODE == 22)) ?
                    sample_6n
1670              : ( (selector ==  -7 && (MODE == 15 || MODE == 21)) ?
                    sample_5n
1671              : ( (selector ==  -7 && (MODE == 16 || MODE == 20)) ?
                    sample_4n
1672              : ( (selector ==  -7 && (MODE == 17 || MODE == 19)) ?
                    sample_3n
1673              : ( (selector ==  -7 &&  MODE == 18)            ? sample_2n
1674              : ( (selector ==  -6 && (MODE == 11 || MODE == 25)) ?
                    sample_31n
1675              : ( (selector ==  -6 && (MODE == 12 || MODE == 24)) ?
                    sample_12n
1676              : ( (selector ==  -6 && (MODE == 13 || MODE == 23)) ?
                    sample_6n
1677              : ( (selector ==  -6 && (MODE == 14 || MODE == 22)) ?
                    sample_4n
```

```
                        : ( (selector ==  -6 && (MODE == 15 || MODE == 21)) ?
                          sample_3n
                        : ( (selector ==  -6 && (MODE == 16 || MODE == 20)) ?
                          sample_2n
                        : ( (selector ==  -6 && (MODE == 17 || MODE == 19)) ?
                          sample_1n
                        : ( (selector ==  -6 &&  MODE == 18)            ? sample_1n
                        : ( (selector ==  -5 && (MODE == 11 || MODE == 25)) ?
                          sample_15n
                        : ( (selector ==  -5 && (MODE == 12 || MODE == 24)) ?
                          sample_5n
                        : ( (selector ==  -5 && (MODE == 13 || MODE == 23)) ?
                          sample_3n
                        : ( (selector ==  -5 && (MODE == 14 || MODE == 22)) ?
                          sample_1n
                        : ( (selector ==  -5 && (MODE == 15 || MODE == 21)) ?
                          sample_1n
                        : ( (selector ==  -5 && (MODE == 16 || MODE == 20)) ?
                          sample_1n
                        : ( (selector ==  -5 && (MODE == 17 || MODE == 19)) ?
                          sample_0n
                        : ( (selector ==  -5 &&  MODE == 18)            ? sample_0n
                        : (selector ==  -4 ? sample_NN
                        : (selector ==  -3 ? sample_n0
                        : (selector ==  -2 ? sample_n1
                        : (selector ==  -1 ? sample_n2
                        : (selector ==   0 ? sample_n3
                        : (selector ==   1 ? sample_n4
                        : (selector ==   2 ? sample_n5
                        : (selector ==   3 ? sample_n6
                        : (selector ==   4 ? sample_n7
                        : (selector ==   5 ? sample_n8
                        : (selector ==   6 ? sample_n9
                        : (selector ==   7 ? sample_n10
                        : (selector ==   8 ? sample_n11
                        : (selector ==   9 ? sample_n12
                        : (selector ==  10 ? sample_n13
                        : (selector ==  11 ? sample_n14
                        : (selector ==  12 ? sample_n15
                        : (selector ==  13 ? sample_n16
                        : (selector ==  14 ? sample_n17
                        : (selector ==  15 ? sample_n18
                        : (selector ==  16 ? sample_n19
                        : (selector ==  17 ? sample_n20
                        : (selector ==  18 ? sample_n21
                        : (selector ==  19 ? sample_n22
                        : (selector ==  20 ? sample_n23
                        : (selector ==  21 ? sample_n24
                        : (selector ==  22 ? sample_n25
                        : (selector ==  23 ? sample_n26
                        : (selector ==  24 ? sample_n27
                        : (selector ==  25 ? sample_n28
                        : (selector ==  26 ? sample_n29
                        : (selector ==  27 ? sample_n30
                        : (selector ==  28 ? sample_n31
                        : (selector ==  29 ? sample_n32
                        : (selector ==  30 ? sample_n33
```

```
: (selector ==  31 ? sample_n34
: (selector ==  32 ? sample_n35
: (selector ==  33 ? sample_n36
: (selector ==  34 ? sample_n37
: (selector ==  35 ? sample_n38
: (selector ==  36 ? sample_n39
: (selector ==  37 ? sample_n40
: (selector ==  38 ? sample_n41
: (selector ==  39 ? sample_n42
: (selector ==  40 ? sample_n43
: (selector ==  41 ? sample_n44
: (selector ==  42 ? sample_n45
: (selector ==  43 ? sample_n46
: (selector ==  44 ? sample_n47
: (selector ==  45 ? sample_n48
: (selector ==  46 ? sample_n49
: (selector ==  47 ? sample_n50
: (selector ==  48 ? sample_n51
: (selector ==  49 ? sample_n52
: (selector ==  50 ? sample_n53
: (selector ==  51 ? sample_n54
: (selector ==  52 ? sample_n55
: (selector ==  53 ? sample_n56
: (selector ==  54 ? sample_n57
: (selector ==  55 ? sample_n58
: (selector ==  56 ? sample_n59
: (selector ==  57 ? sample_n60
: (selector ==  58 ? sample_n61
: (selector ==  59 ? sample_n62
: (selector ==  60 ? sample_n63
: 0 )))))))))))))))))))))))))))))))))))))))))))))))))))))))
  )))))))))))))))))))))))))))))))))))))))))))))))))))))))
  )))))))))))))))))))))))))))))))))))))))))))))))))))))))
  )))))))))))))))))))))))))))))))))))))))))))))))))))))))
  )))))))))))))))
: ( (selector == -32 && (MODE == 15 || MODE == 21)) ?
  sample_n52
: ( (selector == -32 && (MODE == 16 || MODE == 20)) ?
  sample_n42
: ( (selector == -32 && (MODE == 17 || MODE == 19)) ?
  sample_n33
: ( (selector == -32 &&  MODE == 18)             ?
  sample_n27
: ( (selector == -31 && (MODE == 15 || MODE == 21)) ?
  sample_n50
: ( (selector == -31 && (MODE == 16 || MODE == 20)) ?
  sample_n40
: ( (selector == -31 && (MODE == 17 || MODE == 19)) ?
  sample_n32
: ( (selector == -31 &&  MODE == 18)             ?
  sample_n26
: ( (selector == -30 && (MODE == 14 || MODE == 22)) ?
  sample_n63
: ( (selector == -30 && (MODE == 15 || MODE == 21)) ?
  sample_n48
: ( (selector == -30 && (MODE == 16 || MODE == 20)) ?
  sample_n39
```

```
                        : ( (selector == -30 && (MODE == 17 || MODE == 19)) ?
                          sample_n31
                        : ( (selector == -30 &&  MODE == 18)            ?
                          sample_n25
                        : ( (selector == -29 && (MODE == 14 || MODE == 22)) ?
                          sample_n61
                        : ( (selector == -29 && (MODE == 15 || MODE == 21)) ?
                          sample_n46
                        : ( (selector == -29 && (MODE == 16 || MODE == 20)) ?
                          sample_n37
                        : ( (selector == -29 && (MODE == 17 || MODE == 19)) ?
                          sample_n30
                        : ( (selector == -29 &&  MODE == 18)            ?
                          sample_n24
                        : ( (selector == -28 && (MODE == 14 || MODE == 22)) ?
                          sample_n58
                        : ( (selector == -28 && (MODE == 15 || MODE == 21)) ?
                          sample_n44
                        : ( (selector == -28 && (MODE == 16 || MODE == 20)) ?
                          sample_n36
                        : ( (selector == -28 && (MODE == 17 || MODE == 19)) ?
                          sample_n29
                        : ( (selector == -28 &&  MODE == 18)            ?
                          sample_n23
                        : ( (selector == -27 && (MODE == 14 || MODE == 22)) ?
                          sample_n56
                        : ( (selector == -27 && (MODE == 15 || MODE == 21)) ?
                          sample_n42
                        : ( (selector == -27 && (MODE == 16 || MODE == 20)) ?
                          sample_n34
                        : ( (selector == -27 && (MODE == 17 || MODE == 19)) ?
                          sample_n27
                        : ( (selector == -27 &&  MODE == 18)            ?
                          sample_n22
                        : ( (selector == -26 && (MODE == 14 || MODE == 22)) ?
                          sample_n53
                        : ( (selector == -26 && (MODE == 15 || MODE == 21)) ?
                          sample_n40
                        : ( (selector == -26 && (MODE == 16 || MODE == 20)) ?
                          sample_n33
                        : ( (selector == -26 && (MODE == 17 || MODE == 19)) ?
                          sample_n26
                        : ( (selector == -26 &&  MODE == 18)            ?
                          sample_n21
                        : ( (selector == -25 && (MODE == 14 || MODE == 22)) ?
                          sample_n51
                        : ( (selector == -25 && (MODE == 15 || MODE == 21)) ?
                          sample_n39
                        : ( (selector == -25 && (MODE == 16 || MODE == 20)) ?
                          sample_n31
                        : ( (selector == -25 && (MODE == 17 || MODE == 19)) ?
                          sample_n25
                        : ( (selector == -25 &&  MODE == 18)            ?
                          sample_n20
                        : ( (selector == -24 && (MODE == 14 || MODE == 22)) ?
                          sample_n48
                        : ( (selector == -24 && (MODE == 15 || MODE == 21)) ?
```

```
                    sample_n37
1796                : ( (selector == -24 && (MODE == 16 || MODE == 20)) ?
                    sample_n29
1797                : ( (selector == -24 && (MODE == 17 || MODE == 19)) ?
                    sample_n24
1798                : ( (selector == -24 &&  MODE == 18)            ?
                    sample_n19
1799                : ( (selector == -23 && (MODE == 14 || MODE == 22)) ?
                    sample_n46
1800                : ( (selector == -23 && (MODE == 15 || MODE == 21)) ?
                    sample_n35
1801                : ( (selector == -23 && (MODE == 16 || MODE == 20)) ?
                    sample_n28
1802                : ( (selector == -23 && (MODE == 17 || MODE == 19)) ?
                    sample_n22
1803                : ( (selector == -23 &&  MODE == 18)            ?
                    sample_n18
1804                : ( (selector == -22 && (MODE == 13 || MODE == 23)) ?
                    sample_n63
1805                : ( (selector == -22 && (MODE == 14 || MODE == 22)) ?
                    sample_n43
1806                : ( (selector == -22 && (MODE == 15 || MODE == 21)) ?
                    sample_n33
1807                : ( (selector == -22 && (MODE == 16 || MODE == 20)) ?
                    sample_n26
1808                : ( (selector == -22 && (MODE == 17 || MODE == 19)) ?
                    sample_n21
1809                : ( (selector == -22 &&  MODE == 18)            ?
                    sample_n17
1810                : ( (selector == -21 && (MODE == 13 || MODE == 23)) ?
                    sample_n59
1811                : ( (selector == -21 && (MODE == 14 || MODE == 22)) ?
                    sample_n41
1812                : ( (selector == -21 && (MODE == 15 || MODE == 21)) ?
                    sample_n31
1813                : ( (selector == -21 && (MODE == 16 || MODE == 20)) ?
                    sample_n25
1814                : ( (selector == -21 && (MODE == 17 || MODE == 19)) ?
                    sample_n20
1815                : ( (selector == -21 &&  MODE == 18)            ?
                    sample_n16
1816                : ( (selector == -20 && (MODE == 13 || MODE == 23)) ?
                    sample_n56
1817                : ( (selector == -20 && (MODE == 14 || MODE == 22)) ?
                    sample_n38
1818                : ( (selector == -20 && (MODE == 15 || MODE == 21)) ?
                    sample_n29
1819                : ( (selector == -20 && (MODE == 16 || MODE == 20)) ?
                    sample_n23
1820                : ( (selector == -20 && (MODE == 17 || MODE == 19)) ?
                    sample_n19
1821                : ( (selector == -20 &&  MODE == 18)            ?
                    sample_n15
1822                : ( (selector == -19 && (MODE == 13 || MODE == 23)) ?
                    sample_n52
1823                : ( (selector == -19 && (MODE == 14 || MODE == 22)) ?
                    sample_n36
```

```
                        : ( (selector == -19 && (MODE == 15 || MODE == 21)) ?
                          sample_n27
                        : ( (selector == -19 && (MODE == 16 || MODE == 20)) ?
                          sample_n22
                        : ( (selector == -19 && (MODE == 17 || MODE == 19)) ?
                          sample_n17
                        : ( (selector == -19 &&  MODE == 18)            ?
                          sample_n14
                        : ( (selector == -18 && (MODE == 13 || MODE == 23)) ?
                          sample_n49
                        : ( (selector == -18 && (MODE == 14 || MODE == 22)) ?
                          sample_n33
                        : ( (selector == -18 && (MODE == 15 || MODE == 21)) ?
                          sample_n25
                        : ( (selector == -18 && (MODE == 16 || MODE == 20)) ?
                          sample_n20
                        : ( (selector == -18 && (MODE == 17 || MODE == 19)) ?
                          sample_n16
                        : ( (selector == -18 &&  MODE == 18)            ?
                          sample_n13
                        : ( (selector == -17 && (MODE == 13 || MODE == 23)) ?
                          sample_n45
                        : ( (selector == -17 && (MODE == 14 || MODE == 22)) ?
                          sample_n31
                        : ( (selector == -17 && (MODE == 15 || MODE == 21)) ?
                          sample_n23
                        : ( (selector == -17 && (MODE == 16 || MODE == 20)) ?
                          sample_n19
                        : ( (selector == -17 && (MODE == 17 || MODE == 19)) ?
                          sample_n15
                        : ( (selector == -17 &&  MODE == 18)            ?
                          sample_n12
                        : ( (selector == -16 && (MODE == 13 || MODE == 23)) ?
                          sample_n42
                        : ( (selector == -16 && (MODE == 14 || MODE == 22)) ?
                          sample_n29
                        : ( (selector == -16 && (MODE == 15 || MODE == 21)) ?
                          sample_n22
                        : ( (selector == -16 && (MODE == 16 || MODE == 20)) ?
                          sample_n17
                        : ( (selector == -16 && (MODE == 17 || MODE == 19)) ?
                          sample_n14
                        : ( (selector == -16 &&  MODE == 18)            ?
                          sample_n11
                        : ( (selector == -15 && (MODE == 13 || MODE == 23)) ?
                          sample_n38
                        : ( (selector == -15 && (MODE == 14 || MODE == 22)) ?
                          sample_n26
                        : ( (selector == -15 && (MODE == 15 || MODE == 21)) ?
                          sample_n20
                        : ( (selector == -15 && (MODE == 16 || MODE == 20)) ?
                          sample_n16
                        : ( (selector == -15 && (MODE == 17 || MODE == 19)) ?
                          sample_n13
                        : ( (selector == -15 &&  MODE == 18)            ?
                          sample_n10
                        : ( (selector == -14 && (MODE == 12 || MODE == 24)) ?
```

```
                    sample_n63
1853                : ( (selector == -14 && (MODE == 13 || MODE == 23)) ?
                    sample_n35
1854                : ( (selector == -14 && (MODE == 14 || MODE == 22)) ?
                    sample_n24
1855                : ( (selector == -14 && (MODE == 15 || MODE == 21)) ?
                    sample_n18
1856                : ( (selector == -14 && (MODE == 16 || MODE == 20)) ?
                    sample_n14
1857                : ( (selector == -14 && (MODE == 17 || MODE == 19)) ?
                    sample_n11
1858                : ( (selector == -14 &&  MODE == 18)            ? sample_n9
1859                : ( (selector == -13 && (MODE == 12 || MODE == 24)) ?
                    sample_n57
1860                : ( (selector == -13 && (MODE == 13 || MODE == 23)) ?
                    sample_n31
1861                : ( (selector == -13 && (MODE == 14 || MODE == 22)) ?
                    sample_n21
1862                : ( (selector == -13 && (MODE == 15 || MODE == 21)) ?
                    sample_n16
1863                : ( (selector == -13 && (MODE == 16 || MODE == 20)) ?
                    sample_n13
1864                : ( (selector == -13 && (MODE == 17 || MODE == 19)) ?
                    sample_n10
1865                : ( (selector == -13 &&  MODE == 18)            ? sample_n8
1866                : ( (selector == -12 && (MODE == 12 || MODE == 24)) ?
                    sample_n50
1867                : ( (selector == -12 && (MODE == 13 || MODE == 23)) ?
                    sample_n27
1868                : ( (selector == -12 && (MODE == 14 || MODE == 22)) ?
                    sample_n19
1869                : ( (selector == -12 && (MODE == 15 || MODE == 21)) ?
                    sample_n14
1870                : ( (selector == -12 && (MODE == 16 || MODE == 20)) ?
                    sample_n11
1871                : ( (selector == -12 && (MODE == 17 || MODE == 19)) ?
                    sample_n9
1872                : ( (selector == -12 &&  MODE == 18)            ? sample_n7
1873                : ( (selector == -11 && (MODE == 12 || MODE == 24)) ?
                    sample_n44
1874                : ( (selector == -11 && (MODE == 13 || MODE == 23)) ?
                    sample_n24
1875                : ( (selector == -11 && (MODE == 14 || MODE == 22)) ?
                    sample_n16
1876                : ( (selector == -11 && (MODE == 15 || MODE == 21)) ?
                    sample_n12
1877                : ( (selector == -11 && (MODE == 16 || MODE == 20)) ?
                    sample_n10
1878                : ( (selector == -11 && (MODE == 17 || MODE == 19)) ?
                    sample_n8
1879                : ( (selector == -11 &&  MODE == 18)            ? sample_n6
1880                : ( (selector == -10 && (MODE == 12 || MODE == 24)) ?
                    sample_n37
1881                : ( (selector == -10 && (MODE == 13 || MODE == 23)) ?
                    sample_n20
1882                : ( (selector == -10 && (MODE == 14 || MODE == 22)) ?
                    sample_n14
```

```
1883                    : ( (selector == -10 && (MODE == 15 || MODE == 21)) ?
                          sample_n10
1884                    : ( (selector == -10 && (MODE == 16 || MODE == 20)) ?
                          sample_n8
1885                    : ( (selector == -10 && (MODE == 17 || MODE == 19)) ?
                          sample_n6
1886                    : ( (selector == -10 &&  MODE == 18)            ? sample_n5
1887                    : ( (selector ==  -9 && (MODE == 12 || MODE == 24)) ?
                          sample_n31
1888                    : ( (selector ==  -9 && (MODE == 13 || MODE == 23)) ?
                          sample_n17
1889                    : ( (selector ==  -9 && (MODE == 14 || MODE == 22)) ?
                          sample_n11
1890                    : ( (selector ==  -9 && (MODE == 15 || MODE == 21)) ?
                          sample_n8
1891                    : ( (selector ==  -9 && (MODE == 16 || MODE == 20)) ?
                          sample_n7
1892                    : ( (selector ==  -9 && (MODE == 17 || MODE == 19)) ?
                          sample_n5
1893                    : ( (selector ==  -9 &&  MODE == 18)            ? sample_n4
1894                    : ( (selector ==  -8 && (MODE == 11 || MODE == 25)) ?
                          sample_n63
1895                    : ( (selector ==  -8 && (MODE == 12 || MODE == 24)) ?
                          sample_n25
1896                    : ( (selector ==  -8 && (MODE == 13 || MODE == 23)) ?
                          sample_n13
1897                    : ( (selector ==  -8 && (MODE == 14 || MODE == 22)) ?
                          sample_n9
1898                    : ( (selector ==  -8 && (MODE == 15 || MODE == 21)) ?
                          sample_n7
1899                    : ( (selector ==  -8 && (MODE == 16 || MODE == 20)) ?
                          sample_n5
1900                    : ( (selector ==  -8 && (MODE == 17 || MODE == 19)) ?
                          sample_n4
1901                    : ( (selector ==  -8 &&  MODE == 18)            ? sample_n3
1902                    : ( (selector ==  -7 && (MODE == 11 || MODE == 25)) ?
                          sample_n47
1903                    : ( (selector ==  -7 && (MODE == 12 || MODE == 24)) ?
                          sample_n18
1904                    : ( (selector ==  -7 && (MODE == 13 || MODE == 23)) ?
                          sample_n10
1905                    : ( (selector ==  -7 && (MODE == 14 || MODE == 22)) ?
                          sample_n6
1906                    : ( (selector ==  -7 && (MODE == 15 || MODE == 21)) ?
                          sample_n5
1907                    : ( (selector ==  -7 && (MODE == 16 || MODE == 20)) ?
                          sample_n4
1908                    : ( (selector ==  -7 && (MODE == 17 || MODE == 19)) ?
                          sample_n3
1909                    : ( (selector ==  -7 &&  MODE == 18)            ? sample_n2
1910                    : ( (selector ==  -6 && (MODE == 11 || MODE == 25)) ?
                          sample_n31
1911                    : ( (selector ==  -6 && (MODE == 12 || MODE == 24)) ?
                          sample_n12
1912                    : ( (selector ==  -6 && (MODE == 13 || MODE == 23)) ?
                          sample_n6
1913                    : ( (selector ==  -6 && (MODE == 14 || MODE == 22)) ?
```

```
                        sample_n4
1914                  : ( (selector ==  -6 && (MODE == 15 || MODE == 21)) ?
                        sample_n3
1915                  : ( (selector ==  -6 && (MODE == 16 || MODE == 20)) ?
                        sample_n2
1916                  : ( (selector ==  -6 && (MODE == 17 || MODE == 19)) ?
                        sample_n1
1917                  : ( (selector ==  -6 &&  MODE == 18)            ? sample_n1
1918                  : ( (selector ==  -5 && (MODE == 11 || MODE == 25)) ?
                        sample_n15
1919                  : ( (selector ==  -5 && (MODE == 12 || MODE == 24)) ?
                        sample_n5
1920                  : ( (selector ==  -5 && (MODE == 13 || MODE == 23)) ?
                        sample_n3
1921                  : ( (selector ==  -5 && (MODE == 14 || MODE == 22)) ?
                        sample_n1
1922                  : ( (selector ==  -5 && (MODE == 15 || MODE == 21)) ?
                        sample_n1
1923                  : ( (selector ==  -5 && (MODE == 16 || MODE == 20)) ?
                        sample_n1
1924                  : ( (selector ==  -5 && (MODE == 17 || MODE == 19)) ?
                        sample_n0
1925                  : ( (selector ==  -5 &&  MODE == 18)            ? sample_n0
1926                  : (selector ==  -4 ? sample_NN
1927                  : (selector ==  -3 ? sample_0n
1928                  : (selector ==  -2 ? sample_1n
1929                  : (selector ==  -1 ? sample_2n
1930                  : (selector ==   0 ? sample_3n
1931                  : (selector ==   1 ? sample_4n
1932                  : (selector ==   2 ? sample_5n
1933                  : (selector ==   3 ? sample_6n
1934                  : (selector ==   4 ? sample_7n
1935                  : (selector ==   5 ? sample_8n
1936                  : (selector ==   6 ? sample_9n
1937                  : (selector ==   7 ? sample_10n
1938                  : (selector ==   8 ? sample_11n
1939                  : (selector ==   9 ? sample_12n
1940                  : (selector ==  10 ? sample_13n
1941                  : (selector ==  11 ? sample_14n
1942                  : (selector ==  12 ? sample_15n
1943                  : (selector ==  13 ? sample_16n
1944                  : (selector ==  14 ? sample_17n
1945                  : (selector ==  15 ? sample_18n
1946                  : (selector ==  16 ? sample_19n
1947                  : (selector ==  17 ? sample_20n
1948                  : (selector ==  18 ? sample_21n
1949                  : (selector ==  19 ? sample_22n
1950                  : (selector ==  20 ? sample_23n
1951                  : (selector ==  21 ? sample_24n
1952                  : (selector ==  22 ? sample_25n
1953                  : (selector ==  23 ? sample_26n
1954                  : (selector ==  24 ? sample_27n
1955                  : (selector ==  25 ? sample_28n
1956                  : (selector ==  26 ? sample_29n
1957                  : (selector ==  27 ? sample_30n
1958                  : (selector ==  28 ? sample_31n
1959                  : (selector ==  29 ? sample_32n
```

```
                      : (selector ==  30 ? sample_33n
                      : (selector ==  31 ? sample_34n
                      : (selector ==  32 ? sample_35n
                      : (selector ==  33 ? sample_36n
                      : (selector ==  34 ? sample_37n
                      : (selector ==  35 ? sample_38n
                      : (selector ==  36 ? sample_39n
                      : (selector ==  37 ? sample_40n
                      : (selector ==  38 ? sample_41n
                      : (selector ==  39 ? sample_42n
                      : (selector ==  40 ? sample_43n
                      : (selector ==  41 ? sample_44n
                      : (selector ==  42 ? sample_45n
                      : (selector ==  43 ? sample_46n
                      : (selector ==  44 ? sample_47n
                      : (selector ==  45 ? sample_48n
                      : (selector ==  46 ? sample_49n
                      : (selector ==  47 ? sample_50n
                      : (selector ==  48 ? sample_51n
                      : (selector ==  49 ? sample_52n
                      : (selector ==  50 ? sample_53n
                      : (selector ==  51 ? sample_54n
                      : (selector ==  52 ? sample_55n
                      : (selector ==  53 ? sample_56n
                      : (selector ==  54 ? sample_57n
                      : (selector ==  55 ? sample_58n
                      : (selector ==  56 ? sample_59n
                      : (selector ==  57 ? sample_60n
                      : (selector ==  58 ? sample_61n
                      : (selector ==  59 ? sample_62n
                      : (selector ==  60 ? sample_63n
                      : 0 ))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                          ))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                          ))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                          ))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                          ))))))))))))))));

    assign ref5 = MODE < 18 ? ( (selector == -32 && (MODE == 15 || MODE == 21)) ?
        sample_50n
                      : ( (selector == -32 && (MODE == 16 || MODE == 20)) ?
                          sample_40n
                      : ( (selector == -32 && (MODE == 17 || MODE == 19)) ?
                          sample_32n
                      : ( (selector == -32 &&  MODE == 18)            ?
                          sample_26n
                      : ( (selector == -31 && (MODE == 14 || MODE == 22)) ?
                          sample_63n
                      : ( (selector == -31 && (MODE == 15 || MODE == 21)) ?
                          sample_48n
                      : ( (selector == -31 && (MODE == 16 || MODE == 20)) ?
                          sample_39n
                      : ( (selector == -31 && (MODE == 17 || MODE == 19)) ?
                          sample_31n
                      : ( (selector == -31 &&  MODE == 18)            ?
                          sample_25n
                      : ( (selector == -30 && (MODE == 14 || MODE == 22)) ?
                          sample_61n
```

```
2003                    : ( (selector == -30 && (MODE == 15 || MODE == 21)) ?
                          sample_46n
2004                    : ( (selector == -30 && (MODE == 16 || MODE == 20)) ?
                          sample_37n
2005                    : ( (selector == -30 && (MODE == 17 || MODE == 19)) ?
                          sample_30n
2006                    : ( (selector == -30 &&  MODE == 18)            ?
                          sample_24n
2007                    : ( (selector == -29 && (MODE == 14 || MODE == 22)) ?
                          sample_58n
2008                    : ( (selector == -29 && (MODE == 15 || MODE == 21)) ?
                          sample_44n
2009                    : ( (selector == -29 && (MODE == 16 || MODE == 20)) ?
                          sample_36n
2010                    : ( (selector == -29 && (MODE == 17 || MODE == 19)) ?
                          sample_29n
2011                    : ( (selector == -29 &&  MODE == 18)            ?
                          sample_23n
2012                    : ( (selector == -28 && (MODE == 14 || MODE == 22)) ?
                          sample_56n
2013                    : ( (selector == -28 && (MODE == 15 || MODE == 21)) ?
                          sample_42n
2014                    : ( (selector == -28 && (MODE == 16 || MODE == 20)) ?
                          sample_34n
2015                    : ( (selector == -28 && (MODE == 17 || MODE == 19)) ?
                          sample_27n
2016                    : ( (selector == -28 &&  MODE == 18)            ?
                          sample_22n
2017                    : ( (selector == -27 && (MODE == 14 || MODE == 22)) ?
                          sample_53n
2018                    : ( (selector == -27 && (MODE == 15 || MODE == 21)) ?
                          sample_40n
2019                    : ( (selector == -27 && (MODE == 16 || MODE == 20)) ?
                          sample_33n
2020                    : ( (selector == -27 && (MODE == 17 || MODE == 19)) ?
                          sample_26n
2021                    : ( (selector == -27 &&  MODE == 18)            ?
                          sample_21n
2022                    : ( (selector == -26 && (MODE == 14 || MODE == 22)) ?
                          sample_51n
2023                    : ( (selector == -26 && (MODE == 15 || MODE == 21)) ?
                          sample_39n
2024                    : ( (selector == -26 && (MODE == 16 || MODE == 20)) ?
                          sample_31n
2025                    : ( (selector == -26 && (MODE == 17 || MODE == 19)) ?
                          sample_25n
2026                    : ( (selector == -26 &&  MODE == 18)            ?
                          sample_20n
2027                    : ( (selector == -25 && (MODE == 14 || MODE == 22)) ?
                          sample_48n
2028                    : ( (selector == -25 && (MODE == 15 || MODE == 21)) ?
                          sample_37n
2029                    : ( (selector == -25 && (MODE == 16 || MODE == 20)) ?
                          sample_29n
2030                    : ( (selector == -25 && (MODE == 17 || MODE == 19)) ?
                          sample_24n
2031                    : ( (selector == -25 &&  MODE == 18)            ?
```

```
                    sample_19n
2032              : ( (selector == -24 && (MODE == 14 || MODE == 22)) ?
                    sample_46n
2033              : ( (selector == -24 && (MODE == 15 || MODE == 21)) ?
                    sample_35n
2034              : ( (selector == -24 && (MODE == 16 || MODE == 20)) ?
                    sample_28n
2035              : ( (selector == -24 && (MODE == 17 || MODE == 19)) ?
                    sample_22n
2036              : ( (selector == -24 &&  MODE == 18)            ?
                    sample_18n
2037              : ( (selector == -23 && (MODE == 13 || MODE == 23)) ?
                    sample_63n
2038              : ( (selector == -23 && (MODE == 14 || MODE == 22)) ?
                    sample_43n
2039              : ( (selector == -23 && (MODE == 15 || MODE == 21)) ?
                    sample_33n
2040              : ( (selector == -23 && (MODE == 16 || MODE == 20)) ?
                    sample_26n
2041              : ( (selector == -23 && (MODE == 17 || MODE == 19)) ?
                    sample_21n
2042              : ( (selector == -23 &&  MODE == 18)            ?
                    sample_17n
2043              : ( (selector == -22 && (MODE == 13 || MODE == 23)) ?
                    sample_59n
2044              : ( (selector == -22 && (MODE == 14 || MODE == 22)) ?
                    sample_41n
2045              : ( (selector == -22 && (MODE == 15 || MODE == 21)) ?
                    sample_31n
2046              : ( (selector == -22 && (MODE == 16 || MODE == 20)) ?
                    sample_25n
2047              : ( (selector == -22 && (MODE == 17 || MODE == 19)) ?
                    sample_20n
2048              : ( (selector == -22 &&  MODE == 18)            ?
                    sample_16n
2049              : ( (selector == -21 && (MODE == 13 || MODE == 23)) ?
                    sample_56n
2050              : ( (selector == -21 && (MODE == 14 || MODE == 22)) ?
                    sample_38n
2051              : ( (selector == -21 && (MODE == 15 || MODE == 21)) ?
                    sample_29n
2052              : ( (selector == -21 && (MODE == 16 || MODE == 20)) ?
                    sample_23n
2053              : ( (selector == -21 && (MODE == 17 || MODE == 19)) ?
                    sample_19n
2054              : ( (selector == -21 &&  MODE == 18)            ?
                    sample_15n
2055              : ( (selector == -20 && (MODE == 13 || MODE == 23)) ?
                    sample_52n
2056              : ( (selector == -20 && (MODE == 14 || MODE == 22)) ?
                    sample_36n
2057              : ( (selector == -20 && (MODE == 15 || MODE == 21)) ?
                    sample_27n
2058              : ( (selector == -20 && (MODE == 16 || MODE == 20)) ?
                    sample_22n
2059              : ( (selector == -20 && (MODE == 17 || MODE == 19)) ?
                    sample_17n
```

```
2060                        : ( (selector == -20 &&  MODE == 18)            ?
                              sample_14n
2061                        : ( (selector == -19 && (MODE == 13 || MODE == 23)) ?
                              sample_49n
2062                        : ( (selector == -19 && (MODE == 14 || MODE == 22)) ?
                              sample_33n
2063                        : ( (selector == -19 && (MODE == 15 || MODE == 21)) ?
                              sample_25n
2064                        : ( (selector == -19 && (MODE == 16 || MODE == 20)) ?
                              sample_20n
2065                        : ( (selector == -19 && (MODE == 17 || MODE == 19)) ?
                              sample_16n
2066                        : ( (selector == -19 &&  MODE == 18)            ?
                              sample_13n
2067                        : ( (selector == -18 && (MODE == 13 || MODE == 23)) ?
                              sample_45n
2068                        : ( (selector == -18 && (MODE == 14 || MODE == 22)) ?
                              sample_31n
2069                        : ( (selector == -18 && (MODE == 15 || MODE == 21)) ?
                              sample_23n
2070                        : ( (selector == -18 && (MODE == 16 || MODE == 20)) ?
                              sample_19n
2071                        : ( (selector == -18 && (MODE == 17 || MODE == 19)) ?
                              sample_15n
2072                        : ( (selector == -18 &&  MODE == 18)            ?
                              sample_12n
2073                        : ( (selector == -17 && (MODE == 13 || MODE == 23)) ?
                              sample_42n
2074                        : ( (selector == -17 && (MODE == 14 || MODE == 22)) ?
                              sample_29n
2075                        : ( (selector == -17 && (MODE == 15 || MODE == 21)) ?
                              sample_22n
2076                        : ( (selector == -17 && (MODE == 16 || MODE == 20)) ?
                              sample_17n
2077                        : ( (selector == -17 && (MODE == 17 || MODE == 19)) ?
                              sample_14n
2078                        : ( (selector == -17 &&  MODE == 18)            ?
                              sample_11n
2079                        : ( (selector == -16 && (MODE == 13 || MODE == 23)) ?
                              sample_38n
2080                        : ( (selector == -16 && (MODE == 14 || MODE == 22)) ?
                              sample_26n
2081                        : ( (selector == -16 && (MODE == 15 || MODE == 21)) ?
                              sample_20n
2082                        : ( (selector == -16 && (MODE == 16 || MODE == 20)) ?
                              sample_16n
2083                        : ( (selector == -16 && (MODE == 17 || MODE == 19)) ?
                              sample_13n
2084                        : ( (selector == -16 &&  MODE == 18)            ?
                              sample_10n
2085                        : ( (selector == -15 && (MODE == 12 || MODE == 24)) ?
                              sample_63n
2086                        : ( (selector == -15 && (MODE == 13 || MODE == 23)) ?
                              sample_35n
2087                        : ( (selector == -15 && (MODE == 14 || MODE == 22)) ?
                              sample_24n
2088                        : ( (selector == -15 && (MODE == 15 || MODE == 21)) ?
```

```
                    sample_18n
2089                : ( (selector == -15 && (MODE == 16 || MODE == 20)) ?
                    sample_14n
2090                : ( (selector == -15 && (MODE == 17 || MODE == 19)) ?
                    sample_11n
2091                : ( (selector == -15 &&  MODE == 18)            ? sample_9n
2092                : ( (selector == -14 && (MODE == 12 || MODE == 24)) ?
                    sample_57n
2093                : ( (selector == -14 && (MODE == 13 || MODE == 23)) ?
                    sample_31n
2094                : ( (selector == -14 && (MODE == 14 || MODE == 22)) ?
                    sample_21n
2095                : ( (selector == -14 && (MODE == 15 || MODE == 21)) ?
                    sample_16n
2096                : ( (selector == -14 && (MODE == 16 || MODE == 20)) ?
                    sample_13n
2097                : ( (selector == -14 && (MODE == 17 || MODE == 19)) ?
                    sample_10n
2098                : ( (selector == -14 &&  MODE == 18)            ? sample_8n
2099                : ( (selector == -13 && (MODE == 12 || MODE == 24)) ?
                    sample_50n
2100                : ( (selector == -13 && (MODE == 13 || MODE == 23)) ?
                    sample_27n
2101                : ( (selector == -13 && (MODE == 14 || MODE == 22)) ?
                    sample_19n
2102                : ( (selector == -13 && (MODE == 15 || MODE == 21)) ?
                    sample_14n
2103                : ( (selector == -13 && (MODE == 16 || MODE == 20)) ?
                    sample_11n
2104                : ( (selector == -13 && (MODE == 17 || MODE == 19)) ?
                    sample_9n
2105                : ( (selector == -13 &&  MODE == 18)            ? sample_7n
2106                : ( (selector == -12 && (MODE == 12 || MODE == 24)) ?
                    sample_44n
2107                : ( (selector == -12 && (MODE == 13 || MODE == 23)) ?
                    sample_24n
2108                : ( (selector == -12 && (MODE == 14 || MODE == 22)) ?
                    sample_16n
2109                : ( (selector == -12 && (MODE == 15 || MODE == 21)) ?
                    sample_12n
2110                : ( (selector == -12 && (MODE == 16 || MODE == 20)) ?
                    sample_10n
2111                : ( (selector == -12 && (MODE == 17 || MODE == 19)) ?
                    sample_8n
2112                : ( (selector == -12 &&  MODE == 18)            ? sample_6n
2113                : ( (selector == -11 && (MODE == 12 || MODE == 24)) ?
                    sample_37n
2114                : ( (selector == -11 && (MODE == 13 || MODE == 23)) ?
                    sample_20n
2115                : ( (selector == -11 && (MODE == 14 || MODE == 22)) ?
                    sample_14n
2116                : ( (selector == -11 && (MODE == 15 || MODE == 21)) ?
                    sample_10n
2117                : ( (selector == -11 && (MODE == 16 || MODE == 20)) ?
                    sample_8n
2118                : ( (selector == -11 && (MODE == 17 || MODE == 19)) ?
                    sample_6n
```

```
2119                        : ( (selector == -11 &&  MODE == 18)            ? sample_5n
2120                        : ( (selector == -10 && (MODE == 12 || MODE == 24)) ?
                              sample_31n
2121                        : ( (selector == -10 && (MODE == 13 || MODE == 23)) ?
                              sample_17n
2122                        : ( (selector == -10 && (MODE == 14 || MODE == 22)) ?
                              sample_11n
2123                        : ( (selector == -10 && (MODE == 15 || MODE == 21)) ?
                              sample_8n
2124                        : ( (selector == -10 && (MODE == 16 || MODE == 20)) ?
                              sample_7n
2125                        : ( (selector == -10 && (MODE == 17 || MODE == 19)) ?
                              sample_5n
2126                        : ( (selector == -10 &&  MODE == 18)            ? sample_4n
2127                        : ( (selector ==  -9 && (MODE == 11 || MODE == 25)) ?
                              sample_63n
2128                        : ( (selector ==  -9 && (MODE == 12 || MODE == 24)) ?
                              sample_25n
2129                        : ( (selector ==  -9 && (MODE == 13 || MODE == 23)) ?
                              sample_13n
2130                        : ( (selector ==  -9 && (MODE == 14 || MODE == 22)) ?
                              sample_9n
2131                        : ( (selector ==  -9 && (MODE == 15 || MODE == 21)) ?
                              sample_7n
2132                        : ( (selector ==  -9 && (MODE == 16 || MODE == 20)) ?
                              sample_5n
2133                        : ( (selector ==  -9 && (MODE == 17 || MODE == 19)) ?
                              sample_4n
2134                        : ( (selector ==  -9 &&  MODE == 18)            ? sample_3n
2135                        : ( (selector ==  -8 && (MODE == 11 || MODE == 25)) ?
                              sample_47n
2136                        : ( (selector ==  -8 && (MODE == 12 || MODE == 24)) ?
                              sample_18n
2137                        : ( (selector ==  -8 && (MODE == 13 || MODE == 23)) ?
                              sample_10n
2138                        : ( (selector ==  -8 && (MODE == 14 || MODE == 22)) ?
                              sample_6n
2139                        : ( (selector ==  -8 && (MODE == 15 || MODE == 21)) ?
                              sample_5n
2140                        : ( (selector ==  -8 && (MODE == 16 || MODE == 20)) ?
                              sample_4n
2141                        : ( (selector ==  -8 && (MODE == 17 || MODE == 19)) ?
                              sample_3n
2142                        : ( (selector ==  -8 &&  MODE == 18)            ? sample_2n
2143                        : ( (selector ==  -7 && (MODE == 11 || MODE == 25)) ?
                              sample_31n
2144                        : ( (selector ==  -7 && (MODE == 12 || MODE == 24)) ?
                              sample_12n
2145                        : ( (selector ==  -7 && (MODE == 13 || MODE == 23)) ?
                              sample_6n
2146                        : ( (selector ==  -7 && (MODE == 14 || MODE == 22)) ?
                              sample_4n
2147                        : ( (selector ==  -7 && (MODE == 15 || MODE == 21)) ?
                              sample_3n
2148                        : ( (selector ==  -7 && (MODE == 16 || MODE == 20)) ?
                              sample_2n
2149                        : ( (selector ==  -7 && (MODE == 17 || MODE == 19)) ?
```

```
                        sample_1n
2150                  : ( (selector ==  -7 &&  MODE == 18)            ? sample_1n
2151                  : ( (selector ==  -6 && (MODE == 11 || MODE == 25)) ?
                        sample_15n
2152                  : ( (selector ==  -6 && (MODE == 12 || MODE == 24)) ?
                        sample_5n
2153                  : ( (selector ==  -6 && (MODE == 13 || MODE == 23)) ?
                        sample_3n
2154                  : ( (selector ==  -6 && (MODE == 14 || MODE == 22)) ?
                        sample_1n
2155                  : ( (selector ==  -6 && (MODE == 15 || MODE == 21)) ?
                        sample_1n
2156                  : ( (selector ==  -6 && (MODE == 16 || MODE == 20)) ?
                        sample_1n
2157                  : ( (selector ==  -6 && (MODE == 17 || MODE == 19)) ?
                        sample_0n
2158                  : ( (selector ==  -6 &&  MODE == 18)            ? sample_0n
2159                  : (selector ==  -5 ? sample_NN
2160                  : (selector ==  -4 ? sample_n0
2161                  : (selector ==  -3 ? sample_n1
2162                  : (selector ==  -2 ? sample_n2
2163                  : (selector ==  -1 ? sample_n3
2164                  : (selector ==   0 ? sample_n4
2165                  : (selector ==   1 ? sample_n5
2166                  : (selector ==   2 ? sample_n6
2167                  : (selector ==   3 ? sample_n7
2168                  : (selector ==   4 ? sample_n8
2169                  : (selector ==   5 ? sample_n9
2170                  : (selector ==   6 ? sample_n10
2171                  : (selector ==   7 ? sample_n11
2172                  : (selector ==   8 ? sample_n12
2173                  : (selector ==   9 ? sample_n13
2174                  : (selector ==  10 ? sample_n14
2175                  : (selector ==  11 ? sample_n15
2176                  : (selector ==  12 ? sample_n16
2177                  : (selector ==  13 ? sample_n17
2178                  : (selector ==  14 ? sample_n18
2179                  : (selector ==  15 ? sample_n19
2180                  : (selector ==  16 ? sample_n20
2181                  : (selector ==  17 ? sample_n21
2182                  : (selector ==  18 ? sample_n22
2183                  : (selector ==  19 ? sample_n23
2184                  : (selector ==  20 ? sample_n24
2185                  : (selector ==  21 ? sample_n25
2186                  : (selector ==  22 ? sample_n26
2187                  : (selector ==  23 ? sample_n27
2188                  : (selector ==  24 ? sample_n28
2189                  : (selector ==  25 ? sample_n29
2190                  : (selector ==  26 ? sample_n30
2191                  : (selector ==  27 ? sample_n31
2192                  : (selector ==  28 ? sample_n32
2193                  : (selector ==  29 ? sample_n33
2194                  : (selector ==  30 ? sample_n34
2195                  : (selector ==  31 ? sample_n35
2196                  : (selector ==  32 ? sample_n36
2197                  : (selector ==  33 ? sample_n37
2198                  : (selector ==  34 ? sample_n38
```

```
                        : (selector ==  35 ? sample_n39
                        : (selector ==  36 ? sample_n40
                        : (selector ==  37 ? sample_n41
                        : (selector ==  38 ? sample_n42
                        : (selector ==  39 ? sample_n43
                        : (selector ==  40 ? sample_n44
                        : (selector ==  41 ? sample_n45
                        : (selector ==  42 ? sample_n46
                        : (selector ==  43 ? sample_n47
                        : (selector ==  44 ? sample_n48
                        : (selector ==  45 ? sample_n49
                        : (selector ==  46 ? sample_n50
                        : (selector ==  47 ? sample_n51
                        : (selector ==  48 ? sample_n52
                        : (selector ==  49 ? sample_n53
                        : (selector ==  50 ? sample_n54
                        : (selector ==  51 ? sample_n55
                        : (selector ==  52 ? sample_n56
                        : (selector ==  53 ? sample_n57
                        : (selector ==  54 ? sample_n58
                        : (selector ==  55 ? sample_n59
                        : (selector ==  56 ? sample_n60
                        : (selector ==  57 ? sample_n61
                        : (selector ==  58 ? sample_n62
                        : (selector ==  59 ? sample_n63
                        : 0 )))))))))))))))))))))))))))))))))))))))))))))))))))))))
                          )))))))))))))))))))))))))))))))))))))))))))))))))))))))
                          )))))))))))))))))))))))))))))))))))))))))))))))))))))))
                          )))))))))))))))))))))))))))))))))))))))))))))))))))))))
                          )))))))))))
                        : ( (selector == -32 && (MODE == 15 || MODE == 21)) ?
                          sample_n50
                        : ( (selector == -32 && (MODE == 16 || MODE == 20)) ?
                          sample_n40
                        : ( (selector == -32 && (MODE == 17 || MODE == 19)) ?
                          sample_n32
                        : ( (selector == -32 &&  MODE == 18)             ?
                          sample_n26
                        : ( (selector == -31 && (MODE == 14 || MODE == 22)) ?
                          sample_n63
                        : ( (selector == -31 && (MODE == 15 || MODE == 21)) ?
                          sample_n48
                        : ( (selector == -31 && (MODE == 16 || MODE == 20)) ?
                          sample_n39
                        : ( (selector == -31 && (MODE == 17 || MODE == 19)) ?
                          sample_n31
                        : ( (selector == -31 &&  MODE == 18)             ?
                          sample_n25
                        : ( (selector == -30 && (MODE == 14 || MODE == 22)) ?
                          sample_n61
                        : ( (selector == -30 && (MODE == 15 || MODE == 21)) ?
                          sample_n46
                        : ( (selector == -30 && (MODE == 16 || MODE == 20)) ?
                          sample_n37
                        : ( (selector == -30 && (MODE == 17 || MODE == 19)) ?
                          sample_n30
                        : ( (selector == -30 &&  MODE == 18)             ?
```

```
                    sample_n24
2239                : ( (selector == -29 && (MODE == 14 || MODE == 22)) ?
                    sample_n58
2240                : ( (selector == -29 && (MODE == 15 || MODE == 21)) ?
                    sample_n44
2241                : ( (selector == -29 && (MODE == 16 || MODE == 20)) ?
                    sample_n36
2242                : ( (selector == -29 && (MODE == 17 || MODE == 19)) ?
                    sample_n29
2243                : ( (selector == -29 &&  MODE == 18)            ?
                    sample_n23
2244                : ( (selector == -28 && (MODE == 14 || MODE == 22)) ?
                    sample_n56
2245                : ( (selector == -28 && (MODE == 15 || MODE == 21)) ?
                    sample_n42
2246                : ( (selector == -28 && (MODE == 16 || MODE == 20)) ?
                    sample_n34
2247                : ( (selector == -28 && (MODE == 17 || MODE == 19)) ?
                    sample_n27
2248                : ( (selector == -28 &&  MODE == 18)            ?
                    sample_n22
2249                : ( (selector == -27 && (MODE == 14 || MODE == 22)) ?
                    sample_n53
2250                : ( (selector == -27 && (MODE == 15 || MODE == 21)) ?
                    sample_n40
2251                : ( (selector == -27 && (MODE == 16 || MODE == 20)) ?
                    sample_n33
2252                : ( (selector == -27 && (MODE == 17 || MODE == 19)) ?
                    sample_n26
2253                : ( (selector == -27 &&  MODE == 18)            ?
                    sample_n21
2254                : ( (selector == -26 && (MODE == 14 || MODE == 22)) ?
                    sample_n51
2255                : ( (selector == -26 && (MODE == 15 || MODE == 21)) ?
                    sample_n39
2256                : ( (selector == -26 && (MODE == 16 || MODE == 20)) ?
                    sample_n31
2257                : ( (selector == -26 && (MODE == 17 || MODE == 19)) ?
                    sample_n25
2258                : ( (selector == -26 &&  MODE == 18)            ?
                    sample_n20
2259                : ( (selector == -25 && (MODE == 14 || MODE == 22)) ?
                    sample_n48
2260                : ( (selector == -25 && (MODE == 15 || MODE == 21)) ?
                    sample_n37
2261                : ( (selector == -25 && (MODE == 16 || MODE == 20)) ?
                    sample_n29
2262                : ( (selector == -25 && (MODE == 17 || MODE == 19)) ?
                    sample_n24
2263                : ( (selector == -25 &&  MODE == 18)            ?
                    sample_n19
2264                : ( (selector == -24 && (MODE == 14 || MODE == 22)) ?
                    sample_n46
2265                : ( (selector == -24 && (MODE == 15 || MODE == 21)) ?
                    sample_n35
2266                : ( (selector == -24 && (MODE == 16 || MODE == 20)) ?
                    sample_n28
```

```
                    : ( (selector == -24 && (MODE == 17 || MODE == 19)) ?
                      sample_n22
                    : ( (selector == -24 &&  MODE == 18)            ?
                      sample_n18
                    : ( (selector == -23 && (MODE == 13 || MODE == 23)) ?
                      sample_n63
                    : ( (selector == -23 && (MODE == 14 || MODE == 22)) ?
                      sample_n43
                    : ( (selector == -23 && (MODE == 15 || MODE == 21)) ?
                      sample_n33
                    : ( (selector == -23 && (MODE == 16 || MODE == 20)) ?
                      sample_n26
                    : ( (selector == -23 && (MODE == 17 || MODE == 19)) ?
                      sample_n21
                    : ( (selector == -23 &&  MODE == 18)            ?
                      sample_n17
                    : ( (selector == -22 && (MODE == 13 || MODE == 23)) ?
                      sample_n59
                    : ( (selector == -22 && (MODE == 14 || MODE == 22)) ?
                      sample_n41
                    : ( (selector == -22 && (MODE == 15 || MODE == 21)) ?
                      sample_n31
                    : ( (selector == -22 && (MODE == 16 || MODE == 20)) ?
                      sample_n25
                    : ( (selector == -22 && (MODE == 17 || MODE == 19)) ?
                      sample_n20
                    : ( (selector == -22 &&  MODE == 18)            ?
                      sample_n16
                    : ( (selector == -21 && (MODE == 13 || MODE == 23)) ?
                      sample_n56
                    : ( (selector == -21 && (MODE == 14 || MODE == 22)) ?
                      sample_n38
                    : ( (selector == -21 && (MODE == 15 || MODE == 21)) ?
                      sample_n29
                    : ( (selector == -21 && (MODE == 16 || MODE == 20)) ?
                      sample_n23
                    : ( (selector == -21 && (MODE == 17 || MODE == 19)) ?
                      sample_n19
                    : ( (selector == -21 &&  MODE == 18)            ?
                      sample_n15
                    : ( (selector == -20 && (MODE == 13 || MODE == 23)) ?
                      sample_n52
                    : ( (selector == -20 && (MODE == 14 || MODE == 22)) ?
                      sample_n36
                    : ( (selector == -20 && (MODE == 15 || MODE == 21)) ?
                      sample_n27
                    : ( (selector == -20 && (MODE == 16 || MODE == 20)) ?
                      sample_n22
                    : ( (selector == -20 && (MODE == 17 || MODE == 19)) ?
                      sample_n17
                    : ( (selector == -20 &&  MODE == 18)            ?
                      sample_n14
                    : ( (selector == -19 && (MODE == 13 || MODE == 23)) ?
                      sample_n49
                    : ( (selector == -19 && (MODE == 14 || MODE == 22)) ?
                      sample_n33
                    : ( (selector == -19 && (MODE == 15 || MODE == 21)) ?
```

```
                        sample_n25
2296                  : ( (selector == -19 && (MODE == 16 || MODE == 20)) ?
                        sample_n20
2297                  : ( (selector == -19 && (MODE == 17 || MODE == 19)) ?
                        sample_n16
2298                  : ( (selector == -19 &&  MODE == 18)            ?
                        sample_n13
2299                  : ( (selector == -18 && (MODE == 13 || MODE == 23)) ?
                        sample_n45
2300                  : ( (selector == -18 && (MODE == 14 || MODE == 22)) ?
                        sample_n31
2301                  : ( (selector == -18 && (MODE == 15 || MODE == 21)) ?
                        sample_n23
2302                  : ( (selector == -18 && (MODE == 16 || MODE == 20)) ?
                        sample_n19
2303                  : ( (selector == -18 && (MODE == 17 || MODE == 19)) ?
                        sample_n15
2304                  : ( (selector == -18 &&  MODE == 18)            ?
                        sample_n12
2305                  : ( (selector == -17 && (MODE == 13 || MODE == 23)) ?
                        sample_n42
2306                  : ( (selector == -17 && (MODE == 14 || MODE == 22)) ?
                        sample_n29
2307                  : ( (selector == -17 && (MODE == 15 || MODE == 21)) ?
                        sample_n22
2308                  : ( (selector == -17 && (MODE == 16 || MODE == 20)) ?
                        sample_n17
2309                  : ( (selector == -17 && (MODE == 17 || MODE == 19)) ?
                        sample_n14
2310                  : ( (selector == -17 &&  MODE == 18)            ?
                        sample_n11
2311                  : ( (selector == -16 && (MODE == 13 || MODE == 23)) ?
                        sample_n38
2312                  : ( (selector == -16 && (MODE == 14 || MODE == 22)) ?
                        sample_n26
2313                  : ( (selector == -16 && (MODE == 15 || MODE == 21)) ?
                        sample_n20
2314                  : ( (selector == -16 && (MODE == 16 || MODE == 20)) ?
                        sample_n16
2315                  : ( (selector == -16 && (MODE == 17 || MODE == 19)) ?
                        sample_n13
2316                  : ( (selector == -16 &&  MODE == 18)            ?
                        sample_n10
2317                  : ( (selector == -15 && (MODE == 12 || MODE == 24)) ?
                        sample_n63
2318                  : ( (selector == -15 && (MODE == 13 || MODE == 23)) ?
                        sample_n35
2319                  : ( (selector == -15 && (MODE == 14 || MODE == 22)) ?
                        sample_n24
2320                  : ( (selector == -15 && (MODE == 15 || MODE == 21)) ?
                        sample_n18
2321                  : ( (selector == -15 && (MODE == 16 || MODE == 20)) ?
                        sample_n14
2322                  : ( (selector == -15 && (MODE == 17 || MODE == 19)) ?
                        sample_n11
2323                  : ( (selector == -15 &&  MODE == 18)            ? sample_n9
2324                  : ( (selector == -14 && (MODE == 12 || MODE == 24)) ?
```

```
                    sample_n57
2325              : ( (selector == -14 && (MODE == 13 || MODE == 23)) ?
                    sample_n31
2326              : ( (selector == -14 && (MODE == 14 || MODE == 22)) ?
                    sample_n21
2327              : ( (selector == -14 && (MODE == 15 || MODE == 21)) ?
                    sample_n16
2328              : ( (selector == -14 && (MODE == 16 || MODE == 20)) ?
                    sample_n13
2329              : ( (selector == -14 && (MODE == 17 || MODE == 19)) ?
                    sample_n10
2330              : ( (selector == -14 &&  MODE == 18)            ? sample_n8
2331              : ( (selector == -13 && (MODE == 12 || MODE == 24)) ?
                    sample_n50
2332              : ( (selector == -13 && (MODE == 13 || MODE == 23)) ?
                    sample_n27
2333              : ( (selector == -13 && (MODE == 14 || MODE == 22)) ?
                    sample_n19
2334              : ( (selector == -13 && (MODE == 15 || MODE == 21)) ?
                    sample_n14
2335              : ( (selector == -13 && (MODE == 16 || MODE == 20)) ?
                    sample_n11
2336              : ( (selector == -13 && (MODE == 17 || MODE == 19)) ?
                    sample_n9
2337              : ( (selector == -13 &&  MODE == 18)            ? sample_n7
2338              : ( (selector == -12 && (MODE == 12 || MODE == 24)) ?
                    sample_n44
2339              : ( (selector == -12 && (MODE == 13 || MODE == 23)) ?
                    sample_n24
2340              : ( (selector == -12 && (MODE == 14 || MODE == 22)) ?
                    sample_n16
2341              : ( (selector == -12 && (MODE == 15 || MODE == 21)) ?
                    sample_n12
2342              : ( (selector == -12 && (MODE == 16 || MODE == 20)) ?
                    sample_n10
2343              : ( (selector == -12 && (MODE == 17 || MODE == 19)) ?
                    sample_n8
2344              : ( (selector == -12 &&  MODE == 18)            ? sample_n6
2345              : ( (selector == -11 && (MODE == 12 || MODE == 24)) ?
                    sample_n37
2346              : ( (selector == -11 && (MODE == 13 || MODE == 23)) ?
                    sample_n20
2347              : ( (selector == -11 && (MODE == 14 || MODE == 22)) ?
                    sample_n14
2348              : ( (selector == -11 && (MODE == 15 || MODE == 21)) ?
                    sample_n10
2349              : ( (selector == -11 && (MODE == 16 || MODE == 20)) ?
                    sample_n8
2350              : ( (selector == -11 && (MODE == 17 || MODE == 19)) ?
                    sample_n6
2351              : ( (selector == -11 &&  MODE == 18)            ? sample_n5
2352              : ( (selector == -10 && (MODE == 12 || MODE == 24)) ?
                    sample_n31
2353              : ( (selector == -10 && (MODE == 13 || MODE == 23)) ?
                    sample_n17
2354              : ( (selector == -10 && (MODE == 14 || MODE == 22)) ?
                    sample_n11
```

```
2355                        : ( (selector == -10 && (MODE == 15 || MODE == 21)) ?
                              sample_n8
2356                        : ( (selector == -10 && (MODE == 16 || MODE == 20)) ?
                              sample_n7
2357                        : ( (selector == -10 && (MODE == 17 || MODE == 19)) ?
                              sample_n5
2358                        : ( (selector == -10 &&  MODE == 18)            ? sample_n4
2359                        : ( (selector ==  -9 && (MODE == 11 || MODE == 25)) ?
                              sample_n63
2360                        : ( (selector ==  -9 && (MODE == 12 || MODE == 24)) ?
                              sample_n25
2361                        : ( (selector ==  -9 && (MODE == 13 || MODE == 23)) ?
                              sample_n13
2362                        : ( (selector ==  -9 && (MODE == 14 || MODE == 22)) ?
                              sample_n9
2363                        : ( (selector ==  -9 && (MODE == 15 || MODE == 21)) ?
                              sample_n7
2364                        : ( (selector ==  -9 && (MODE == 16 || MODE == 20)) ?
                              sample_n5
2365                        : ( (selector ==  -9 && (MODE == 17 || MODE == 19)) ?
                              sample_n4
2366                        : ( (selector ==  -9 &&  MODE == 18)            ? sample_n3
2367                        : ( (selector ==  -8 && (MODE == 11 || MODE == 25)) ?
                              sample_n47
2368                        : ( (selector ==  -8 && (MODE == 12 || MODE == 24)) ?
                              sample_n18
2369                        : ( (selector ==  -8 && (MODE == 13 || MODE == 23)) ?
                              sample_n10
2370                        : ( (selector ==  -8 && (MODE == 14 || MODE == 22)) ?
                              sample_n6
2371                        : ( (selector ==  -8 && (MODE == 15 || MODE == 21)) ?
                              sample_n5
2372                        : ( (selector ==  -8 && (MODE == 16 || MODE == 20)) ?
                              sample_n4
2373                        : ( (selector ==  -8 && (MODE == 17 || MODE == 19)) ?
                              sample_n3
2374                        : ( (selector ==  -8 &&  MODE == 18)            ? sample_n2
2375                        : ( (selector ==  -7 && (MODE == 11 || MODE == 25)) ?
                              sample_n31
2376                        : ( (selector ==  -7 && (MODE == 12 || MODE == 24)) ?
                              sample_n12
2377                        : ( (selector ==  -7 && (MODE == 13 || MODE == 23)) ?
                              sample_n6
2378                        : ( (selector ==  -7 && (MODE == 14 || MODE == 22)) ?
                              sample_n4
2379                        : ( (selector ==  -7 && (MODE == 15 || MODE == 21)) ?
                              sample_n3
2380                        : ( (selector ==  -7 && (MODE == 16 || MODE == 20)) ?
                              sample_n2
2381                        : ( (selector ==  -7 && (MODE == 17 || MODE == 19)) ?
                              sample_n1
2382                        : ( (selector ==  -7 &&  MODE == 18)            ? sample_n1
2383                        : ( (selector ==  -6 && (MODE == 11 || MODE == 25)) ?
                              sample_n15
2384                        : ( (selector ==  -6 && (MODE == 12 || MODE == 24)) ?
                              sample_n5
2385                        : ( (selector ==  -6 && (MODE == 13 || MODE == 23)) ?
```

```
                        sample_n3
2386                  : ( (selector ==  -6 && (MODE == 14 || MODE == 22)) ?
                        sample_n1
2387                  : ( (selector ==  -6 && (MODE == 15 || MODE == 21)) ?
                        sample_n1
2388                  : ( (selector ==  -6 && (MODE == 16 || MODE == 20)) ?
                        sample_n1
2389                  : ( (selector ==  -6 && (MODE == 17 || MODE == 19)) ?
                        sample_n0
2390                  : ( (selector ==  -6 &&  MODE == 18)            ? sample_n0
2391                  : (selector ==  -5 ? sample_NN
2392                  : (selector ==  -4 ? sample_0n
2393                  : (selector ==  -3 ? sample_1n
2394                  : (selector ==  -2 ? sample_2n
2395                  : (selector ==  -1 ? sample_3n
2396                  : (selector ==   0 ? sample_4n
2397                  : (selector ==   1 ? sample_5n
2398                  : (selector ==   2 ? sample_6n
2399                  : (selector ==   3 ? sample_7n
2400                  : (selector ==   4 ? sample_8n
2401                  : (selector ==   5 ? sample_9n
2402                  : (selector ==   6 ? sample_10n
2403                  : (selector ==   7 ? sample_11n
2404                  : (selector ==   8 ? sample_12n
2405                  : (selector ==   9 ? sample_13n
2406                  : (selector ==  10 ? sample_14n
2407                  : (selector ==  11 ? sample_15n
2408                  : (selector ==  12 ? sample_16n
2409                  : (selector ==  13 ? sample_17n
2410                  : (selector ==  14 ? sample_18n
2411                  : (selector ==  15 ? sample_19n
2412                  : (selector ==  16 ? sample_20n
2413                  : (selector ==  17 ? sample_21n
2414                  : (selector ==  18 ? sample_22n
2415                  : (selector ==  19 ? sample_23n
2416                  : (selector ==  20 ? sample_24n
2417                  : (selector ==  21 ? sample_25n
2418                  : (selector ==  22 ? sample_26n
2419                  : (selector ==  23 ? sample_27n
2420                  : (selector ==  24 ? sample_28n
2421                  : (selector ==  25 ? sample_29n
2422                  : (selector ==  26 ? sample_30n
2423                  : (selector ==  27 ? sample_31n
2424                  : (selector ==  28 ? sample_32n
2425                  : (selector ==  29 ? sample_33n
2426                  : (selector ==  30 ? sample_34n
2427                  : (selector ==  31 ? sample_35n
2428                  : (selector ==  32 ? sample_36n
2429                  : (selector ==  33 ? sample_37n
2430                  : (selector ==  34 ? sample_38n
2431                  : (selector ==  35 ? sample_39n
2432                  : (selector ==  36 ? sample_40n
2433                  : (selector ==  37 ? sample_41n
2434                  : (selector ==  38 ? sample_42n
2435                  : (selector ==  39 ? sample_43n
2436                  : (selector ==  40 ? sample_44n
2437                  : (selector ==  41 ? sample_45n
```

```
2438                        : (selector ==  42 ? sample_46n
2439                        : (selector ==  43 ? sample_47n
2440                        : (selector ==  44 ? sample_48n
2441                        : (selector ==  45 ? sample_49n
2442                        : (selector ==  46 ? sample_50n
2443                        : (selector ==  47 ? sample_51n
2444                        : (selector ==  48 ? sample_52n
2445                        : (selector ==  49 ? sample_53n
2446                        : (selector ==  50 ? sample_54n
2447                        : (selector ==  51 ? sample_55n
2448                        : (selector ==  52 ? sample_56n
2449                        : (selector ==  53 ? sample_57n
2450                        : (selector ==  54 ? sample_58n
2451                        : (selector ==  55 ? sample_59n
2452                        : (selector ==  56 ? sample_60n
2453                        : (selector ==  57 ? sample_61n
2454                        : (selector ==  58 ? sample_62n
2455                        : (selector ==  59 ? sample_63n
2456                        : 0 ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                               ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                               ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                               ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
                               ))))))))))));
2457
2458 endmodule
```

Listing B.12 – Código fonte do *shift buffer* do preditor angular

```
1
2  module angular_shift_buffer
3
4     #(parameter WIDTH = 8)
5
6     (
7        input clock,
8              reset,
9              shift_enable,
10
11       input[WIDTH-1:0]  pn0,
12                         pn1,
13                         pn2,
14                         pn3,
15
16       output[WIDTH-1:0] p00,
17                         p01,
18                         p02,
19                         p03,
20                         p10,
21                         p11,
22                         p12,
23                         p13,
24                         p20,
25                         p21,
26                         p22,
27                         p23,
28                         p30,
29                         p31,
```

```
30                      p32,
31                      p33
32      );
33
34      reg[WIDTH-1:0] reg_p00,
35                  reg_p01,
36                  reg_p02,
37                  reg_p03,
38                  reg_p10,
39                  reg_p11,
40                  reg_p12,
41                  reg_p13,
42                  reg_p20,
43                  reg_p21,
44                  reg_p22,
45                  reg_p23,
46                  reg_p30,
47                  reg_p31,
48                  reg_p32,
49                  reg_p33;
50
51      always @(posedge clock)
52      begin
53
54          if (reset)
55          begin
56              reg_p00 <= 8'b00000000;
57              reg_p01 <= 8'b00000000;
58              reg_p02 <= 8'b00000000;
59              reg_p03 <= 8'b00000000;
60              reg_p10 <= 8'b00000000;
61              reg_p11 <= 8'b00000000;
62              reg_p12 <= 8'b00000000;
63              reg_p13 <= 8'b00000000;
64              reg_p20 <= 8'b00000000;
65              reg_p21 <= 8'b00000000;
66              reg_p22 <= 8'b00000000;
67              reg_p23 <= 8'b00000000;
68              reg_p30 <= 8'b00000000;
69              reg_p31 <= 8'b00000000;
70              reg_p32 <= 8'b00000000;
71              reg_p33 <= 8'b00000000;
72          end
73          else
74          begin
75              if (shift_enable)
76              begin
77                  reg_p30 <= reg_p20;
78                  reg_p31 <= reg_p21;
79                  reg_p32 <= reg_p22;
80                  reg_p33 <= reg_p23;
81                  reg_p20 <= reg_p10;
82                  reg_p21 <= reg_p11;
83                  reg_p22 <= reg_p12;
84                  reg_p23 <= reg_p13;
85                  reg_p10 <= reg_p00;
86                  reg_p11 <= reg_p01;
```

```
87              reg_p12 <= reg_p02;
88              reg_p13 <= reg_p03;
89              reg_p00 <= pn0;
90              reg_p01 <= pn1;
91              reg_p02 <= pn2;
92              reg_p03 <= pn3;
93           end
94        end
95     end
96
97     assign p00 = reg_p30;
98     assign p01 = reg_p31;
99     assign p02 = reg_p32;
100    assign p03 = reg_p33;
101    assign p10 = reg_p20;
102    assign p11 = reg_p21;
103    assign p12 = reg_p22;
104    assign p13 = reg_p23;
105    assign p20 = reg_p10;
106    assign p21 = reg_p11;
107    assign p22 = reg_p12;
108    assign p23 = reg_p13;
109    assign p30 = reg_p00;
110    assign p31 = reg_p01;
111    assign p32 = reg_p02;
112    assign p33 = reg_p03;
113
114 endmodule
```

Listing B.13 – Código fonte da interface do preditor angular

```
1
2  module angular_top_level
3
4     #(parameter WIDTH = 8, parameter MODE = 2)
5
6     (
7        input clock,
8              reset,
9              start,
10
11       input [WIDTH-1:0] ref1,
12                         ref2,
13                         ref3,
14                         ref4,
15                         ref5,
16
17       input[WIDTH-3:0]  N,
18
19       output signed [WIDTH-1:0]  samples,
20
21       output   part_done,
22             done,
23
24       output [WIDTH-1:0]   p00,
25                         p01,
26                         p02,
```

```
27                      p03,
28                      p10,
29                      p11,
30                      p12,
31                      p13,
32                      p20,
33                      p21,
34                      p22,
35                      p23,
36                      p30,
37                      p31,
38                      p32,
39                      p33
40      );
41
42      wire  m_condition,
43            i_condition,
44            reset_registers,
45            m_reset,
46            def_enable,
47            refs_enable,
48            shift_enable,
49            i_index_enable,
50            m_index_enable,
51            indexes_enable,
52            displacement_enable;
53
54      angular_control angular_control_block(
55         .clock(clock),
56         .reset(reset),
57         .start(start),
58         .m_condition(m_condition),
59         .i_condition(i_condition),
60
61         .reset_registers(reset_registers),
62         .m_reset(m_reset),
63         .def_enable(def_enable),
64         .refs_enable(refs_enable),
65         .shift_enable(shift_enable),
66         .i_index_enable(i_index_enable),
67         .m_index_enable(m_index_enable),
68         .indexes_enable(indexes_enable),
69         .displacement_enable(displacement_enable),
70         .part_done(part_done),
71         .done(done)
72      );
73
74      angular_operative #(WIDTH, MODE) angular_operative_block(
75         .clock(clock),
76         .reset(reset_registers),
77         .m_reset(m_reset),
78         .def_enable(def_enable),
79         .refs_enable(refs_enable),
80         .shift_enable(shift_enable),
81         .i_index_enable(i_index_enable),
82         .m_index_enable(m_index_enable),
83         .indexes_enable(indexes_enable),
```

```
84          .displacement_enable(displacement_enable),
85          .ref1(ref1),
86          .ref2(ref2),
87          .ref3(ref3),
88          .ref4(ref4),
89          .ref5(ref5),
90          .N(N),
91
92          .m_condition(m_condition),
93          .i_condition(i_condition),
94          .samples(samples),
95          .p00(p00),
96          .p01(p01),
97          .p02(p02),
98          .p03(p03),
99          .p10(p10),
100         .p11(p11),
101         .p12(p12),
102         .p13(p13),
103         .p20(p20),
104         .p21(p21),
105         .p22(p22),
106         .p23(p23),
107         .p30(p30),
108         .p31(p31),
109         .p32(p32),
110         .p33(p33)
111     );
112
113 endmodule
```

Listing B.14 – Código fonte do bloco de controle do preditor DC

```
1
2  module dc_control
3
4      (
5          input     clock,
6                    reset,
7                    start,
8                    l_condition,
9                    k_condition,
10
11         output    dc_calc,
12                   dc_enable,
13                   def_enable,
14                   i_index_reset,
15                   i_index_enable,
16                   indexes_enable,
17                   samples_enable,
18                   mask_condition,
19                   reset_registers,
20                   update_results,
21                   part_done,
22                   done
23     );
24
```

```
25     reg [3:0] state;
26
27     reg reg_dc_calc,
28        reg_dc_enable,
29        reg_def_enable,
30        reg_i_index_reset,
31        reg_i_index_enable,
32        reg_indexes_enable,
33        reg_samples_enable,
34        reg_mask_condition,
35        reg_reset_registers,
36        reg_update_results,
37        reg_part_done,
38        reg_done;
39
40     parameter   INIT     = 0,
41              LOAD     = 1,
42              LLOOP    = 2,
43              LTEST    = 3,
44              CALC     = 4,
45              KLOOP    = 5,
46              KTEST    = 6,
47              BUBBLE1     = 7,
48              UPDATE1     = 8,
49              BUBBLE2     = 9,
50              UPDATE2     = 10,
51              DONE     = 11;
52
53     always @(state)
54     begin
55        case (state)
56
57           INIT:
58              begin
59                 reg_dc_calc       = 1'b0;
60                 reg_dc_enable     = 1'b0;
61                 reg_def_enable    = 1'b0;
62                 reg_i_index_reset = 1'b0;
63                 reg_i_index_enable   = 1'b0;
64                 reg_indexes_enable   = 1'b0;
65                 reg_samples_enable   = 1'b0;
66                 reg_mask_condition   = 1'b0;
67                 reg_reset_registers  = 1'b1;
68                 reg_update_results   = 1'b0;
69
70                 reg_part_done     = 1'b0;
71                 reg_done       = 1'b0;
72              end
73
74           LOAD:
75              begin
76                 reg_dc_calc       = 1'b0;
77                 reg_dc_enable     = 1'b0;
78                 reg_def_enable    = 1'b1;
79                 reg_i_index_reset = 1'b0;
80                 reg_i_index_enable   = 1'b0;
81                 reg_indexes_enable   = 1'b0;
```

```
82                reg_samples_enable    = 1'b0;
83                reg_mask_condition    = 1'b0;
84                reg_reset_registers   = 1'b0;
85                reg_update_results    = 1'b0;
86
87                reg_part_done     = 1'b0;
88                reg_done       = 1'b0;
89             end
90
91          LLOOP:
92             begin
93                reg_dc_calc       = 1'b0;
94                reg_dc_enable     = 1'b0;
95                reg_def_enable    = 1'b0;
96                reg_i_index_reset = 1'b0;
97                reg_i_index_enable    = 1'b1;
98                reg_indexes_enable    = 1'b0;
99                reg_samples_enable    = 1'b1;
100               reg_mask_condition    = 1'b1;
101               reg_reset_registers   = 1'b0;
102               reg_update_results    = 1'b0;
103
104               reg_part_done     = 1'b0;
105               reg_done       = 1'b0;
106            end
107
108         LTEST:
109            begin
110               reg_dc_calc       = 1'b0;
111               reg_dc_enable     = 1'b1;
112               reg_def_enable    = 1'b0;
113               reg_i_index_reset = 1'b0;
114               reg_i_index_enable    = 1'b0;
115               reg_indexes_enable    = 1'b0;
116               reg_samples_enable    = 1'b0;
117               reg_mask_condition    = 1'b1;
118               reg_reset_registers   = 1'b0;
119               reg_update_results    = 1'b0;
120
121               reg_part_done     = 1'b0;
122               reg_done       = 1'b0;
123            end
124
125         CALC:
126            begin
127               reg_dc_calc       = 1'b1;
128               reg_dc_enable     = 1'b0;
129               reg_def_enable    = 1'b0;
130               reg_i_index_reset = 1'b1;
131               reg_i_index_enable    = 1'b0;
132               reg_indexes_enable    = 1'b0;
133               reg_samples_enable    = 1'b0;
134               reg_mask_condition    = 1'b0;
135               reg_reset_registers   = 1'b0;
136               reg_update_results    = 1'b0;
137
138               reg_part_done     = 1'b0;
```

```
139                reg_done       = 1'b0;
140             end
141
142          KLOOP:
143             begin
144                reg_dc_calc       = 1'b0;
145                reg_dc_enable     = 1'b0;
146                reg_def_enable    = 1'b0;
147                reg_i_index_reset = 1'b0;
148                reg_i_index_enable   = 1'b1;
149                reg_indexes_enable   = 1'b1;
150                reg_samples_enable   = 1'b0;
151                reg_mask_condition   = 1'b0;
152                reg_reset_registers  = 1'b0;
153                reg_update_results   = 1'b0;
154
155                reg_part_done     = 1'b0;
156                reg_done       = 1'b0;
157             end
158
159          KTEST:
160             begin
161                reg_dc_calc       = 1'b0;
162                reg_dc_enable     = 1'b0;
163                reg_def_enable    = 1'b0;
164                reg_i_index_reset = 1'b0;
165                reg_i_index_enable   = 1'b0;
166                reg_indexes_enable   = 1'b0;
167                reg_samples_enable   = 1'b1;
168                reg_mask_condition   = 1'b0;
169                reg_reset_registers  = 1'b0;
170                reg_update_results   = 1'b0;
171
172                reg_part_done     = 1'b0;
173                reg_done       = 1'b0;
174             end
175
176          BUBBLE1:
177             begin
178                reg_dc_calc       = 1'b0;
179                reg_dc_enable     = 1'b0;
180                reg_def_enable    = 1'b0;
181                reg_i_index_reset = 1'b0;
182                reg_i_index_enable   = 1'b0;
183                reg_indexes_enable   = 1'b0;
184                reg_samples_enable   = 1'b0;
185                reg_mask_condition   = 1'b0;
186                reg_reset_registers  = 1'b0;
187                reg_update_results   = 1'b0;
188
189                reg_part_done     = 1'b0;
190                reg_done       = 1'b0;
191             end
192
193          UPDATE1:
194             begin
195                reg_dc_calc       = 1'b0;
```

```
196                 reg_dc_enable     = 1'b0;
197                 reg_def_enable    = 1'b0;
198                 reg_i_index_reset = 1'b0;
199                 reg_i_index_enable   = 1'b0;
200                 reg_indexes_enable   = 1'b0;
201                 reg_samples_enable   = 1'b0;
202                 reg_mask_condition   = 1'b0;
203                 reg_reset_registers  = 1'b0;
204                 reg_update_results   = 1'b1;
205
206                 reg_part_done     = 1'b0;
207                 reg_done       = 1'b0;
208              end
209
210           BUBBLE2:
211              begin
212                 reg_dc_calc       = 1'b0;
213                 reg_dc_enable     = 1'b0;
214                 reg_def_enable    = 1'b0;
215                 reg_i_index_reset = 1'b0;
216                 reg_i_index_enable   = 1'b0;
217                 reg_indexes_enable   = 1'b0;
218                 reg_samples_enable   = 1'b0;
219                 reg_mask_condition   = 1'b0;
220                 reg_reset_registers  = 1'b0;
221                 reg_update_results   = 1'b0;
222
223                 reg_part_done     = 1'b1;
224                 reg_done       = 1'b0;
225              end
226
227           UPDATE2:
228              begin
229                 reg_dc_calc       = 1'b0;
230                 reg_dc_enable     = 1'b0;
231                 reg_def_enable    = 1'b0;
232                 reg_i_index_reset = 1'b0;
233                 reg_i_index_enable   = 1'b0;
234                 reg_indexes_enable   = 1'b0;
235                 reg_samples_enable   = 1'b0;
236                 reg_mask_condition   = 1'b0;
237                 reg_reset_registers  = 1'b0;
238                 reg_update_results   = 1'b1;
239
240                 reg_part_done     = 1'b0;
241                 reg_done       = 1'b0;
242              end
243
244           DONE:
245              begin
246                 reg_dc_calc       = 1'b0;
247                 reg_dc_enable     = 1'b0;
248                 reg_def_enable    = 1'b0;
249                 reg_i_index_reset = 1'b0;
250                 reg_i_index_enable   = 1'b0;
251                 reg_indexes_enable   = 1'b0;
252                 reg_samples_enable   = 1'b0;
```

```
253                reg_mask_condition   = 1'b0;
254                reg_reset_registers  = 1'b0;
255                reg_update_results   = 1'b0;
256
257                reg_part_done     = 1'b1;
258                reg_done       = 1'b1;
259             end
260
261          default:
262             begin
263                reg_dc_calc       = 1'b0;
264                reg_dc_enable     = 1'b0;
265                reg_def_enable    = 1'b0;
266                reg_i_index_reset = 1'b0;
267                reg_i_index_enable   = 1'b0;
268                reg_indexes_enable   = 1'b0;
269                reg_samples_enable   = 1'b0;
270                reg_mask_condition   = 1'b0;
271                reg_reset_registers  = 1'b0;
272                reg_update_results   = 1'b0;
273
274                reg_part_done     = 1'b0;
275                reg_done       = 1'b0;
276             end
277       endcase
278    end
279
280    always @(posedge clock)
281    begin
282       if (reset)
283          state <= INIT;
284       else
285       begin
286          case (state)
287
288             INIT:
289                begin
290                   if (start)
291                      state <= LOAD;
292                   else
293                      state <= INIT;
294                end
295
296             LOAD:
297                begin
298                   state <= LLOOP;
299                end
300
301             LLOOP:
302                begin
303                   state <= LTEST;
304                end
305
306             LTEST:
307                begin
308                   if (l_condition)
309                      state <= LLOOP;
```

```
310                     else
311                         state <= CALC;
312                 end
313
314             CALC:
315                 begin
316                     state <= KLOOP;
317                 end
318
319             KLOOP:
320                 begin
321                     state <= KTEST;
322                 end
323
324             KTEST:
325                 begin
326                     if (k_condition)
327                         state <= BUBBLE1;
328                     else
329                         state <= UPDATE2;
330                 end
331
332             BUBBLE1:
333                 begin
334                     state <= UPDATE1;
335                 end
336
337             UPDATE1:
338                 begin
339                     state <= BUBBLE2;
340                 end
341
342             BUBBLE2:
343                 begin
344                     state <= KLOOP;
345                 end
346
347             UPDATE2:
348                 begin
349                     state <= DONE;
350                 end
351
352             DONE:
353                 begin
354                     state <= DONE;
355                 end
356
357             default:
358                 begin
359                     state <= INIT;
360                 end
361         endcase
362     end
363 end
364
365 assign dc_calc       = reg_dc_calc;
366 assign dc_enable     = reg_dc_enable;
```

```
367     assign def_enable    = reg_def_enable;
368     assign i_index_reset = reg_i_index_reset;
369     assign i_index_enable   = reg_i_index_enable;
370     assign indexes_enable   = reg_indexes_enable;
371     assign reset_registers  = reg_reset_registers;
372     assign samples_enable   = reg_samples_enable;
373     assign mask_condition   = reg_mask_condition;
374     assign update_results   = reg_update_results;
375     assign part_done     = reg_part_done;
376     assign done          = reg_done;
377
378 endmodule
```

Listing B.15 – Código fonte do bloco operativo do preditor DC

```
1
2  module dc_operative
3
4     #(parameter WIDTH = 8)
5
6     (
7        input clock,
8              reset,
9              dc_calc,
10             dc_enable,
11             def_enable,
12             i_index_reset,
13             i_index_enable,
14             indexes_enable,
15             samples_enable,
16             mask_condition,
17             update_results,
18
19       input[WIDTH-1:0]  sample_0n,
20                         sample_1n,
21                         sample_2n,
22                         sample_3n,
23                         sample_n0,
24                         sample_n1,
25                         sample_n2,
26                         sample_n3,
27
28       input[WIDTH-3:0]  N,
29
30       output   l_condition,
31              k_condition,
32
33       output[WIDTH-6:0] samples_x,
34                         samples_y,
35
36       output[WIDTH-1:0] p00,
37                         p01,
38                         p02,
39                         p03,
40                         p10,
41                         p11,
42                         p12,
```

```
43                      p13,
44                      p20,
45                      p21,
46                      p22,
47                      p23,
48                      p30,
49                      p31,
50                      p32,
51                      p33
52      );
53
54      reg[WIDTH+5:0] reg_dc_val;
55
56      reg[WIDTH-1:0] reg_sample_0n,
57                     reg_sample_1n,
58                     reg_sample_2n,
59                     reg_sample_3n,
60                     reg_sample_n0,
61                     reg_sample_n1,
62                     reg_sample_n2,
63                     reg_sample_n3,
64                     reg_p00,
65                     reg_p01,
66                     reg_p02,
67                     reg_p03,
68                     reg_p10,
69                     reg_p20,
70                     reg_p30;
71
72      reg[WIDTH-2:0] reg_i,
73                     reg_iterations;
74
75      reg[WIDTH-3:0] reg_N;
76
77      reg[WIDTH-6:0] reg_logN,
78                     reg_x_mask,
79                     reg_y_mask;
80
81      reg[WIDTH-7:0] reg_x,
82                     reg_y;
83
84      wire[WIDTH+5:0]   dc_parc_val;
85
86      wire[WIDTH-1:0] dc_val,
87                     temp_p01,
88                     temp_p02,
89                     temp_p03,
90                     temp_p10,
91                     temp_p20,
92                     temp_p30;
93
94      wire[WIDTH-2:0] x_shift,
95                     i;
96
97      wire[WIDTH-7:0]   x,
98                     y;
99
```

```
100     always @(posedge clock)
101     begin
102
103        if (reset)
104        begin
105           reg_iterations <= 7'b0000000;
106           reg_dc_val     <= 14'b00000000000000;
107           reg_logN    <= 3'b000;
108           reg_N       <= 6'b000000;
109           reg_i       <= 7'b0000000;
110           reg_x_mask     <= 3'b000;
111           reg_y_mask     <= 3'b000;
112           reg_sample_0n  <= 8'b00000000;
113           reg_sample_1n  <= 8'b00000000;
114           reg_sample_2n  <= 8'b00000000;
115           reg_sample_3n  <= 8'b00000000;
116           reg_sample_n0  <= 8'b00000000;
117           reg_sample_n1  <= 8'b00000000;
118           reg_sample_n2  <= 8'b00000000;
119           reg_sample_n3  <= 8'b00000000;
120           reg_x       <= 2'b00;
121           reg_y       <= 2'b00;
122           reg_p00        <= 8'b00000000;
123           reg_p01        <= 8'b00000000;
124           reg_p02        <= 8'b00000000;
125           reg_p03        <= 8'b00000000;
126           reg_p10        <= 8'b00000000;
127           reg_p20        <= 8'b00000000;
128           reg_p30        <= 8'b00000000;
129        end
130        else
131        begin
132
133           if (def_enable)
134           begin
135              reg_N <= N;
136              reg_dc_val <= N;
137
138              case (N)
139                 4: begin
140                       reg_logN <= 3'b011;
141                       reg_x_mask <= 3'b000;
142                       reg_y_mask <= 3'b000;
143                       reg_iterations <= 7'b0000001;
144                    end
145
146                 8: begin
147                       reg_logN <= 3'b100;
148                       reg_x_mask <= 3'b001;
149                       reg_y_mask <= 3'b001;
150                       reg_iterations <= 7'b0000100;
151                    end
152
153                 16:    begin
154                       reg_logN <= 3'b101;
155                       reg_x_mask <= 3'b011;
156                       reg_y_mask <= 3'b011;
```

```
157                     reg_iterations <= 7'b0010000;
158                 end
159
160             32:   begin
161                     reg_logN <= 3'b110;
162                     reg_x_mask <= 3'b111;
163                     reg_y_mask <= 3'b011;
164                     reg_iterations <= 7'b1000000;
165                 end
166
167             default:
168                 begin
169                     reg_logN <= 3'b011;
170                     reg_x_mask <= 3'b000;
171                     reg_y_mask <= 3'b000;
172                     reg_iterations <= 7'b0000001;
173                 end
174         endcase
175     end
176
177     if (samples_enable)
178     begin
179         reg_sample_0n <= sample_0n;
180         reg_sample_1n <= sample_1n;
181         reg_sample_2n <= sample_2n;
182         reg_sample_3n <= sample_3n;
183         reg_sample_n0 <= sample_n0;
184         reg_sample_n1 <= sample_n1;
185         reg_sample_n2 <= sample_n2;
186         reg_sample_n3 <= sample_n3;
187     end
188
189     if (i_index_reset)
190     begin
191         reg_i <= 0;
192     end
193     else
194     begin
195         if (i_index_enable)
196         begin
197             reg_i <= i;
198         end
199     end
200
201     if (dc_enable)
202     begin
203         reg_dc_val <= dc_parc_val;
204     end
205
206     if (dc_calc)
207     begin
208         reg_dc_val <= dc_val;
209     end
210
211     if (indexes_enable)
212     begin
213         reg_x <= x;
```

```
214             reg_y <= y;
215          end
216
217          if (reg_x + reg_y == 2)
218          begin
219             reg_p00 <= (reg_sample_n0 + reg_sample_0n + 2 + 2 * reg_dc_val) >>
                    2;
220          end
221          else
222          begin
223             if (reg_x == 1)
224             begin
225                reg_p00 <= (reg_sample_0n + 2 + 3 * reg_dc_val) >> 2;
226             end
227             else
228             begin
229                if (reg_y == 1)
230                begin
231                   reg_p00 <= (reg_sample_n0 + 2 + 3 * reg_dc_val) >> 2;
232                end
233                else
234                begin
235                   reg_p00 <= reg_dc_val;
236                end
237             end
238          end
239
240          if (update_results)
241          begin
242             reg_p01 <= temp_p01;
243             reg_p02 <= temp_p02;
244             reg_p03 <= temp_p03;
245             reg_p10 <= temp_p10;
246             reg_p20 <= temp_p20;
247             reg_p30 <= temp_p30;
248          end
249       end
250    end
251
252    assign i       = reg_i + 1;
253
254    assign dc_parc_val   = reg_dc_val
255                   + reg_sample_0n
256                   + reg_sample_1n
257                   + reg_sample_2n
258                   + reg_sample_3n
259                   + reg_sample_n0
260                   + reg_sample_n1
261                   + reg_sample_n2
262                   + reg_sample_n3;
263
264    assign dc_val     = reg_dc_val >> reg_logN;
265
266    assign l_condition   = (reg_i < (reg_N >> 2)) ? 1 : 0;
267    assign k_condition   = (reg_i < reg_iterations) ? 1 : 0;
268
269    assign x       = (reg_i <= reg_x_mask) ? 1 : 0;
```

```
270     assign y       = ((reg_i & reg_x_mask) == 0) ? 1 : 0;
271
272     assign temp_p01      = (reg_y == 1) ? (reg_sample_n1 + 2 + 3 * reg_dc_val) >>
            2 : reg_dc_val;
273     assign temp_p02      = (reg_y == 1) ? (reg_sample_n2 + 2 + 3 * reg_dc_val) >>
            2 : reg_dc_val;
274     assign temp_p03      = (reg_y == 1) ? (reg_sample_n3 + 2 + 3 * reg_dc_val) >>
            2 : reg_dc_val;
275
276     assign temp_p10      = (reg_x == 1) ? (reg_sample_1n + 2 + 3 * reg_dc_val) >>
            2 : reg_dc_val;
277     assign temp_p20      = (reg_x == 1) ? (reg_sample_2n + 2 + 3 * reg_dc_val) >>
            2 : reg_dc_val;
278     assign temp_p30      = (reg_x == 1) ? (reg_sample_3n + 2 + 3 * reg_dc_val) >>
            2 : reg_dc_val;
279
280     assign p00        = reg_p00;
281
282     assign p01        = reg_p01;
283     assign p02        = reg_p02;
284     assign p03        = reg_p03;
285
286     assign p10        = reg_p10;
287     assign p20        = reg_p20;
288     assign p30        = reg_p30;
289
290     assign p11        = reg_dc_val;
291     assign p12        = reg_dc_val;
292     assign p13        = reg_dc_val;
293     assign p21        = reg_dc_val;
294     assign p22        = reg_dc_val;
295     assign p23        = reg_dc_val;
296     assign p31        = reg_dc_val;
297     assign p32        = reg_dc_val;
298     assign p33        = reg_dc_val;
299
300     assign x_shift = reg_i >> reg_y_mask;
301
302     assign samples_x = (mask_condition) ? reg_i & reg_x_mask : reg_i & reg_x_mask
            ;
303     assign samples_y = (mask_condition) ? reg_i & reg_x_mask : x_shift;
304
305 endmodule
```

Listing B.16 – Código fonte da interface do preditor DC

```
1
2  module dc_top_level
3
4     #(parameter WIDTH = 8)
5
6     (
7        input clock,
8              reset,
9              start,
10
11       input [WIDTH-1:0] sample_0n,
```

```
12                      sample_1n,
13                      sample_2n,
14                      sample_3n,
15                      sample_n0,
16                      sample_n1,
17                      sample_n2,
18                      sample_n3,
19
20      input[WIDTH-3:0]  N,
21
22      output[WIDTH-6:0]    samples_x,
23                      samples_y,
24
25      output   part_done,
26             done,
27
28      output [WIDTH-1:0]   p00,
29                      p01,
30                      p02,
31                      p03,
32                      p10,
33                      p11,
34                      p12,
35                      p13,
36                      p20,
37                      p21,
38                      p22,
39                      p23,
40                      p30,
41                      p31,
42                      p32,
43                      p33
44    );
45
46    wire  l_condition,
47          k_condition,
48          dc_calc,
49          dc_enable,
50          def_enable,
51          i_index_reset,
52          i_index_enable,
53          indexes_enable,
54          samples_enable,
55          mask_condition,
56          reset_registers,
57          update_results;
58
59    dc_control dc_control_block(
60       .clock(clock),
61       .reset(reset),
62       .start(start),
63       .l_condition(l_condition),
64       .k_condition(k_condition),
65
66       .dc_calc(dc_calc),
67       .dc_enable(dc_enable),
68       .def_enable(def_enable),
```

```
69          .i_index_reset(i_index_reset),
70          .i_index_enable(i_index_enable),
71          .indexes_enable(indexes_enable),
72          .samples_enable(samples_enable),
73          .mask_condition(mask_condition),
74          .reset_registers(reset_registers),
75          .update_results(update_results),
76          .part_done(part_done),
77          .done(done)
78      );
79
80      dc_operative #(WIDTH) dc_operative_block(
81          .clock(clock),
82          .reset(reset_registers),
83          .dc_calc(dc_calc),
84          .dc_enable(dc_enable),
85          .def_enable(def_enable),
86          .i_index_reset(i_index_reset),
87          .i_index_enable(i_index_enable),
88          .indexes_enable(indexes_enable),
89          .samples_enable(samples_enable),
90          .mask_condition(mask_condition),
91          .update_results(update_results),
92          .sample_0n(sample_0n),
93          .sample_1n(sample_1n),
94          .sample_2n(sample_2n),
95          .sample_3n(sample_3n),
96          .sample_n0(sample_n0),
97          .sample_n1(sample_n1),
98          .sample_n2(sample_n2),
99          .sample_n3(sample_n3),
100         .N(N),
101
102         .l_condition(l_condition),
103         .k_condition(k_condition),
104         .samples_x(samples_x),
105         .samples_y(samples_y),
106         .p00(p00),
107         .p01(p01),
108         .p02(p02),
109         .p03(p03),
110         .p10(p10),
111         .p11(p11),
112         .p12(p12),
113         .p13(p13),
114         .p20(p20),
115         .p21(p21),
116         .p22(p22),
117         .p23(p23),
118         .p30(p30),
119         .p31(p31),
120         .p32(p32),
121         .p33(p33)
122     );
123
124 endmodule
```

Listing B.17 – Código fonte do bloco de controle do preditor planar

```
1
2  module planar_control
3
4     (
5        input     clock,
6              reset,
7              start,
8              t_condition,
9              m_condition,
10
11       output    k_reset,
12             p_enable,
13             def_enable,
14             diffs_enable,
15             shift_enable,
16             indexes_enable,
17             samples_enable,
18             k_index_enable,
19             m_index_enable,
20             t_index_enable,
21             reset_registers,
22             update_results,
23             part_done,
24             done
25    );
26
27    reg [3:0] state;
28
29    reg reg_k_reset,
30       reg_p_enable,
31       reg_def_enable,
32       reg_diffs_enable,
33       reg_shift_enable,
34       reg_indexes_enable,
35       reg_samples_enable,
36       reg_k_index_enable,
37       reg_m_index_enable,
38       reg_t_index_enable,
39       reg_reset_registers,
40       reg_update_results,
41       reg_part_done,
42       reg_done;
43
44    parameter    INIT = 0,
45             LOAD = 1,
46             PRE_CALC = 2,
47             KLOOP = 3,
48             LLOOP = 4,
49             LTEST = 5,
50             KTEST = 6,
51             PREDICTION = 7,
52             UPDATE = 8,
53             NTEST = 9,
54             DONE = 10;
55
56    always @(state)
```

```
57      begin
58         case (state)
59
60            INIT:
61               begin
62                  reg_k_reset       = 1'b0;
63                  reg_p_enable      = 1'b0;
64                  reg_def_enable    = 1'b0;
65                  reg_diffs_enable  = 1'b0;
66                  reg_shift_enable  = 1'b0;
67                  reg_indexes_enable   = 1'b0;
68                  reg_samples_enable   = 1'b0;
69                  reg_k_index_enable   = 1'b0;
70                  reg_m_index_enable   = 1'b0;
71                  reg_t_index_enable   = 1'b0;
72                  reg_reset_registers  = 1'b1;
73                  reg_update_results   = 1'b0;
74
75                  reg_part_done     = 1'b0;
76                  reg_done       = 1'b0;
77               end
78
79            LOAD:
80               begin
81                  reg_k_reset       = 1'b0;
82                  reg_p_enable      = 1'b0;
83                  reg_def_enable    = 1'b1;
84                  reg_diffs_enable  = 1'b0;
85                  reg_shift_enable  = 1'b0;
86                  reg_indexes_enable   = 1'b0;
87                  reg_samples_enable   = 1'b0;
88                  reg_k_index_enable   = 1'b0;
89                  reg_m_index_enable   = 1'b0;
90                  reg_t_index_enable   = 1'b0;
91                  reg_reset_registers  = 1'b0;
92                  reg_update_results   = 1'b0;
93
94                  reg_part_done     = 1'b0;
95                  reg_done       = 1'b0;
96               end
97
98            PRE_CALC:
99               begin
100                 reg_k_reset       = 1'b0;
101                 reg_p_enable      = 1'b0;
102                 reg_def_enable    = 1'b0;
103                 reg_diffs_enable  = 1'b0;
104                 reg_shift_enable  = 1'b0;
105                 reg_indexes_enable   = 1'b1;
106                 reg_samples_enable   = 1'b1;
107                 reg_k_index_enable   = 1'b0;
108                 reg_m_index_enable   = 1'b0;
109                 reg_t_index_enable   = 1'b0;
110                 reg_reset_registers  = 1'b0;
111                 reg_update_results   = 1'b0;
112
113                 reg_part_done     = 1'b0;
```

```
114                reg_done       = 1'b0;
115             end
116
117          KLOOP:
118             begin
119                reg_k_reset       = 1'b0;
120                reg_p_enable      = 1'b0;
121                reg_def_enable    = 1'b0;
122                reg_diffs_enable  = 1'b1;
123                reg_shift_enable  = 1'b0;
124                reg_indexes_enable   = 1'b0;
125                reg_samples_enable   = 1'b0;
126                reg_k_index_enable   = 1'b0;
127                reg_m_index_enable   = 1'b0;
128                reg_t_index_enable   = 1'b0;
129                reg_reset_registers  = 1'b0;
130                reg_update_results   = 1'b0;
131
132                reg_part_done     = 1'b0;
133                reg_done       = 1'b0;
134             end
135
136          LLOOP:
137             begin
138                reg_k_reset       = 1'b0;
139                reg_p_enable      = 1'b1;
140                reg_def_enable    = 1'b0;
141                reg_diffs_enable  = 1'b0;
142                reg_shift_enable  = 1'b0;
143                reg_indexes_enable   = 1'b0;
144                reg_samples_enable   = 1'b0;
145                reg_k_index_enable   = 1'b0;
146                reg_m_index_enable   = 1'b0;
147                reg_t_index_enable   = 1'b0;
148                reg_reset_registers  = 1'b0;
149                reg_update_results   = 1'b0;
150
151                reg_part_done     = 1'b0;
152                reg_done       = 1'b0;
153             end
154
155          LTEST:
156             begin
157                reg_k_reset       = 1'b0;
158                reg_p_enable      = 1'b0;
159                reg_def_enable    = 1'b0;
160                reg_diffs_enable  = 1'b0;
161                reg_shift_enable  = 1'b1;
162                reg_indexes_enable   = 1'b0;
163                reg_samples_enable   = 1'b0;
164                reg_k_index_enable   = 1'b0;
165                reg_m_index_enable   = 1'b0;
166                reg_t_index_enable   = 1'b1;
167                reg_reset_registers  = 1'b0;
168                reg_update_results   = 1'b0;
169
170                reg_part_done     = 1'b0;
```

```
171                 reg_done        = 1'b0;
172             end
173
174         KTEST:
175             begin
176                 reg_k_reset         = 1'b0;
177                 reg_p_enable        = 1'b0;
178                 reg_def_enable      = 1'b0;
179                 reg_diffs_enable    = 1'b0;
180                 reg_shift_enable    = 1'b0;
181                 reg_indexes_enable    = 1'b0;
182                 reg_samples_enable    = 1'b0;
183                 reg_k_index_enable    = 1'b1;
184                 reg_m_index_enable    = 1'b0;
185                 reg_t_index_enable    = 1'b0;
186                 reg_reset_registers   = 1'b0;
187                 reg_update_results    = 1'b0;
188
189                 reg_part_done      = 1'b0;
190                 reg_done        = 1'b0;
191             end
192
193         PREDICTION:
194             begin
195                 reg_k_reset         = 1'b0;
196                 reg_p_enable        = 1'b0;
197                 reg_def_enable      = 1'b0;
198                 reg_diffs_enable    = 1'b0;
199                 reg_shift_enable    = 1'b0;
200                 reg_indexes_enable    = 1'b0;
201                 reg_samples_enable    = 1'b0;
202                 reg_k_index_enable    = 1'b0;
203                 reg_m_index_enable    = 1'b1;
204                 reg_t_index_enable    = 1'b0;
205                 reg_reset_registers   = 1'b0;
206                 reg_update_results    = 1'b0;
207
208                 reg_part_done      = 1'b0;
209                 reg_done        = 1'b0;
210             end
211
212         UPDATE:
213             begin
214                 reg_k_reset         = 1'b0;
215                 reg_p_enable        = 1'b0;
216                 reg_def_enable      = 1'b0;
217                 reg_diffs_enable    = 1'b0;
218                 reg_shift_enable    = 1'b0;
219                 reg_indexes_enable    = 1'b0;
220                 reg_samples_enable    = 1'b0;
221                 reg_k_index_enable    = 1'b0;
222                 reg_m_index_enable    = 1'b0;
223                 reg_t_index_enable    = 1'b0;
224                 reg_reset_registers   = 1'b0;
225                 reg_update_results    = 1'b1;
226
227                 reg_part_done      = 1'b0;
```

```
228                 reg_done        = 1'b0;
229             end
230
231         NTEST:
232             begin
233                 reg_k_reset         = 1'b1;
234                 reg_p_enable        = 1'b0;
235                 reg_def_enable      = 1'b0;
236                 reg_diffs_enable    = 1'b0;
237                 reg_shift_enable    = 1'b0;
238                 reg_indexes_enable    = 1'b0;
239                 reg_samples_enable    = 1'b0;
240                 reg_k_index_enable    = 1'b0;
241                 reg_m_index_enable    = 1'b0;
242                 reg_t_index_enable    = 1'b0;
243                 reg_reset_registers   = 1'b0;
244                 reg_update_results    = 1'b0;
245
246                 reg_part_done      = 1'b1;
247                 reg_done        = 1'b0;
248             end
249
250         DONE:
251             begin
252                 reg_k_reset         = 1'b0;
253                 reg_p_enable        = 1'b0;
254                 reg_def_enable      = 1'b0;
255                 reg_diffs_enable    = 1'b0;
256                 reg_shift_enable    = 1'b0;
257                 reg_indexes_enable    = 1'b0;
258                 reg_samples_enable    = 1'b0;
259                 reg_k_index_enable    = 1'b0;
260                 reg_m_index_enable    = 1'b0;
261                 reg_t_index_enable    = 1'b0;
262                 reg_reset_registers   = 1'b0;
263                 reg_update_results    = 1'b0;
264
265                 reg_part_done      = 1'b1;
266                 reg_done        = 1'b1;
267             end
268
269         default:
270             begin
271                 reg_k_reset         = 1'b0;
272                 reg_p_enable        = 1'b0;
273                 reg_def_enable      = 1'b0;
274                 reg_diffs_enable    = 1'b0;
275                 reg_shift_enable    = 1'b0;
276                 reg_indexes_enable    = 1'b0;
277                 reg_samples_enable    = 1'b0;
278                 reg_k_index_enable    = 1'b0;
279                 reg_m_index_enable    = 1'b0;
280                 reg_t_index_enable    = 1'b0;
281                 reg_reset_registers   = 1'b0;
282                 reg_update_results    = 1'b0;
283
284                 reg_part_done      = 1'b0;
```

```
285                reg_done        = 1'b0;
286             end
287          endcase
288       end
289
290       always @(posedge clock)
291       begin
292          if (reset)
293             state <= INIT;
294          else
295          begin
296             case (state)
297
298                INIT:
299                   begin
300                      $display("####␣Planar␣####␣INIT");
301                      $fflush();
302                      if (start)
303                         state <= LOAD;
304                      else
305                         state <= INIT;
306                   end
307
308                LOAD:
309                   begin
310                      $display("####␣Planar␣####␣LOAD");
311                      $fflush();
312                      state <= PRE_CALC;
313                   end
314
315                PRE_CALC:
316                   begin
317                      $display("####␣Planar␣####␣PRE_CALC");
318                      $fflush();
319                      state <= KLOOP;
320                   end
321
322                KLOOP:
323                   begin
324                      $display("####␣Planar␣####␣KLOOP");
325                      $fflush();
326                      state <= LLOOP;
327                   end
328
329                LLOOP:
330                   begin
331                      $display("####␣Planar␣####␣LLOOP");
332                      $fflush();
333                      state <= LTEST;
334                   end
335
336                LTEST:
337                   begin
338                      $display("####␣Planar␣####␣LTEST");
339                      $fflush();
340                      state <= KTEST;
341                   end
```

```
342
343             KTEST:
344                begin
345                   $display("####␣Planar␣####␣KTEST");
346                   $fflush();
347                   if (t_condition)
348                      state <= KLOOP;
349                   else
350                      state <= PREDICTION;
351                end
352
353             PREDICTION:
354                begin
355                   $display("####␣Planar␣####␣PREDICTION");
356                   $fflush();
357                   state <= UPDATE;
358                end
359
360             UPDATE:
361                begin
362                   $display("####␣Planar␣####␣UPDATE");
363                   $fflush();
364                   state <= NTEST;
365                end
366
367             NTEST:
368                begin
369                   $display("####␣Planar␣####␣NTEST");
370                   $fflush();
371                   if (m_condition)
372                      state <= PRE_CALC;
373                   else
374                      state <= DONE;
375                end
376
377             DONE:
378                begin
379                   $display("####␣Planar␣####␣DONE");
380                   $fflush();
381                   state <= DONE;
382                end
383
384             default:
385                begin
386                   $display("####␣Planar␣####␣default");
387                   $fflush();
388                   state <= INIT;
389                end
390          endcase
391       end
392    end
393
394    assign k_reset       = reg_k_reset;
395    assign p_enable         = reg_p_enable;
396    assign def_enable    = reg_def_enable;
397    assign diffs_enable     = reg_diffs_enable;
398    assign shift_enable     = reg_shift_enable;
```

```
399     assign indexes_enable   = reg_indexes_enable;
400     assign k_index_enable   = reg_k_index_enable;
401     assign m_index_enable   = reg_m_index_enable;
402     assign t_index_enable   = reg_t_index_enable;
403     assign reset_registers  = reg_reset_registers;
404     assign samples_enable   = reg_samples_enable;
405     assign update_results   = reg_update_results;
406     assign part_done     = reg_part_done;
407     assign done          = reg_done;
408
409 endmodule
```

Listing B.18 – Código fonte do bloco operativo do preditor planar

```
1
2  module planar_operative
3
4     #(parameter WIDTH = 8)
5
6     (
7        input clock,
8              k_reset,
9              p_enable,
10             def_enable,
11             diffs_enable,
12             shift_enable,
13             indexes_enable,
14             samples_enable,
15             k_index_enable,
16             m_index_enable,
17             t_index_enable,
18             reset_registers,
19             update_results,
20
21       input[WIDTH-1:0]  sample_0n,
22                         sample_1n,
23                         sample_2n,
24                         sample_3n,
25                         sample_Nn,
26                         sample_n0,
27                         sample_n1,
28                         sample_n2,
29                         sample_n3,
30                         sample_nN,
31
32       input[WIDTH-3:0]  N,
33
34       output   t_condition,
35              m_condition,
36
37       output[WIDTH-6:0] samples_x,
38                         samples_y,
39
40       output[WIDTH-1:0] p00,
41                         p01,
42                         p02,
43                         p03,
```

```
44                      p10,
45                      p11,
46                      p12,
47                      p13,
48                      p20,
49                      p21,
50                      p22,
51                      p23,
52                      p30,
53                      p31,
54                      p32,
55                      p33
56      );
57
58      reg[WIDTH-6:0] reg_t,
59                     reg_k,
60                     reg_logN,
61                     reg_x_mask,
62                     reg_y_mask,
63                     reg_index_mask;
64
65      reg[WIDTH-3:0] reg_coef_h1,
66                     reg_coef_v1,
67                     reg_N;
68
69      reg[WIDTH-2:0] reg_iterations,
70                     reg_m,
71                     reg_x,
72                     reg_y;
73
74      reg[WIDTH-1:0] reg_sample_0n,
75                     reg_sample_1n,
76                     reg_sample_2n,
77                     reg_sample_3n,
78                     reg_sample_Nn,
79                     reg_sample_n0,
80                     reg_sample_n1,
81                     reg_sample_n2,
82                     reg_sample_n3,
83                     reg_sample_nN;
84
85      reg[WIDTH+5:0] reg_ph0,
86                     reg_ph1,
87                     reg_ph2,
88                     reg_ph3,
89                     reg_pv0,
90                     reg_pv1,
91                     reg_pv2,
92                     reg_pv3;
93
94      reg[WIDTH+5:0] reg_coef_h2,
95                     reg_coef_v2;
96
97      wire[WIDTH-6:0] t;
98
99      wire[WIDTH-4:0]   def;
100
```

```
101     wire[WIDTH-2:0] m,
102                     x,
103                     y;
104
105     wire[WIDTH-3:0]   coef_h1,
106                     coef_v1;
107
108     wire[WIDTH+5:0]   ph0,
109                     ph1,
110                     ph2,
111                     ph3,
112                     pv0,
113                     pv1,
114                     pv2,
115                     pv3;
116
117     wire[WIDTH+5:0]   coef_h2,
118                     coef_v2;
119
120     planar_shift_buffer #(WIDTH) ph_pv(
121        .clock(clock),
122        .reset_registers(reset_registers),
123        .shift_enable(shift_enable),
124        .update_results(update_results),
125        .logN(reg_logN),
126        .N(reg_N),
127        .ph0(reg_ph0),
128        .ph1(reg_ph1),
129        .ph2(reg_ph2),
130        .ph3(reg_ph3),
131        .pv0(reg_pv0),
132        .pv1(reg_pv1),
133        .pv2(reg_pv2),
134        .pv3(reg_pv3),
135        .p00(p00),
136        .p01(p01),
137        .p02(p02),
138        .p03(p03),
139        .p10(p10),
140        .p11(p11),
141        .p12(p12),
142        .p13(p13),
143        .p20(p20),
144        .p21(p21),
145        .p22(p22),
146        .p23(p23),
147        .p30(p30),
148        .p31(p31),
149        .p32(p32),
150        .p33(p33)
151     );
152
153     always @(posedge clock)
154     begin
155
156        if (reset_registers)
157        begin
```

```
158            reg_sample_0n <= 8'b00000000;
159            reg_sample_1n <= 8'b00000000;
160            reg_sample_2n <= 8'b00000000;
161            reg_sample_3n <= 8'b00000000;
162            reg_sample_Nn <= 8'b00000000;
163            reg_sample_n0 <= 8'b00000000;
164            reg_sample_n1 <= 8'b00000000;
165            reg_sample_n2 <= 8'b00000000;
166            reg_sample_n3 <= 8'b00000000;
167            reg_sample_nN <= 8'b00000000;
168            reg_ph0 <= 14'b00000000000000;
169            reg_ph1 <= 14'b00000000000000;
170            reg_ph2 <= 14'b00000000000000;
171            reg_ph3 <= 14'b00000000000000;
172            reg_pv0 <= 14'b00000000000000;
173            reg_pv1 <= 14'b00000000000000;
174            reg_pv2 <= 14'b00000000000000;
175            reg_pv3 <= 14'b00000000000000;
176            reg_m <= 7'b0000000;
177            reg_t <= 3'b000;
178            reg_k <= 3'b000;
179            reg_N <= 6'b000000;
180            reg_x <= 7'b0000000;
181            reg_y <= 7'b0000000;
182            reg_logN <= 3'b000;
183            reg_coef_h1 <= 6'b000000;
184            reg_coef_h2 <= 14'b00000000000000;
185            reg_coef_v1 <= 6'b000000;
186            reg_coef_v2 <= 14'b00000000000000;
187            reg_iterations <= 7'b0000000;
188            reg_index_mask <= 3'b000;
189            reg_x_mask <= 3'b000;
190            reg_y_mask <= 3'b000;
191         end
192         else
193         begin
194
195            if (def_enable)
196            begin
197               reg_N <= N;
198               reg_sample_Nn <= sample_Nn;
199               reg_sample_nN <= sample_nN;
200
201               case (N)
202                  4: begin
203                        reg_logN <= 3'b011;
204                        reg_iterations <= 7'b0000001;
205                        reg_index_mask <= 3'b000;
206                        reg_x_mask <= 3'b000;
207                        reg_y_mask <= 3'b000;
208                     end
209
210                  8: begin
211                        reg_logN <= 3'b100;
212                        reg_iterations <= 7'b0000100;
213                        reg_index_mask <= 3'b001;
214                        reg_x_mask <= 3'b001;
```

```
215                     reg_y_mask <= 3'b001;
216                 end
217
218             16:   begin
219                     reg_logN <= 3'b101;
220                     reg_iterations <= 7'b0010000;
221                     reg_index_mask <= 3'b011;
222                     reg_x_mask <= 3'b011;
223                     reg_y_mask <= 3'b010;
224                 end
225
226             32:   begin
227                     reg_logN <= 3'b110;
228                     reg_iterations <= 7'b1000000;
229                     reg_index_mask <= 3'b111;
230                     reg_x_mask <= 3'b111;
231                     reg_y_mask <= 3'b011;
232                 end
233
234             default: begin
235                     reg_logN <= 3'b011;
236                     reg_iterations <= 7'b0000001;
237                     reg_index_mask <= 3'b000;
238                     reg_x_mask <= 3'b000;
239                     reg_y_mask <= 3'b000;
240                 end
241         endcase
242     end
243
244     if (indexes_enable)
245     begin
246         reg_x <= x;
247         reg_y <= y;
248     end
249
250     if (diffs_enable)
251     begin
252         reg_coef_h1 <= coef_h1;
253         reg_coef_h2 <= coef_h2;
254         reg_coef_v1 <= coef_v1;
255         reg_coef_v2 <= coef_v2;
256     end
257
258     if (p_enable)
259     begin
260         reg_ph0 <= ph0;
261         reg_ph1 <= ph1;
262         reg_ph2 <= ph2;
263         reg_ph3 <= ph3;
264         reg_pv0 <= pv0;
265         reg_pv1 <= pv1;
266         reg_pv2 <= pv2;
267         reg_pv3 <= pv3;
268     end
269
270     if (t_index_enable)
271     begin
```

```
272             reg_t <= t;
273          end
274
275          if (k_reset)
276          begin
277             reg_k <= 0;
278          end
279          else
280          begin
281             if (k_index_enable)
282             begin
283                reg_k <= reg_t;
284             end
285          end
286
287          if (m_index_enable)
288          begin
289             reg_m <= m;
290          end
291
292          if (samples_enable)
293          begin
294             reg_sample_0n <= sample_0n;
295             reg_sample_1n <= sample_1n;
296             reg_sample_2n <= sample_2n;
297             reg_sample_3n <= sample_3n;
298             reg_sample_n0 <= sample_n0;
299             reg_sample_n1 <= sample_n1;
300             reg_sample_n2 <= sample_n2;
301             reg_sample_n3 <= sample_n3;
302          end
303
304       end
305    end
306
307    assign coef_h1    = reg_N - 1 - (reg_x + reg_k);
308    assign coef_h2    = (reg_x + reg_k + 1) * reg_sample_Nn;
309
310    assign coef_v1    = reg_N - 1 - (reg_y + reg_k);
311    assign coef_v2    = (reg_y + reg_k + 1) * reg_sample_nN;
312
313    assign ph0        = reg_coef_h1 * reg_sample_n0 + reg_coef_h2;
314    assign ph1        = reg_coef_h1 * reg_sample_n1 + reg_coef_h2;
315    assign ph2        = reg_coef_h1 * reg_sample_n2 + reg_coef_h2;
316    assign ph3        = reg_coef_h1 * reg_sample_n3 + reg_coef_h2;
317
318    assign pv0        = reg_coef_v1 * reg_sample_0n + reg_coef_v2;
319    assign pv1        = reg_coef_v1 * reg_sample_1n + reg_coef_v2;
320    assign pv2        = reg_coef_v1 * reg_sample_2n + reg_coef_v2;
321    assign pv3        = reg_coef_v1 * reg_sample_3n + reg_coef_v2;
322
323    assign t       = reg_k + 1;
324    assign m       = reg_m + 1;
325
326    assign x       = (reg_N == 4) ? 0 : (reg_m & reg_index_mask) << 2;
327    assign y       = (reg_N == 4) ? 0 : (reg_m / (reg_N >> 2)) << 2;
328
```

```
329     assign t_condition   = (reg_t < 4) ? 1 : 0;
330     assign m_condition   = (reg_m < reg_iterations) ? 1 : 0;
331
332     assign samples_x  = reg_m & reg_x_mask;
333     assign samples_y  = reg_m >> reg_y_mask;
334
335 endmodule
```

Listing B.19 – Código fonte do *shift buffer* do preditor planar

```
1
2  module planar_shift_buffer
3
4     #(parameter WIDTH = 8)
5
6     (
7        input clock,
8              reset_registers,
9              shift_enable,
10             update_results,
11
12       input[WIDTH-6:0]  logN,
13
14       input[WIDTH-3:0]  N,
15
16       input[WIDTH+5:0]  ph0,
17                         ph1,
18                         ph2,
19                         ph3,
20                         pv0,
21                         pv1,
22                         pv2,
23                         pv3,
24
25       output[WIDTH-1:0] p00,
26                         p01,
27                         p02,
28                         p03,
29                         p10,
30                         p11,
31                         p12,
32                         p13,
33                         p20,
34                         p21,
35                         p22,
36                         p23,
37                         p30,
38                         p31,
39                         p32,
40                         p33
41    );
42
43    reg[WIDTH+5:0] reg_ph00,
44                   reg_ph01,
45                   reg_ph02,
46                   reg_ph03,
47                   reg_ph10,
```

```
48                  reg_ph11,
49                  reg_ph12,
50                  reg_ph13,
51                  reg_ph20,
52                  reg_ph21,
53                  reg_ph22,
54                  reg_ph23,
55                  reg_ph30,
56                  reg_ph31,
57                  reg_ph32,
58                  reg_ph33,
59                  reg_pv00,
60                  reg_pv01,
61                  reg_pv02,
62                  reg_pv03,
63                  reg_pv10,
64                  reg_pv11,
65                  reg_pv12,
66                  reg_pv13,
67                  reg_pv20,
68                  reg_pv21,
69                  reg_pv22,
70                  reg_pv23,
71                  reg_pv30,
72                  reg_pv31,
73                  reg_pv32,
74                  reg_pv33;
75
76      reg[WIDTH-1:0] reg_p00,
77                  reg_p01,
78                  reg_p02,
79                  reg_p03,
80                  reg_p10,
81                  reg_p11,
82                  reg_p12,
83                  reg_p13,
84                  reg_p20,
85                  reg_p21,
86                  reg_p22,
87                  reg_p23,
88                  reg_p30,
89                  reg_p31,
90                  reg_p32,
91                  reg_p33;
92
93      wire[WIDTH+6:0]   temp_p00,
94                  temp_p01,
95                  temp_p02,
96                  temp_p03,
97                  temp_p10,
98                  temp_p11,
99                  temp_p12,
100                 temp_p13,
101                 temp_p20,
102                 temp_p21,
103                 temp_p22,
104                 temp_p23,
```

```
105                 temp_p30,
106                 temp_p31,
107                 temp_p32,
108                 temp_p33;
109
110     always @(posedge clock)
111     begin
112
113        if (reset_registers)
114        begin
115           reg_ph00 <= 14'b00000000000000;
116           reg_ph01 <= 14'b00000000000000;
117           reg_ph02 <= 14'b00000000000000;
118           reg_ph03 <= 14'b00000000000000;
119           reg_ph10 <= 14'b00000000000000;
120           reg_ph11 <= 14'b00000000000000;
121           reg_ph12 <= 14'b00000000000000;
122           reg_ph13 <= 14'b00000000000000;
123           reg_ph20 <= 14'b00000000000000;
124           reg_ph21 <= 14'b00000000000000;
125           reg_ph22 <= 14'b00000000000000;
126           reg_ph23 <= 14'b00000000000000;
127           reg_ph30 <= 14'b00000000000000;
128           reg_ph31 <= 14'b00000000000000;
129           reg_ph32 <= 14'b00000000000000;
130           reg_ph33 <= 14'b00000000000000;
131           reg_pv00 <= 14'b00000000000000;
132           reg_pv01 <= 14'b00000000000000;
133           reg_pv02 <= 14'b00000000000000;
134           reg_pv03 <= 14'b00000000000000;
135           reg_pv10 <= 14'b00000000000000;
136           reg_pv11 <= 14'b00000000000000;
137           reg_pv12 <= 14'b00000000000000;
138           reg_pv13 <= 14'b00000000000000;
139           reg_pv20 <= 14'b00000000000000;
140           reg_pv21 <= 14'b00000000000000;
141           reg_pv22 <= 14'b00000000000000;
142           reg_pv23 <= 14'b00000000000000;
143           reg_pv30 <= 14'b00000000000000;
144           reg_pv31 <= 14'b00000000000000;
145           reg_pv32 <= 14'b00000000000000;
146           reg_pv33 <= 14'b00000000000000;
147           reg_p00   <= 8'b00000000;
148           reg_p01   <= 8'b00000000;
149           reg_p02   <= 8'b00000000;
150           reg_p03   <= 8'b00000000;
151           reg_p10   <= 8'b00000000;
152           reg_p11   <= 8'b00000000;
153           reg_p12   <= 8'b00000000;
154           reg_p13   <= 8'b00000000;
155           reg_p20   <= 8'b00000000;
156           reg_p21   <= 8'b00000000;
157           reg_p22   <= 8'b00000000;
158           reg_p23   <= 8'b00000000;
159           reg_p30   <= 8'b00000000;
160           reg_p31   <= 8'b00000000;
161           reg_p32   <= 8'b00000000;
```

```
162             reg_p33   <= 8'b00000000;
163         end
164         else
165         begin
166             if (shift_enable)
167             begin
168                 reg_ph00 <= reg_ph10;
169                 reg_ph01 <= reg_ph11;
170                 reg_ph02 <= reg_ph12;
171                 reg_ph03 <= reg_ph13;
172
173                 reg_ph10 <= reg_ph20;
174                 reg_ph11 <= reg_ph21;
175                 reg_ph12 <= reg_ph22;
176                 reg_ph13 <= reg_ph23;
177
178                 reg_ph20 <= reg_ph30;
179                 reg_ph21 <= reg_ph31;
180                 reg_ph22 <= reg_ph32;
181                 reg_ph23 <= reg_ph33;
182
183                 reg_ph30 <= ph0;
184                 reg_ph31 <= ph1;
185                 reg_ph32 <= ph2;
186                 reg_ph33 <= ph3;
187
188                 reg_pv00 <= reg_pv01;
189                 reg_pv10 <= reg_pv11;
190                 reg_pv20 <= reg_pv21;
191                 reg_pv30 <= reg_pv31;
192
193                 reg_pv01 <= reg_pv02;
194                 reg_pv11 <= reg_pv12;
195                 reg_pv21 <= reg_pv22;
196                 reg_pv31 <= reg_pv32;
197
198                 reg_pv02 <= reg_pv03;
199                 reg_pv12 <= reg_pv13;
200                 reg_pv22 <= reg_pv23;
201                 reg_pv32 <= reg_pv33;
202
203                 reg_pv03 <= pv0;
204                 reg_pv13 <= pv1;
205                 reg_pv23 <= pv2;
206                 reg_pv33 <= pv3;
207             end
208
209             if (update_results)
210             begin
211                 reg_p00 <= temp_p00;
212                 reg_p01 <= temp_p01;
213                 reg_p02 <= temp_p02;
214                 reg_p03 <= temp_p03;
215                 reg_p10 <= temp_p10;
216                 reg_p11 <= temp_p11;
217                 reg_p12 <= temp_p12;
218                 reg_p13 <= temp_p13;
```

```
219             reg_p20 <= temp_p20;
220             reg_p21 <= temp_p21;
221             reg_p22 <= temp_p22;
222             reg_p23 <= temp_p23;
223             reg_p30 <= temp_p30;
224             reg_p31 <= temp_p31;
225             reg_p32 <= temp_p32;
226             reg_p33 <= temp_p33;
227          end
228       end
229    end
230
231    assign temp_p00 = (reg_ph00 + reg_pv00 + N) >> logN;
232    assign temp_p01 = (reg_ph01 + reg_pv01 + N) >> logN;
233    assign temp_p02 = (reg_ph02 + reg_pv02 + N) >> logN;
234    assign temp_p03 = (reg_ph03 + reg_pv03 + N) >> logN;
235    assign temp_p10 = (reg_ph10 + reg_pv10 + N) >> logN;
236    assign temp_p11 = (reg_ph11 + reg_pv11 + N) >> logN;
237    assign temp_p12 = (reg_ph12 + reg_pv12 + N) >> logN;
238    assign temp_p13 = (reg_ph13 + reg_pv13 + N) >> logN;
239    assign temp_p20 = (reg_ph20 + reg_pv20 + N) >> logN;
240    assign temp_p21 = (reg_ph21 + reg_pv21 + N) >> logN;
241    assign temp_p22 = (reg_ph22 + reg_pv22 + N) >> logN;
242    assign temp_p23 = (reg_ph23 + reg_pv23 + N) >> logN;
243    assign temp_p30 = (reg_ph30 + reg_pv30 + N) >> logN;
244    assign temp_p31 = (reg_ph31 + reg_pv31 + N) >> logN;
245    assign temp_p32 = (reg_ph32 + reg_pv32 + N) >> logN;
246    assign temp_p33 = (reg_ph33 + reg_pv33 + N) >> logN;
247
248    assign p00 = reg_p00;
249    assign p01 = reg_p01;
250    assign p02 = reg_p02;
251    assign p03 = reg_p03;
252    assign p10 = reg_p10;
253    assign p11 = reg_p11;
254    assign p12 = reg_p12;
255    assign p13 = reg_p13;
256    assign p20 = reg_p20;
257    assign p21 = reg_p21;
258    assign p22 = reg_p22;
259    assign p23 = reg_p23;
260    assign p30 = reg_p30;
261    assign p31 = reg_p31;
262    assign p32 = reg_p32;
263    assign p33 = reg_p33;
264
265 endmodule
```

Listing B.20 – Código fonte da interface do preditor planar

```
1
2  module planar_top_level
3
4     #(parameter WIDTH = 8)
5
6     (
7        input clock,
```

```
8               reset,
9               start,
10
11      input [WIDTH-1:0] sample_0n,
12                        sample_1n,
13                        sample_2n,
14                        sample_3n,
15                        sample_Nn,
16                        sample_n0,
17                        sample_n1,
18                        sample_n2,
19                        sample_n3,
20                        sample_nN,
21
22      input[WIDTH-3:0]  N,
23
24      output[WIDTH-6:0] samples_x,
25                        samples_y,
26
27      output    part_done,
28              done,
29
30      output [WIDTH-1:0]    p00,
31                        p01,
32                        p02,
33                        p03,
34                        p10,
35                        p11,
36                        p12,
37                        p13,
38                        p20,
39                        p21,
40                        p22,
41                        p23,
42                        p30,
43                        p31,
44                        p32,
45                        p33
46  );
47
48  wire  t_condition,
49        m_condition,
50        k_reset,
51        p_enable,
52        def_enable,
53        diffs_enable,
54        shift_enable,
55        indexes_enable,
56        samples_enable,
57        k_index_enable,
58        m_index_enable,
59        t_index_enable,
60        reset_registers,
61        update_results;
62
63  planar_control planar_control_block(
64     .clock(clock),
```

```
65          .reset(reset),
66          .start(start),
67          .t_condition(t_condition),
68          .m_condition(m_condition),
69
70          .k_reset(k_reset),
71          .p_enable(p_enable),
72          .def_enable(def_enable),
73          .diffs_enable(diffs_enable),
74          .shift_enable(shift_enable),
75          .indexes_enable(indexes_enable),
76          .samples_enable(samples_enable),
77          .k_index_enable(k_index_enable),
78          .m_index_enable(m_index_enable),
79          .t_index_enable(t_index_enable),
80          .reset_registers(reset_registers),
81          .update_results(update_results),
82          .part_done(part_done),
83          .done(done)
84      );
85
86      planar_operative #(WIDTH) planar_operative_block(
87          .clock(clock),
88          .k_reset(k_reset),
89          .p_enable(p_enable),
90          .def_enable(def_enable),
91          .diffs_enable(diffs_enable),
92          .shift_enable(shift_enable),
93          .indexes_enable(indexes_enable),
94          .samples_enable(samples_enable),
95          .k_index_enable(k_index_enable),
96          .m_index_enable(m_index_enable),
97          .t_index_enable(t_index_enable),
98          .reset_registers(reset_registers),
99          .update_results(update_results),
100         .sample_0n(sample_0n),
101         .sample_1n(sample_1n),
102         .sample_2n(sample_2n),
103         .sample_3n(sample_3n),
104         .sample_Nn(sample_Nn),
105         .sample_n0(sample_n0),
106         .sample_n1(sample_n1),
107         .sample_n2(sample_n2),
108         .sample_n3(sample_n3),
109         .sample_nN(sample_nN),
110         .N(N),
111
112         .t_condition(t_condition),
113         .m_condition(m_condition),
114         .samples_x(samples_x),
115         .samples_y(samples_y),
116         .p00(p00),
117         .p01(p01),
118         .p02(p02),
119         .p03(p03),
120         .p10(p10),
121         .p11(p11),
```

```
122         .p12(p12),
123         .p13(p13),
124         .p20(p20),
125         .p21(p21),
126         .p22(p22),
127         .p23(p23),
128         .p30(p30),
129         .p31(p31),
130         .p32(p32),
131         .p33(p33)
132     );
133 
134 endmodule
```

Listing B.21 – Código fonte do bloco de controle do gerenciador de amostras

```
1 
2  module main_control
3 
4      (
5         input     ack,
6               clock,
7               reset,
8               start,
9               i_condition,
10              planar_done,
11              dc_done,
12              angular_mode2_done,
13              angular_mode3_done,
14              angular_mode4_done,
15              angular_mode5_done,
16              angular_mode6_done,
17              angular_mode7_done,
18              angular_mode8_done,
19              angular_mode9_done,
20              angular_mode10_done,
21              angular_mode11_done,
22              angular_mode12_done,
23              angular_mode13_done,
24              angular_mode14_done,
25              angular_mode15_done,
26              angular_mode16_done,
27              angular_mode17_done,
28              angular_mode18_done,
29              angular_mode19_done,
30              angular_mode20_done,
31              angular_mode21_done,
32              angular_mode22_done,
33              angular_mode23_done,
34              angular_mode24_done,
35              angular_mode25_done,
36              angular_mode26_done,
37              angular_mode27_done,
38              angular_mode28_done,
39              angular_mode29_done,
40              angular_mode30_done,
41              angular_mode31_done,
```

```
42              angular_mode32_done,
43              angular_mode33_done,
44              angular_mode34_done,
45
46        output    samples_request,
47              def_enable,
48              i_index_enable,
49              samples_enable,
50              reset_registers,
51              start_predictors,
52              done
53    );
54
55    reg [2:0] state;
56
57    reg reg_def_enable,
58        reg_i_index_enable,
59        reg_samples_enable,
60        reg_samples_request,
61        reg_reset_registers,
62        reg_start_predictors,
63        reg_done;
64
65    parameter    INIT      = 0,
66              PRE_LOAD = 1,
67              LOAD      = 2,
68              PREDICTION   = 3,
69              DONE      = 4;
70
71    always @(state)
72    begin
73        case (state)
74
75           INIT:
76              begin
77                 reg_samples_request     = 1'b0;
78                 reg_def_enable       = 1'b0;
79                 reg_samples_enable      = 1'b0;
80                 reg_i_index_enable      = 1'b0;
81                 reg_reset_registers     = 1'b1;
82                 reg_start_predictors = 1'b0;
83                 reg_done          = 1'b0;
84              end
85
86           PRE_LOAD:
87              begin
88                 reg_samples_request     = 1'b0;
89                 reg_def_enable       = 1'b1;
90                 reg_samples_enable      = 1'b0;
91                 reg_i_index_enable      = 1'b1;
92                 reg_reset_registers     = 1'b0;
93                 reg_start_predictors = 1'b0;
94                 reg_done          = 1'b0;
95              end
96
97           LOAD:
98              begin
```

```
99                reg_samples_request     = 1'b1;
100               reg_def_enable       = 1'b0;
101               reg_samples_enable      = 1'b1;
102               reg_i_index_enable      = 1'b1;
103               reg_reset_registers     = 1'b0;
104               reg_start_predictors = 1'b0;
105               reg_done          = 1'b0;
106            end
107
108         PREDICTION:
109            begin
110               reg_samples_request     = 1'b0;
111               reg_def_enable       = 1'b0;
112               reg_samples_enable      = 1'b0;
113               reg_i_index_enable      = 1'b0;
114               reg_reset_registers     = 1'b0;
115               reg_start_predictors = 1'b1;
116               reg_done          = 1'b0;
117            end
118
119         DONE:
120            begin
121               reg_samples_request     = 1'b0;
122               reg_def_enable       = 1'b0;
123               reg_samples_enable      = 1'b0;
124               reg_i_index_enable      = 1'b0;
125               reg_reset_registers     = 1'b0;
126               reg_start_predictors = 1'b0;
127               reg_done          = 1'b1;
128            end
129
130         default:
131            begin
132               reg_samples_request     = 1'b0;
133               reg_def_enable       = 1'b0;
134               reg_samples_enable      = 1'b0;
135               reg_i_index_enable      = 1'b0;
136               reg_reset_registers     = 1'b0;
137               reg_start_predictors = 1'b0;
138               reg_done          = 1'b0;
139            end
140      endcase
141   end
142
143   always @(posedge clock)
144   begin
145      if (reset)
146         state <= INIT;
147      else
148      begin
149         case (state)
150
151            INIT:
152               begin
153                  $display("########␣Main␣########␣INIT");
154                  $fflush();
155                  if (start)
```

```
156                     state <= PRE_LOAD;
157                 else
158                     state <= INIT;
159             end
160
161         PRE_LOAD:
162             begin
163                 $display("########␣Main␣########␣PRE_LOAD");
164                 $fflush();
165                 state <= LOAD;
166             end
167
168         LOAD:
169             begin
170                 $display("########␣Main␣########␣LOAD");
171                 $fflush();
172                 if (i_condition)
173                     state <= LOAD;
174                 else
175                     state <= PREDICTION;
176             end
177
178         PREDICTION:
179             begin
180                 $display("########␣Main␣########␣PREDICTION");
181                 $fflush();
182                 if  (  planar_done
183                     & dc_done
184                     & angular_mode2_done
185                     & angular_mode3_done
186                     & angular_mode4_done
187                     & angular_mode5_done
188                     & angular_mode6_done
189                     & angular_mode7_done
190                     & angular_mode8_done
191                     & angular_mode9_done
192                     & angular_mode10_done
193                     & angular_mode11_done
194                     & angular_mode12_done
195                     & angular_mode13_done
196                     & angular_mode14_done
197                     & angular_mode15_done
198                     & angular_mode16_done
199                     & angular_mode17_done
200                     & angular_mode18_done
201                     & angular_mode19_done
202                     & angular_mode20_done
203                     & angular_mode21_done
204                     & angular_mode22_done
205                     & angular_mode23_done
206                     & angular_mode24_done
207                     & angular_mode25_done
208                     & angular_mode26_done
209                     & angular_mode27_done
210                     & angular_mode28_done
211                     & angular_mode29_done
212                     & angular_mode30_done
```

```
213                      & angular_mode31_done
214                      & angular_mode32_done
215                      & angular_mode33_done
216                      & angular_mode34_done
217                     )
218                     state <= DONE;
219                  else
220                     state <= PREDICTION;
221               end
222
223            DONE:
224               begin
225                  $display("########␣Main␣########␣DONE");
226                  $fflush();
227                  if (ack)
228                     state <= INIT;
229                  else
230                     state <= DONE;
231               end
232
233            default:
234               begin
235                  state <= INIT;
236               end
237         endcase
238      end
239   end
240
241   assign samples_request  = reg_samples_request;
242   assign def_enable    = reg_def_enable;
243   assign i_index_enable   = reg_i_index_enable;
244   assign samples_enable   = reg_samples_enable;
245   assign reset_registers  = reg_reset_registers;
246   assign start_predictors = reg_start_predictors;
247   assign done          = reg_done;
248
249 endmodule
```

Listing B.22 – Código fonte do bloco operativo do gerenciador de amostras

```
1
2  module main_operative
3
4     #(parameter WIDTH = 8)
5
6     (
7        input clock,
8              reset,
9              def_enable,
10             samples_enable,
11             i_index_enable,
12
13       input[WIDTH-6:0]  planar_samples_x,
14                         planar_samples_y,
15
16                         dc_samples_x,
17                         dc_samples_y,
```

```
18
19          input[WIDTH-2:0]  filter_samples_mode10,
20                            filter_samples_mode26,
21
22          input signed [WIDTH-1:0]   angular_mode2_samples,
23                                     angular_mode3_samples,
24                                     angular_mode4_samples,
25                                     angular_mode5_samples,
26                                     angular_mode6_samples,
27                                     angular_mode7_samples,
28                                     angular_mode8_samples,
29                                     angular_mode9_samples,
30                                     angular_mode10_samples,
31                                     angular_mode11_samples,
32                                     angular_mode12_samples,
33                                     angular_mode13_samples,
34                                     angular_mode14_samples,
35                                     angular_mode15_samples,
36                                     angular_mode16_samples,
37                                     angular_mode17_samples,
38                                     angular_mode18_samples,
39                                     angular_mode19_samples,
40                                     angular_mode20_samples,
41                                     angular_mode21_samples,
42                                     angular_mode22_samples,
43                                     angular_mode23_samples,
44                                     angular_mode24_samples,
45                                     angular_mode25_samples,
46                                     angular_mode26_samples,
47                                     angular_mode27_samples,
48                                     angular_mode28_samples,
49                                     angular_mode29_samples,
50                                     angular_mode30_samples,
51                                     angular_mode31_samples,
52                                     angular_mode32_samples,
53                                     angular_mode33_samples,
54                                     angular_mode34_samples,
55
56          input[WIDTH-1:0]  sample_0n,
57                            sample_1n,
58                            sample_2n,
59                            sample_3n,
60                            sample_n0,
61                            sample_n1,
62                            sample_n2,
63                            sample_n3,
64                            sample_Nn,
65                            sample_nN,
66                            sample_NN,
67
68          input[WIDTH-3:0]  N,
69
70          output    i_condition,
71
72          output[WIDTH-1:0] planar_sample_0n,
73                            planar_sample_1n,
74                            planar_sample_2n,
```

```
                    planar_sample_3n,
                    planar_sample_Nn,
                    planar_sample_n0,
                    planar_sample_n1,
                    planar_sample_n2,
                    planar_sample_n3,
                    planar_sample_nN,

                    dc_sample_0n,
                    dc_sample_1n,
                    dc_sample_2n,
                    dc_sample_3n,
                    dc_sample_n0,
                    dc_sample_n1,
                    dc_sample_n2,
                    dc_sample_n3,

                    angular_mode2_ref1,
                    angular_mode2_ref2,
                    angular_mode2_ref3,
                    angular_mode2_ref4,
                    angular_mode2_ref5,

                    angular_mode3_ref1,
                    angular_mode3_ref2,
                    angular_mode3_ref3,
                    angular_mode3_ref4,
                    angular_mode3_ref5,

                    angular_mode4_ref1,
                    angular_mode4_ref2,
                    angular_mode4_ref3,
                    angular_mode4_ref4,
                    angular_mode4_ref5,

                    angular_mode5_ref1,
                    angular_mode5_ref2,
                    angular_mode5_ref3,
                    angular_mode5_ref4,
                    angular_mode5_ref5,

                    angular_mode6_ref1,
                    angular_mode6_ref2,
                    angular_mode6_ref3,
                    angular_mode6_ref4,
                    angular_mode6_ref5,

                    angular_mode7_ref1,
                    angular_mode7_ref2,
                    angular_mode7_ref3,
                    angular_mode7_ref4,
                    angular_mode7_ref5,

                    angular_mode8_ref1,
                    angular_mode8_ref2,
                    angular_mode8_ref3,
                    angular_mode8_ref4,
```

```
                    angular_mode8_ref5,

                    angular_mode9_ref1,
                    angular_mode9_ref2,
                    angular_mode9_ref3,
                    angular_mode9_ref4,
                    angular_mode9_ref5,

                    angular_mode10_ref0,
                    angular_mode10_ref1,
                    angular_mode10_ref2,
                    angular_mode10_ref3,
                    angular_mode10_ref4,
                    angular_mode10_ref5,

                    angular_mode11_ref1,
                    angular_mode11_ref2,
                    angular_mode11_ref3,
                    angular_mode11_ref4,
                    angular_mode11_ref5,

                    angular_mode12_ref1,
                    angular_mode12_ref2,
                    angular_mode12_ref3,
                    angular_mode12_ref4,
                    angular_mode12_ref5,

                    angular_mode13_ref1,
                    angular_mode13_ref2,
                    angular_mode13_ref3,
                    angular_mode13_ref4,
                    angular_mode13_ref5,

                    angular_mode14_ref1,
                    angular_mode14_ref2,
                    angular_mode14_ref3,
                    angular_mode14_ref4,
                    angular_mode14_ref5,

                    angular_mode15_ref1,
                    angular_mode15_ref2,
                    angular_mode15_ref3,
                    angular_mode15_ref4,
                    angular_mode15_ref5,

                    angular_mode16_ref1,
                    angular_mode16_ref2,
                    angular_mode16_ref3,
                    angular_mode16_ref4,
                    angular_mode16_ref5,

                    angular_mode17_ref1,
                    angular_mode17_ref2,
                    angular_mode17_ref3,
                    angular_mode17_ref4,
                    angular_mode17_ref5,

```

```
                    angular_mode18_ref1,
                    angular_mode18_ref2,
                    angular_mode18_ref3,
                    angular_mode18_ref4,
                    angular_mode18_ref5,

                    angular_mode19_ref1,
                    angular_mode19_ref2,
                    angular_mode19_ref3,
                    angular_mode19_ref4,
                    angular_mode19_ref5,

                    angular_mode20_ref1,
                    angular_mode20_ref2,
                    angular_mode20_ref3,
                    angular_mode20_ref4,
                    angular_mode20_ref5,

                    angular_mode21_ref1,
                    angular_mode21_ref2,
                    angular_mode21_ref3,
                    angular_mode21_ref4,
                    angular_mode21_ref5,

                    angular_mode22_ref1,
                    angular_mode22_ref2,
                    angular_mode22_ref3,
                    angular_mode22_ref4,
                    angular_mode22_ref5,

                    angular_mode23_ref1,
                    angular_mode23_ref2,
                    angular_mode23_ref3,
                    angular_mode23_ref4,
                    angular_mode23_ref5,

                    angular_mode24_ref1,
                    angular_mode24_ref2,
                    angular_mode24_ref3,
                    angular_mode24_ref4,
                    angular_mode24_ref5,

                    angular_mode25_ref1,
                    angular_mode25_ref2,
                    angular_mode25_ref3,
                    angular_mode25_ref4,
                    angular_mode25_ref5,

                    angular_mode26_ref0,
                    angular_mode26_ref1,
                    angular_mode26_ref2,
                    angular_mode26_ref3,
                    angular_mode26_ref4,
                    angular_mode26_ref5,

                    angular_mode27_ref1,
                    angular_mode27_ref2,
```

```
                    angular_mode27_ref3,
                    angular_mode27_ref4,
                    angular_mode27_ref5,

                    angular_mode28_ref1,
                    angular_mode28_ref2,
                    angular_mode28_ref3,
                    angular_mode28_ref4,
                    angular_mode28_ref5,

                    angular_mode29_ref1,
                    angular_mode29_ref2,
                    angular_mode29_ref3,
                    angular_mode29_ref4,
                    angular_mode29_ref5,

                    angular_mode30_ref1,
                    angular_mode30_ref2,
                    angular_mode30_ref3,
                    angular_mode30_ref4,
                    angular_mode30_ref5,

                    angular_mode31_ref1,
                    angular_mode31_ref2,
                    angular_mode31_ref3,
                    angular_mode31_ref4,
                    angular_mode31_ref5,

                    angular_mode32_ref1,
                    angular_mode32_ref2,
                    angular_mode32_ref3,
                    angular_mode32_ref4,
                    angular_mode32_ref5,

                    angular_mode33_ref1,
                    angular_mode33_ref2,
                    angular_mode33_ref3,
                    angular_mode33_ref4,
                    angular_mode33_ref5,

                    angular_mode34_ref1,
                    angular_mode34_ref2,
                    angular_mode34_ref3,
                    angular_mode34_ref4,
                    angular_mode34_ref5
    );

    reg[WIDTH-4:0] reg_i,
                reg_iterations;

    reg[WIDTH-1:0] reg_sample_0n, reg_sample_32n, reg_sample_n0,  reg_sample_n32,
                reg_sample_1n, reg_sample_33n, reg_sample_n1,  reg_sample_n33,
                reg_sample_2n, reg_sample_34n, reg_sample_n2,  reg_sample_n34,
                reg_sample_3n, reg_sample_35n, reg_sample_n3,  reg_sample_n35,
                reg_sample_4n, reg_sample_36n, reg_sample_n4,  reg_sample_n36,
                reg_sample_5n, reg_sample_37n, reg_sample_n5,  reg_sample_n37,
                reg_sample_6n, reg_sample_38n, reg_sample_n6,  reg_sample_n38,
```

```
303                 reg_sample_7n, reg_sample_39n, reg_sample_n7,  reg_sample_n39,
304                 reg_sample_8n, reg_sample_40n, reg_sample_n8,  reg_sample_n40,
305                 reg_sample_9n, reg_sample_41n, reg_sample_n9,  reg_sample_n41,
306                 reg_sample_10n,   reg_sample_42n, reg_sample_n10, reg_sample_n42,
307                 reg_sample_11n,   reg_sample_43n, reg_sample_n11, reg_sample_n43,
308                 reg_sample_12n,   reg_sample_44n, reg_sample_n12, reg_sample_n44,
309                 reg_sample_13n,   reg_sample_45n, reg_sample_n13, reg_sample_n45,
310                 reg_sample_14n,   reg_sample_46n, reg_sample_n14, reg_sample_n46,
311                 reg_sample_15n,   reg_sample_47n, reg_sample_n15, reg_sample_n47,
312                 reg_sample_16n,   reg_sample_48n, reg_sample_n16, reg_sample_n48,
313                 reg_sample_17n,   reg_sample_49n, reg_sample_n17, reg_sample_n49,
314                 reg_sample_18n,   reg_sample_50n, reg_sample_n18, reg_sample_n50,
315                 reg_sample_19n,   reg_sample_51n, reg_sample_n19, reg_sample_n51,
316                 reg_sample_20n,   reg_sample_52n, reg_sample_n20, reg_sample_n52,
317                 reg_sample_21n,   reg_sample_53n, reg_sample_n21, reg_sample_n53,
318                 reg_sample_22n,   reg_sample_54n, reg_sample_n22, reg_sample_n54,
319                 reg_sample_23n,   reg_sample_55n, reg_sample_n23, reg_sample_n55,
320                 reg_sample_24n,   reg_sample_56n, reg_sample_n24, reg_sample_n56,
321                 reg_sample_25n,   reg_sample_57n, reg_sample_n25, reg_sample_n57,
322                 reg_sample_26n,   reg_sample_58n, reg_sample_n26, reg_sample_n58,
323                 reg_sample_27n,   reg_sample_59n, reg_sample_n27, reg_sample_n59,
324                 reg_sample_28n,   reg_sample_60n, reg_sample_n28, reg_sample_n60,
325                 reg_sample_29n,   reg_sample_61n, reg_sample_n29, reg_sample_n61,
326                 reg_sample_30n,   reg_sample_62n, reg_sample_n30, reg_sample_n62,
327                 reg_sample_31n,   reg_sample_63n, reg_sample_n31, reg_sample_n63,
328
329                 reg_sample_Nn,
330                 reg_sample_nN,
331                 reg_sample_NN,
332
333                 reg_planar_sample_0n,
334                 reg_planar_sample_1n,
335                 reg_planar_sample_2n,
336                 reg_planar_sample_3n,
337                 reg_planar_sample_n0,
338                 reg_planar_sample_n1,
339                 reg_planar_sample_n2,
340                 reg_planar_sample_n3,
341
342                 reg_dc_sample_0n,
343                 reg_dc_sample_1n,
344                 reg_dc_sample_2n,
345                 reg_dc_sample_3n,
346                 reg_dc_sample_n0,
347                 reg_dc_sample_n1,
348                 reg_dc_sample_n2,
349                 reg_dc_sample_n3,
350
351                 reg_angular_mode2_ref1,
352                 reg_angular_mode2_ref2,
353                 reg_angular_mode2_ref3,
354                 reg_angular_mode2_ref4,
355                 reg_angular_mode2_ref5,
356
357                 reg_angular_mode3_ref1,
358                 reg_angular_mode3_ref2,
359                 reg_angular_mode3_ref3,
```

```
360                 reg_angular_mode3_ref4,
361                 reg_angular_mode3_ref5,
362
363                 reg_angular_mode4_ref1,
364                 reg_angular_mode4_ref2,
365                 reg_angular_mode4_ref3,
366                 reg_angular_mode4_ref4,
367                 reg_angular_mode4_ref5,
368
369                 reg_angular_mode5_ref1,
370                 reg_angular_mode5_ref2,
371                 reg_angular_mode5_ref3,
372                 reg_angular_mode5_ref4,
373                 reg_angular_mode5_ref5,
374
375                 reg_angular_mode6_ref1,
376                 reg_angular_mode6_ref2,
377                 reg_angular_mode6_ref3,
378                 reg_angular_mode6_ref4,
379                 reg_angular_mode6_ref5,
380
381                 reg_angular_mode7_ref1,
382                 reg_angular_mode7_ref2,
383                 reg_angular_mode7_ref3,
384                 reg_angular_mode7_ref4,
385                 reg_angular_mode7_ref5,
386
387                 reg_angular_mode8_ref1,
388                 reg_angular_mode8_ref2,
389                 reg_angular_mode8_ref3,
390                 reg_angular_mode8_ref4,
391                 reg_angular_mode8_ref5,
392
393                 reg_angular_mode9_ref1,
394                 reg_angular_mode9_ref2,
395                 reg_angular_mode9_ref3,
396                 reg_angular_mode9_ref4,
397                 reg_angular_mode9_ref5,
398
399                 reg_angular_mode10_ref0,
400                 reg_angular_mode10_ref1,
401                 reg_angular_mode10_ref2,
402                 reg_angular_mode10_ref3,
403                 reg_angular_mode10_ref4,
404                 reg_angular_mode10_ref5,
405
406                 reg_angular_mode11_ref1,
407                 reg_angular_mode11_ref2,
408                 reg_angular_mode11_ref3,
409                 reg_angular_mode11_ref4,
410                 reg_angular_mode11_ref5,
411
412                 reg_angular_mode12_ref1,
413                 reg_angular_mode12_ref2,
414                 reg_angular_mode12_ref3,
415                 reg_angular_mode12_ref4,
416                 reg_angular_mode12_ref5,
```

```
417
418                 reg_angular_mode13_ref1,
419                 reg_angular_mode13_ref2,
420                 reg_angular_mode13_ref3,
421                 reg_angular_mode13_ref4,
422                 reg_angular_mode13_ref5,
423
424                 reg_angular_mode14_ref1,
425                 reg_angular_mode14_ref2,
426                 reg_angular_mode14_ref3,
427                 reg_angular_mode14_ref4,
428                 reg_angular_mode14_ref5,
429
430                 reg_angular_mode15_ref1,
431                 reg_angular_mode15_ref2,
432                 reg_angular_mode15_ref3,
433                 reg_angular_mode15_ref4,
434                 reg_angular_mode15_ref5,
435
436                 reg_angular_mode16_ref1,
437                 reg_angular_mode16_ref2,
438                 reg_angular_mode16_ref3,
439                 reg_angular_mode16_ref4,
440                 reg_angular_mode16_ref5,
441
442                 reg_angular_mode17_ref1,
443                 reg_angular_mode17_ref2,
444                 reg_angular_mode17_ref3,
445                 reg_angular_mode17_ref4,
446                 reg_angular_mode17_ref5,
447
448                 reg_angular_mode18_ref1,
449                 reg_angular_mode18_ref2,
450                 reg_angular_mode18_ref3,
451                 reg_angular_mode18_ref4,
452                 reg_angular_mode18_ref5,
453
454                 reg_angular_mode19_ref1,
455                 reg_angular_mode19_ref2,
456                 reg_angular_mode19_ref3,
457                 reg_angular_mode19_ref4,
458                 reg_angular_mode19_ref5,
459
460                 reg_angular_mode20_ref1,
461                 reg_angular_mode20_ref2,
462                 reg_angular_mode20_ref3,
463                 reg_angular_mode20_ref4,
464                 reg_angular_mode20_ref5,
465
466                 reg_angular_mode21_ref1,
467                 reg_angular_mode21_ref2,
468                 reg_angular_mode21_ref3,
469                 reg_angular_mode21_ref4,
470                 reg_angular_mode21_ref5,
471
472                 reg_angular_mode22_ref1,
473                 reg_angular_mode22_ref2,
```

```
474                 reg_angular_mode22_ref3,
475                 reg_angular_mode22_ref4,
476                 reg_angular_mode22_ref5,
477
478                 reg_angular_mode23_ref1,
479                 reg_angular_mode23_ref2,
480                 reg_angular_mode23_ref3,
481                 reg_angular_mode23_ref4,
482                 reg_angular_mode23_ref5,
483
484                 reg_angular_mode24_ref1,
485                 reg_angular_mode24_ref2,
486                 reg_angular_mode24_ref3,
487                 reg_angular_mode24_ref4,
488                 reg_angular_mode24_ref5,
489
490                 reg_angular_mode25_ref1,
491                 reg_angular_mode25_ref2,
492                 reg_angular_mode25_ref3,
493                 reg_angular_mode25_ref4,
494                 reg_angular_mode25_ref5,
495
496                 reg_angular_mode26_ref0,
497                 reg_angular_mode26_ref1,
498                 reg_angular_mode26_ref2,
499                 reg_angular_mode26_ref3,
500                 reg_angular_mode26_ref4,
501                 reg_angular_mode26_ref5,
502
503                 reg_angular_mode27_ref1,
504                 reg_angular_mode27_ref2,
505                 reg_angular_mode27_ref3,
506                 reg_angular_mode27_ref4,
507                 reg_angular_mode27_ref5,
508
509                 reg_angular_mode28_ref1,
510                 reg_angular_mode28_ref2,
511                 reg_angular_mode28_ref3,
512                 reg_angular_mode28_ref4,
513                 reg_angular_mode28_ref5,
514
515                 reg_angular_mode29_ref1,
516                 reg_angular_mode29_ref2,
517                 reg_angular_mode29_ref3,
518                 reg_angular_mode29_ref4,
519                 reg_angular_mode29_ref5,
520
521                 reg_angular_mode30_ref1,
522                 reg_angular_mode30_ref2,
523                 reg_angular_mode30_ref3,
524                 reg_angular_mode30_ref4,
525                 reg_angular_mode30_ref5,
526
527                 reg_angular_mode31_ref1,
528                 reg_angular_mode31_ref2,
529                 reg_angular_mode31_ref3,
530                 reg_angular_mode31_ref4,
```

```
531                 reg_angular_mode31_ref5,
532
533                 reg_angular_mode32_ref1,
534                 reg_angular_mode32_ref2,
535                 reg_angular_mode32_ref3,
536                 reg_angular_mode32_ref4,
537                 reg_angular_mode32_ref5,
538
539                 reg_angular_mode33_ref1,
540                 reg_angular_mode33_ref2,
541                 reg_angular_mode33_ref3,
542                 reg_angular_mode33_ref4,
543                 reg_angular_mode33_ref5,
544
545                 reg_angular_mode34_ref1,
546                 reg_angular_mode34_ref2,
547                 reg_angular_mode34_ref3,
548                 reg_angular_mode34_ref4,
549                 reg_angular_mode34_ref5;
550
551     wire[WIDTH-4:0] i;
552
553     wire[WIDTH-1:0] angular_mode2_sample1,
554                 angular_mode2_sample2,
555                 angular_mode2_sample3,
556                 angular_mode2_sample4,
557                 angular_mode2_sample5,
558
559                 angular_mode3_sample1,
560                 angular_mode3_sample2,
561                 angular_mode3_sample3,
562                 angular_mode3_sample4,
563                 angular_mode3_sample5,
564
565                 angular_mode4_sample1,
566                 angular_mode4_sample2,
567                 angular_mode4_sample3,
568                 angular_mode4_sample4,
569                 angular_mode4_sample5,
570
571                 angular_mode5_sample1,
572                 angular_mode5_sample2,
573                 angular_mode5_sample3,
574                 angular_mode5_sample4,
575                 angular_mode5_sample5,
576
577                 angular_mode6_sample1,
578                 angular_mode6_sample2,
579                 angular_mode6_sample3,
580                 angular_mode6_sample4,
581                 angular_mode6_sample5,
582
583                 angular_mode7_sample1,
584                 angular_mode7_sample2,
585                 angular_mode7_sample3,
586                 angular_mode7_sample4,
587                 angular_mode7_sample5,
```

```
                angular_mode8_sample1,
                angular_mode8_sample2,
                angular_mode8_sample3,
                angular_mode8_sample4,
                angular_mode8_sample5,

                angular_mode9_sample1,
                angular_mode9_sample2,
                angular_mode9_sample3,
                angular_mode9_sample4,
                angular_mode9_sample5,

                angular_mode10_sample0,
                angular_mode10_sample1,
                angular_mode10_sample2,
                angular_mode10_sample3,
                angular_mode10_sample4,
                angular_mode10_sample5,

                angular_mode11_sample1,
                angular_mode11_sample2,
                angular_mode11_sample3,
                angular_mode11_sample4,
                angular_mode11_sample5,

                angular_mode12_sample1,
                angular_mode12_sample2,
                angular_mode12_sample3,
                angular_mode12_sample4,
                angular_mode12_sample5,

                angular_mode13_sample1,
                angular_mode13_sample2,
                angular_mode13_sample3,
                angular_mode13_sample4,
                angular_mode13_sample5,

                angular_mode14_sample1,
                angular_mode14_sample2,
                angular_mode14_sample3,
                angular_mode14_sample4,
                angular_mode14_sample5,

                angular_mode15_sample1,
                angular_mode15_sample2,
                angular_mode15_sample3,
                angular_mode15_sample4,
                angular_mode15_sample5,

                angular_mode16_sample1,
                angular_mode16_sample2,
                angular_mode16_sample3,
                angular_mode16_sample4,
                angular_mode16_sample5,

                angular_mode17_sample1,
```

```
                angular_mode17_sample2,
                angular_mode17_sample3,
                angular_mode17_sample4,
                angular_mode17_sample5,

                angular_mode18_sample1,
                angular_mode18_sample2,
                angular_mode18_sample3,
                angular_mode18_sample4,
                angular_mode18_sample5,

                angular_mode19_sample1,
                angular_mode19_sample2,
                angular_mode19_sample3,
                angular_mode19_sample4,
                angular_mode19_sample5,

                angular_mode20_sample1,
                angular_mode20_sample2,
                angular_mode20_sample3,
                angular_mode20_sample4,
                angular_mode20_sample5,

                angular_mode21_sample1,
                angular_mode21_sample2,
                angular_mode21_sample3,
                angular_mode21_sample4,
                angular_mode21_sample5,

                angular_mode22_sample1,
                angular_mode22_sample2,
                angular_mode22_sample3,
                angular_mode22_sample4,
                angular_mode22_sample5,

                angular_mode23_sample1,
                angular_mode23_sample2,
                angular_mode23_sample3,
                angular_mode23_sample4,
                angular_mode23_sample5,

                angular_mode24_sample1,
                angular_mode24_sample2,
                angular_mode24_sample3,
                angular_mode24_sample4,
                angular_mode24_sample5,

                angular_mode25_sample1,
                angular_mode25_sample2,
                angular_mode25_sample3,
                angular_mode25_sample4,
                angular_mode25_sample5,

                angular_mode26_sample0,
                angular_mode26_sample1,
                angular_mode26_sample2,
                angular_mode26_sample3,
```

```
                angular_mode26_sample4,
                angular_mode26_sample5,

                angular_mode27_sample1,
                angular_mode27_sample2,
                angular_mode27_sample3,
                angular_mode27_sample4,
                angular_mode27_sample5,

                angular_mode28_sample1,
                angular_mode28_sample2,
                angular_mode28_sample3,
                angular_mode28_sample4,
                angular_mode28_sample5,

                angular_mode29_sample1,
                angular_mode29_sample2,
                angular_mode29_sample3,
                angular_mode29_sample4,
                angular_mode29_sample5,

                angular_mode30_sample1,
                angular_mode30_sample2,
                angular_mode30_sample3,
                angular_mode30_sample4,
                angular_mode30_sample5,

                angular_mode31_sample1,
                angular_mode31_sample2,
                angular_mode31_sample3,
                angular_mode31_sample4,
                angular_mode31_sample5,

                angular_mode32_sample1,
                angular_mode32_sample2,
                angular_mode32_sample3,
                angular_mode32_sample4,
                angular_mode32_sample5,

                angular_mode33_sample1,
                angular_mode33_sample2,
                angular_mode33_sample3,
                angular_mode33_sample4,
                angular_mode33_sample5,

                angular_mode34_sample1,
                angular_mode34_sample2,
                angular_mode34_sample3,
                angular_mode34_sample4,
                angular_mode34_sample5;

    angular_refs_selector #(WIDTH, 2) angular_mode2_selector(
        .clock(clock),
        .selector(angular_mode2_samples),
        .sample_0n(reg_sample_0n),  .sample_32n(reg_sample_32n), .sample_n0(
            reg_sample_n0),   .sample_n32(reg_sample_n32),
        .sample_1n(reg_sample_1n),  .sample_33n(reg_sample_33n), .sample_n1(
```

```
            reg_sample_n1),   .sample_n33(reg_sample_n33),
758         .sample_2n(reg_sample_2n),  .sample_34n(reg_sample_34n), .sample_n2(
               reg_sample_n2),   .sample_n34(reg_sample_n34),
759         .sample_3n(reg_sample_3n),  .sample_35n(reg_sample_35n), .sample_n3(
               reg_sample_n3),   .sample_n35(reg_sample_n35),
760         .sample_4n(reg_sample_4n),  .sample_36n(reg_sample_36n), .sample_n4(
               reg_sample_n4),   .sample_n36(reg_sample_n36),
761         .sample_5n(reg_sample_5n),  .sample_37n(reg_sample_37n), .sample_n5(
               reg_sample_n5),   .sample_n37(reg_sample_n37),
762         .sample_6n(reg_sample_6n),  .sample_38n(reg_sample_38n), .sample_n6(
               reg_sample_n6),   .sample_n38(reg_sample_n38),
763         .sample_7n(reg_sample_7n),  .sample_39n(reg_sample_39n), .sample_n7(
               reg_sample_n7),   .sample_n39(reg_sample_n39),
764         .sample_8n(reg_sample_8n),  .sample_40n(reg_sample_40n), .sample_n8(
               reg_sample_n8),   .sample_n40(reg_sample_n40),
765         .sample_9n(reg_sample_9n),  .sample_41n(reg_sample_41n), .sample_n9(
               reg_sample_n9),   .sample_n41(reg_sample_n41),
766         .sample_10n(reg_sample_10n), .sample_42n(reg_sample_42n), .sample_n10(
               reg_sample_n10), .sample_n42(reg_sample_n42),
767         .sample_11n(reg_sample_11n), .sample_43n(reg_sample_43n), .sample_n11(
               reg_sample_n11), .sample_n43(reg_sample_n43),
768         .sample_12n(reg_sample_12n), .sample_44n(reg_sample_44n), .sample_n12(
               reg_sample_n12), .sample_n44(reg_sample_n44),
769         .sample_13n(reg_sample_13n), .sample_45n(reg_sample_45n), .sample_n13(
               reg_sample_n13), .sample_n45(reg_sample_n45),
770         .sample_14n(reg_sample_14n), .sample_46n(reg_sample_46n), .sample_n14(
               reg_sample_n14), .sample_n46(reg_sample_n46),
771         .sample_15n(reg_sample_15n), .sample_47n(reg_sample_47n), .sample_n15(
               reg_sample_n15), .sample_n47(reg_sample_n47),
772         .sample_16n(reg_sample_16n), .sample_48n(reg_sample_48n), .sample_n16(
               reg_sample_n16), .sample_n48(reg_sample_n48),
773         .sample_17n(reg_sample_17n), .sample_49n(reg_sample_49n), .sample_n17(
               reg_sample_n17), .sample_n49(reg_sample_n49),
774         .sample_18n(reg_sample_18n), .sample_50n(reg_sample_50n), .sample_n18(
               reg_sample_n18), .sample_n50(reg_sample_n50),
775         .sample_19n(reg_sample_19n), .sample_51n(reg_sample_51n), .sample_n19(
               reg_sample_n19), .sample_n51(reg_sample_n51),
776         .sample_20n(reg_sample_20n), .sample_52n(reg_sample_52n), .sample_n20(
               reg_sample_n20), .sample_n52(reg_sample_n52),
777         .sample_21n(reg_sample_21n), .sample_53n(reg_sample_53n), .sample_n21(
               reg_sample_n21), .sample_n53(reg_sample_n53),
778         .sample_22n(reg_sample_22n), .sample_54n(reg_sample_54n), .sample_n22(
               reg_sample_n22), .sample_n54(reg_sample_n54),
779         .sample_23n(reg_sample_23n), .sample_55n(reg_sample_55n), .sample_n23(
               reg_sample_n23), .sample_n55(reg_sample_n55),
780         .sample_24n(reg_sample_24n), .sample_56n(reg_sample_56n), .sample_n24(
               reg_sample_n24), .sample_n56(reg_sample_n56),
781         .sample_25n(reg_sample_25n), .sample_57n(reg_sample_57n), .sample_n25(
               reg_sample_n25), .sample_n57(reg_sample_n57),
782         .sample_26n(reg_sample_26n), .sample_58n(reg_sample_58n), .sample_n26(
               reg_sample_n26), .sample_n58(reg_sample_n58),
783         .sample_27n(reg_sample_27n), .sample_59n(reg_sample_59n), .sample_n27(
               reg_sample_n27), .sample_n59(reg_sample_n59),
784         .sample_28n(reg_sample_28n), .sample_60n(reg_sample_60n), .sample_n28(
               reg_sample_n28), .sample_n60(reg_sample_n60),
785         .sample_29n(reg_sample_29n), .sample_61n(reg_sample_61n), .sample_n29(
               reg_sample_n29), .sample_n61(reg_sample_n61),
```

```
786         .sample_30n(reg_sample_30n), .sample_62n(reg_sample_62n), .sample_n30(
                reg_sample_n30), .sample_n62(reg_sample_n62),
787         .sample_31n(reg_sample_31n), .sample_63n(reg_sample_63n), .sample_n31(
                reg_sample_n31), .sample_n63(reg_sample_n63),
788
789         .sample_NN(reg_sample_NN),
790
791         .ref1(angular_mode2_sample1),
792         .ref2(angular_mode2_sample2),
793         .ref3(angular_mode2_sample3),
794         .ref4(angular_mode2_sample4),
795         .ref5(angular_mode2_sample5)
796     );
797
798     angular_refs_selector #(WIDTH, 3) angular_mode3_selector(
799         .clock(clock),
800         .selector(angular_mode3_samples),
801         .sample_0n(reg_sample_0n),  .sample_32n(reg_sample_32n), .sample_n0(
                reg_sample_n0),   .sample_n32(reg_sample_n32),
802         .sample_1n(reg_sample_1n),  .sample_33n(reg_sample_33n), .sample_n1(
                reg_sample_n1),   .sample_n33(reg_sample_n33),
803         .sample_2n(reg_sample_2n),  .sample_34n(reg_sample_34n), .sample_n2(
                reg_sample_n2),   .sample_n34(reg_sample_n34),
804         .sample_3n(reg_sample_3n),  .sample_35n(reg_sample_35n), .sample_n3(
                reg_sample_n3),   .sample_n35(reg_sample_n35),
805         .sample_4n(reg_sample_4n),  .sample_36n(reg_sample_36n), .sample_n4(
                reg_sample_n4),   .sample_n36(reg_sample_n36),
806         .sample_5n(reg_sample_5n),  .sample_37n(reg_sample_37n), .sample_n5(
                reg_sample_n5),   .sample_n37(reg_sample_n37),
807         .sample_6n(reg_sample_6n),  .sample_38n(reg_sample_38n), .sample_n6(
                reg_sample_n6),   .sample_n38(reg_sample_n38),
808         .sample_7n(reg_sample_7n),  .sample_39n(reg_sample_39n), .sample_n7(
                reg_sample_n7),   .sample_n39(reg_sample_n39),
809         .sample_8n(reg_sample_8n),  .sample_40n(reg_sample_40n), .sample_n8(
                reg_sample_n8),   .sample_n40(reg_sample_n40),
810         .sample_9n(reg_sample_9n),  .sample_41n(reg_sample_41n), .sample_n9(
                reg_sample_n9),   .sample_n41(reg_sample_n41),
811         .sample_10n(reg_sample_10n), .sample_42n(reg_sample_42n), .sample_n10(
                reg_sample_n10), .sample_n42(reg_sample_n42),
812         .sample_11n(reg_sample_11n), .sample_43n(reg_sample_43n), .sample_n11(
                reg_sample_n11), .sample_n43(reg_sample_n43),
813         .sample_12n(reg_sample_12n), .sample_44n(reg_sample_44n), .sample_n12(
                reg_sample_n12), .sample_n44(reg_sample_n44),
814         .sample_13n(reg_sample_13n), .sample_45n(reg_sample_45n), .sample_n13(
                reg_sample_n13), .sample_n45(reg_sample_n45),
815         .sample_14n(reg_sample_14n), .sample_46n(reg_sample_46n), .sample_n14(
                reg_sample_n14), .sample_n46(reg_sample_n46),
816         .sample_15n(reg_sample_15n), .sample_47n(reg_sample_47n), .sample_n15(
                reg_sample_n15), .sample_n47(reg_sample_n47),
817         .sample_16n(reg_sample_16n), .sample_48n(reg_sample_48n), .sample_n16(
                reg_sample_n16), .sample_n48(reg_sample_n48),
818         .sample_17n(reg_sample_17n), .sample_49n(reg_sample_49n), .sample_n17(
                reg_sample_n17), .sample_n49(reg_sample_n49),
819         .sample_18n(reg_sample_18n), .sample_50n(reg_sample_50n), .sample_n18(
                reg_sample_n18), .sample_n50(reg_sample_n50),
820         .sample_19n(reg_sample_19n), .sample_51n(reg_sample_51n), .sample_n19(
                reg_sample_n19), .sample_n51(reg_sample_n51),
```

```
821         .sample_20n(reg_sample_20n), .sample_52n(reg_sample_52n), .sample_n20(
                reg_sample_n20), .sample_n52(reg_sample_n52),
822         .sample_21n(reg_sample_21n), .sample_53n(reg_sample_53n), .sample_n21(
                reg_sample_n21), .sample_n53(reg_sample_n53),
823         .sample_22n(reg_sample_22n), .sample_54n(reg_sample_54n), .sample_n22(
                reg_sample_n22), .sample_n54(reg_sample_n54),
824         .sample_23n(reg_sample_23n), .sample_55n(reg_sample_55n), .sample_n23(
                reg_sample_n23), .sample_n55(reg_sample_n55),
825         .sample_24n(reg_sample_24n), .sample_56n(reg_sample_56n), .sample_n24(
                reg_sample_n24), .sample_n56(reg_sample_n56),
826         .sample_25n(reg_sample_25n), .sample_57n(reg_sample_57n), .sample_n25(
                reg_sample_n25), .sample_n57(reg_sample_n57),
827         .sample_26n(reg_sample_26n), .sample_58n(reg_sample_58n), .sample_n26(
                reg_sample_n26), .sample_n58(reg_sample_n58),
828         .sample_27n(reg_sample_27n), .sample_59n(reg_sample_59n), .sample_n27(
                reg_sample_n27), .sample_n59(reg_sample_n59),
829         .sample_28n(reg_sample_28n), .sample_60n(reg_sample_60n), .sample_n28(
                reg_sample_n28), .sample_n60(reg_sample_n60),
830         .sample_29n(reg_sample_29n), .sample_61n(reg_sample_61n), .sample_n29(
                reg_sample_n29), .sample_n61(reg_sample_n61),
831         .sample_30n(reg_sample_30n), .sample_62n(reg_sample_62n), .sample_n30(
                reg_sample_n30), .sample_n62(reg_sample_n62),
832         .sample_31n(reg_sample_31n), .sample_63n(reg_sample_63n), .sample_n31(
                reg_sample_n31), .sample_n63(reg_sample_n63),
833
834         .sample_NN(reg_sample_NN),
835
836         .ref1(angular_mode3_sample1),
837         .ref2(angular_mode3_sample2),
838         .ref3(angular_mode3_sample3),
839         .ref4(angular_mode3_sample4),
840         .ref5(angular_mode3_sample5)
841     );
842
843     angular_refs_selector #(WIDTH, 4) angular_mode4_selector(
844         .clock(clock),
845         .selector(angular_mode4_samples),
846         .sample_0n(reg_sample_0n),  .sample_32n(reg_sample_32n), .sample_n0(
                reg_sample_n0),   .sample_n32(reg_sample_n32),
847         .sample_1n(reg_sample_1n),  .sample_33n(reg_sample_33n), .sample_n1(
                reg_sample_n1),   .sample_n33(reg_sample_n33),
848         .sample_2n(reg_sample_2n),  .sample_34n(reg_sample_34n), .sample_n2(
                reg_sample_n2),   .sample_n34(reg_sample_n34),
849         .sample_3n(reg_sample_3n),  .sample_35n(reg_sample_35n), .sample_n3(
                reg_sample_n3),   .sample_n35(reg_sample_n35),
850         .sample_4n(reg_sample_4n),  .sample_36n(reg_sample_36n), .sample_n4(
                reg_sample_n4),   .sample_n36(reg_sample_n36),
851         .sample_5n(reg_sample_5n),  .sample_37n(reg_sample_37n), .sample_n5(
                reg_sample_n5),   .sample_n37(reg_sample_n37),
852         .sample_6n(reg_sample_6n),  .sample_38n(reg_sample_38n), .sample_n6(
                reg_sample_n6),   .sample_n38(reg_sample_n38),
853         .sample_7n(reg_sample_7n),  .sample_39n(reg_sample_39n), .sample_n7(
                reg_sample_n7),   .sample_n39(reg_sample_n39),
854         .sample_8n(reg_sample_8n),  .sample_40n(reg_sample_40n), .sample_n8(
                reg_sample_n8),   .sample_n40(reg_sample_n40),
855         .sample_9n(reg_sample_9n),  .sample_41n(reg_sample_41n), .sample_n9(
                reg_sample_n9),   .sample_n41(reg_sample_n41),
```

```
856         .sample_10n(reg_sample_10n), .sample_42n(reg_sample_42n), .sample_n10(
                reg_sample_n10), .sample_n42(reg_sample_n42),
857         .sample_11n(reg_sample_11n), .sample_43n(reg_sample_43n), .sample_n11(
                reg_sample_n11), .sample_n43(reg_sample_n43),
858         .sample_12n(reg_sample_12n), .sample_44n(reg_sample_44n), .sample_n12(
                reg_sample_n12), .sample_n44(reg_sample_n44),
859         .sample_13n(reg_sample_13n), .sample_45n(reg_sample_45n), .sample_n13(
                reg_sample_n13), .sample_n45(reg_sample_n45),
860         .sample_14n(reg_sample_14n), .sample_46n(reg_sample_46n), .sample_n14(
                reg_sample_n14), .sample_n46(reg_sample_n46),
861         .sample_15n(reg_sample_15n), .sample_47n(reg_sample_47n), .sample_n15(
                reg_sample_n15), .sample_n47(reg_sample_n47),
862         .sample_16n(reg_sample_16n), .sample_48n(reg_sample_48n), .sample_n16(
                reg_sample_n16), .sample_n48(reg_sample_n48),
863         .sample_17n(reg_sample_17n), .sample_49n(reg_sample_49n), .sample_n17(
                reg_sample_n17), .sample_n49(reg_sample_n49),
864         .sample_18n(reg_sample_18n), .sample_50n(reg_sample_50n), .sample_n18(
                reg_sample_n18), .sample_n50(reg_sample_n50),
865         .sample_19n(reg_sample_19n), .sample_51n(reg_sample_51n), .sample_n19(
                reg_sample_n19), .sample_n51(reg_sample_n51),
866         .sample_20n(reg_sample_20n), .sample_52n(reg_sample_52n), .sample_n20(
                reg_sample_n20), .sample_n52(reg_sample_n52),
867         .sample_21n(reg_sample_21n), .sample_53n(reg_sample_53n), .sample_n21(
                reg_sample_n21), .sample_n53(reg_sample_n53),
868         .sample_22n(reg_sample_22n), .sample_54n(reg_sample_54n), .sample_n22(
                reg_sample_n22), .sample_n54(reg_sample_n54),
869         .sample_23n(reg_sample_23n), .sample_55n(reg_sample_55n), .sample_n23(
                reg_sample_n23), .sample_n55(reg_sample_n55),
870         .sample_24n(reg_sample_24n), .sample_56n(reg_sample_56n), .sample_n24(
                reg_sample_n24), .sample_n56(reg_sample_n56),
871         .sample_25n(reg_sample_25n), .sample_57n(reg_sample_57n), .sample_n25(
                reg_sample_n25), .sample_n57(reg_sample_n57),
872         .sample_26n(reg_sample_26n), .sample_58n(reg_sample_58n), .sample_n26(
                reg_sample_n26), .sample_n58(reg_sample_n58),
873         .sample_27n(reg_sample_27n), .sample_59n(reg_sample_59n), .sample_n27(
                reg_sample_n27), .sample_n59(reg_sample_n59),
874         .sample_28n(reg_sample_28n), .sample_60n(reg_sample_60n), .sample_n28(
                reg_sample_n28), .sample_n60(reg_sample_n60),
875         .sample_29n(reg_sample_29n), .sample_61n(reg_sample_61n), .sample_n29(
                reg_sample_n29), .sample_n61(reg_sample_n61),
876         .sample_30n(reg_sample_30n), .sample_62n(reg_sample_62n), .sample_n30(
                reg_sample_n30), .sample_n62(reg_sample_n62),
877         .sample_31n(reg_sample_31n), .sample_63n(reg_sample_63n), .sample_n31(
                reg_sample_n31), .sample_n63(reg_sample_n63),
878
879         .sample_NN(reg_sample_NN),
880
881         .ref1(angular_mode4_sample1),
882         .ref2(angular_mode4_sample2),
883         .ref3(angular_mode4_sample3),
884         .ref4(angular_mode4_sample4),
885         .ref5(angular_mode4_sample5)
886     );
887
888     angular_refs_selector #(WIDTH, 5) angular_mode5_selector(
889         .clock(clock),
890         .selector(angular_mode5_samples),
```

```
891         .sample_0n(reg_sample_0n),  .sample_32n(reg_sample_32n), .sample_n0(
                reg_sample_n0),   .sample_n32(reg_sample_n32),
892         .sample_1n(reg_sample_1n),  .sample_33n(reg_sample_33n), .sample_n1(
                reg_sample_n1),   .sample_n33(reg_sample_n33),
893         .sample_2n(reg_sample_2n),  .sample_34n(reg_sample_34n), .sample_n2(
                reg_sample_n2),   .sample_n34(reg_sample_n34),
894         .sample_3n(reg_sample_3n),  .sample_35n(reg_sample_35n), .sample_n3(
                reg_sample_n3),   .sample_n35(reg_sample_n35),
895         .sample_4n(reg_sample_4n),  .sample_36n(reg_sample_36n), .sample_n4(
                reg_sample_n4),   .sample_n36(reg_sample_n36),
896         .sample_5n(reg_sample_5n),  .sample_37n(reg_sample_37n), .sample_n5(
                reg_sample_n5),   .sample_n37(reg_sample_n37),
897         .sample_6n(reg_sample_6n),  .sample_38n(reg_sample_38n), .sample_n6(
                reg_sample_n6),   .sample_n38(reg_sample_n38),
898         .sample_7n(reg_sample_7n),  .sample_39n(reg_sample_39n), .sample_n7(
                reg_sample_n7),   .sample_n39(reg_sample_n39),
899         .sample_8n(reg_sample_8n),  .sample_40n(reg_sample_40n), .sample_n8(
                reg_sample_n8),   .sample_n40(reg_sample_n40),
900         .sample_9n(reg_sample_9n),  .sample_41n(reg_sample_41n), .sample_n9(
                reg_sample_n9),   .sample_n41(reg_sample_n41),
901         .sample_10n(reg_sample_10n), .sample_42n(reg_sample_42n), .sample_n10(
                reg_sample_n10), .sample_n42(reg_sample_n42),
902         .sample_11n(reg_sample_11n), .sample_43n(reg_sample_43n), .sample_n11(
                reg_sample_n11), .sample_n43(reg_sample_n43),
903         .sample_12n(reg_sample_12n), .sample_44n(reg_sample_44n), .sample_n12(
                reg_sample_n12), .sample_n44(reg_sample_n44),
904         .sample_13n(reg_sample_13n), .sample_45n(reg_sample_45n), .sample_n13(
                reg_sample_n13), .sample_n45(reg_sample_n45),
905         .sample_14n(reg_sample_14n), .sample_46n(reg_sample_46n), .sample_n14(
                reg_sample_n14), .sample_n46(reg_sample_n46),
906         .sample_15n(reg_sample_15n), .sample_47n(reg_sample_47n), .sample_n15(
                reg_sample_n15), .sample_n47(reg_sample_n47),
907         .sample_16n(reg_sample_16n), .sample_48n(reg_sample_48n), .sample_n16(
                reg_sample_n16), .sample_n48(reg_sample_n48),
908         .sample_17n(reg_sample_17n), .sample_49n(reg_sample_49n), .sample_n17(
                reg_sample_n17), .sample_n49(reg_sample_n49),
909         .sample_18n(reg_sample_18n), .sample_50n(reg_sample_50n), .sample_n18(
                reg_sample_n18), .sample_n50(reg_sample_n50),
910         .sample_19n(reg_sample_19n), .sample_51n(reg_sample_51n), .sample_n19(
                reg_sample_n19), .sample_n51(reg_sample_n51),
911         .sample_20n(reg_sample_20n), .sample_52n(reg_sample_52n), .sample_n20(
                reg_sample_n20), .sample_n52(reg_sample_n52),
912         .sample_21n(reg_sample_21n), .sample_53n(reg_sample_53n), .sample_n21(
                reg_sample_n21), .sample_n53(reg_sample_n53),
913         .sample_22n(reg_sample_22n), .sample_54n(reg_sample_54n), .sample_n22(
                reg_sample_n22), .sample_n54(reg_sample_n54),
914         .sample_23n(reg_sample_23n), .sample_55n(reg_sample_55n), .sample_n23(
                reg_sample_n23), .sample_n55(reg_sample_n55),
915         .sample_24n(reg_sample_24n), .sample_56n(reg_sample_56n), .sample_n24(
                reg_sample_n24), .sample_n56(reg_sample_n56),
916         .sample_25n(reg_sample_25n), .sample_57n(reg_sample_57n), .sample_n25(
                reg_sample_n25), .sample_n57(reg_sample_n57),
917         .sample_26n(reg_sample_26n), .sample_58n(reg_sample_58n), .sample_n26(
                reg_sample_n26), .sample_n58(reg_sample_n58),
918         .sample_27n(reg_sample_27n), .sample_59n(reg_sample_59n), .sample_n27(
                reg_sample_n27), .sample_n59(reg_sample_n59),
919         .sample_28n(reg_sample_28n), .sample_60n(reg_sample_60n), .sample_n28(
```

```
            reg_sample_n28), .sample_n60(reg_sample_n60),
920         .sample_29n(reg_sample_29n), .sample_61n(reg_sample_61n), .sample_n29(
            reg_sample_n29), .sample_n61(reg_sample_n61),
921         .sample_30n(reg_sample_30n), .sample_62n(reg_sample_62n), .sample_n30(
            reg_sample_n30), .sample_n62(reg_sample_n62),
922         .sample_31n(reg_sample_31n), .sample_63n(reg_sample_63n), .sample_n31(
            reg_sample_n31), .sample_n63(reg_sample_n63),
923
924         .sample_NN(reg_sample_NN),
925
926         .ref1(angular_mode5_sample1),
927         .ref2(angular_mode5_sample2),
928         .ref3(angular_mode5_sample3),
929         .ref4(angular_mode5_sample4),
930         .ref5(angular_mode5_sample5)
931     );
932
933     angular_refs_selector #(WIDTH, 6) angular_mode6_selector(
934         .clock(clock),
935         .selector(angular_mode6_samples),
936         .sample_0n(reg_sample_0n),  .sample_32n(reg_sample_32n), .sample_n0(
            reg_sample_n0),   .sample_n32(reg_sample_n32),
937         .sample_1n(reg_sample_1n),  .sample_33n(reg_sample_33n), .sample_n1(
            reg_sample_n1),   .sample_n33(reg_sample_n33),
938         .sample_2n(reg_sample_2n),  .sample_34n(reg_sample_34n), .sample_n2(
            reg_sample_n2),   .sample_n34(reg_sample_n34),
939         .sample_3n(reg_sample_3n),  .sample_35n(reg_sample_35n), .sample_n3(
            reg_sample_n3),   .sample_n35(reg_sample_n35),
940         .sample_4n(reg_sample_4n),  .sample_36n(reg_sample_36n), .sample_n4(
            reg_sample_n4),   .sample_n36(reg_sample_n36),
941         .sample_5n(reg_sample_5n),  .sample_37n(reg_sample_37n), .sample_n5(
            reg_sample_n5),   .sample_n37(reg_sample_n37),
942         .sample_6n(reg_sample_6n),  .sample_38n(reg_sample_38n), .sample_n6(
            reg_sample_n6),   .sample_n38(reg_sample_n38),
943         .sample_7n(reg_sample_7n),  .sample_39n(reg_sample_39n), .sample_n7(
            reg_sample_n7),   .sample_n39(reg_sample_n39),
944         .sample_8n(reg_sample_8n),  .sample_40n(reg_sample_40n), .sample_n8(
            reg_sample_n8),   .sample_n40(reg_sample_n40),
945         .sample_9n(reg_sample_9n),  .sample_41n(reg_sample_41n), .sample_n9(
            reg_sample_n9),   .sample_n41(reg_sample_n41),
946         .sample_10n(reg_sample_10n), .sample_42n(reg_sample_42n), .sample_n10(
            reg_sample_n10), .sample_n42(reg_sample_n42),
947         .sample_11n(reg_sample_11n), .sample_43n(reg_sample_43n), .sample_n11(
            reg_sample_n11), .sample_n43(reg_sample_n43),
948         .sample_12n(reg_sample_12n), .sample_44n(reg_sample_44n), .sample_n12(
            reg_sample_n12), .sample_n44(reg_sample_n44),
949         .sample_13n(reg_sample_13n), .sample_45n(reg_sample_45n), .sample_n13(
            reg_sample_n13), .sample_n45(reg_sample_n45),
950         .sample_14n(reg_sample_14n), .sample_46n(reg_sample_46n), .sample_n14(
            reg_sample_n14), .sample_n46(reg_sample_n46),
951         .sample_15n(reg_sample_15n), .sample_47n(reg_sample_47n), .sample_n15(
            reg_sample_n15), .sample_n47(reg_sample_n47),
952         .sample_16n(reg_sample_16n), .sample_48n(reg_sample_48n), .sample_n16(
            reg_sample_n16), .sample_n48(reg_sample_n48),
953         .sample_17n(reg_sample_17n), .sample_49n(reg_sample_49n), .sample_n17(
            reg_sample_n17), .sample_n49(reg_sample_n49),
954         .sample_18n(reg_sample_18n), .sample_50n(reg_sample_50n), .sample_n18(
```

```
            reg_sample_n18), .sample_n50(reg_sample_n50),
955         .sample_19n(reg_sample_19n), .sample_51n(reg_sample_51n), .sample_n19(
                reg_sample_n19), .sample_n51(reg_sample_n51),
956         .sample_20n(reg_sample_20n), .sample_52n(reg_sample_52n), .sample_n20(
                reg_sample_n20), .sample_n52(reg_sample_n52),
957         .sample_21n(reg_sample_21n), .sample_53n(reg_sample_53n), .sample_n21(
                reg_sample_n21), .sample_n53(reg_sample_n53),
958         .sample_22n(reg_sample_22n), .sample_54n(reg_sample_54n), .sample_n22(
                reg_sample_n22), .sample_n54(reg_sample_n54),
959         .sample_23n(reg_sample_23n), .sample_55n(reg_sample_55n), .sample_n23(
                reg_sample_n23), .sample_n55(reg_sample_n55),
960         .sample_24n(reg_sample_24n), .sample_56n(reg_sample_56n), .sample_n24(
                reg_sample_n24), .sample_n56(reg_sample_n56),
961         .sample_25n(reg_sample_25n), .sample_57n(reg_sample_57n), .sample_n25(
                reg_sample_n25), .sample_n57(reg_sample_n57),
962         .sample_26n(reg_sample_26n), .sample_58n(reg_sample_58n), .sample_n26(
                reg_sample_n26), .sample_n58(reg_sample_n58),
963         .sample_27n(reg_sample_27n), .sample_59n(reg_sample_59n), .sample_n27(
                reg_sample_n27), .sample_n59(reg_sample_n59),
964         .sample_28n(reg_sample_28n), .sample_60n(reg_sample_60n), .sample_n28(
                reg_sample_n28), .sample_n60(reg_sample_n60),
965         .sample_29n(reg_sample_29n), .sample_61n(reg_sample_61n), .sample_n29(
                reg_sample_n29), .sample_n61(reg_sample_n61),
966         .sample_30n(reg_sample_30n), .sample_62n(reg_sample_62n), .sample_n30(
                reg_sample_n30), .sample_n62(reg_sample_n62),
967         .sample_31n(reg_sample_31n), .sample_63n(reg_sample_63n), .sample_n31(
                reg_sample_n31), .sample_n63(reg_sample_n63),
968
969         .sample_NN(reg_sample_NN),
970
971         .ref1(angular_mode6_sample1),
972         .ref2(angular_mode6_sample2),
973         .ref3(angular_mode6_sample3),
974         .ref4(angular_mode6_sample4),
975         .ref5(angular_mode6_sample5)
976     );
977
978     angular_refs_selector #(WIDTH, 7) angular_mode7_selector(
979         .clock(clock),
980         .selector(angular_mode7_samples),
981         .sample_0n(reg_sample_0n),  .sample_32n(reg_sample_32n), .sample_n0(
                reg_sample_n0),   .sample_n32(reg_sample_n32),
982         .sample_1n(reg_sample_1n),  .sample_33n(reg_sample_33n), .sample_n1(
                reg_sample_n1),   .sample_n33(reg_sample_n33),
983         .sample_2n(reg_sample_2n),  .sample_34n(reg_sample_34n), .sample_n2(
                reg_sample_n2),   .sample_n34(reg_sample_n34),
984         .sample_3n(reg_sample_3n),  .sample_35n(reg_sample_35n), .sample_n3(
                reg_sample_n3),   .sample_n35(reg_sample_n35),
985         .sample_4n(reg_sample_4n),  .sample_36n(reg_sample_36n), .sample_n4(
                reg_sample_n4),   .sample_n36(reg_sample_n36),
986         .sample_5n(reg_sample_5n),  .sample_37n(reg_sample_37n), .sample_n5(
                reg_sample_n5),   .sample_n37(reg_sample_n37),
987         .sample_6n(reg_sample_6n),  .sample_38n(reg_sample_38n), .sample_n6(
                reg_sample_n6),   .sample_n38(reg_sample_n38),
988         .sample_7n(reg_sample_7n),  .sample_39n(reg_sample_39n), .sample_n7(
                reg_sample_n7),   .sample_n39(reg_sample_n39),
989         .sample_8n(reg_sample_8n),  .sample_40n(reg_sample_40n), .sample_n8(
```

```
            reg_sample_n8),   .sample_n40(reg_sample_n40),
990         .sample_9n(reg_sample_9n),  .sample_41n(reg_sample_41n), .sample_n9(
               reg_sample_n9),   .sample_n41(reg_sample_n41),
991         .sample_10n(reg_sample_10n), .sample_42n(reg_sample_42n), .sample_n10(
               reg_sample_n10), .sample_n42(reg_sample_n42),
992         .sample_11n(reg_sample_11n), .sample_43n(reg_sample_43n), .sample_n11(
               reg_sample_n11), .sample_n43(reg_sample_n43),
993         .sample_12n(reg_sample_12n), .sample_44n(reg_sample_44n), .sample_n12(
               reg_sample_n12), .sample_n44(reg_sample_n44),
994         .sample_13n(reg_sample_13n), .sample_45n(reg_sample_45n), .sample_n13(
               reg_sample_n13), .sample_n45(reg_sample_n45),
995         .sample_14n(reg_sample_14n), .sample_46n(reg_sample_46n), .sample_n14(
               reg_sample_n14), .sample_n46(reg_sample_n46),
996         .sample_15n(reg_sample_15n), .sample_47n(reg_sample_47n), .sample_n15(
               reg_sample_n15), .sample_n47(reg_sample_n47),
997         .sample_16n(reg_sample_16n), .sample_48n(reg_sample_48n), .sample_n16(
               reg_sample_n16), .sample_n48(reg_sample_n48),
998         .sample_17n(reg_sample_17n), .sample_49n(reg_sample_49n), .sample_n17(
               reg_sample_n17), .sample_n49(reg_sample_n49),
999         .sample_18n(reg_sample_18n), .sample_50n(reg_sample_50n), .sample_n18(
               reg_sample_n18), .sample_n50(reg_sample_n50),
1000        .sample_19n(reg_sample_19n), .sample_51n(reg_sample_51n), .sample_n19(
               reg_sample_n19), .sample_n51(reg_sample_n51),
1001        .sample_20n(reg_sample_20n), .sample_52n(reg_sample_52n), .sample_n20(
               reg_sample_n20), .sample_n52(reg_sample_n52),
1002        .sample_21n(reg_sample_21n), .sample_53n(reg_sample_53n), .sample_n21(
               reg_sample_n21), .sample_n53(reg_sample_n53),
1003        .sample_22n(reg_sample_22n), .sample_54n(reg_sample_54n), .sample_n22(
               reg_sample_n22), .sample_n54(reg_sample_n54),
1004        .sample_23n(reg_sample_23n), .sample_55n(reg_sample_55n), .sample_n23(
               reg_sample_n23), .sample_n55(reg_sample_n55),
1005        .sample_24n(reg_sample_24n), .sample_56n(reg_sample_56n), .sample_n24(
               reg_sample_n24), .sample_n56(reg_sample_n56),
1006        .sample_25n(reg_sample_25n), .sample_57n(reg_sample_57n), .sample_n25(
               reg_sample_n25), .sample_n57(reg_sample_n57),
1007        .sample_26n(reg_sample_26n), .sample_58n(reg_sample_58n), .sample_n26(
               reg_sample_n26), .sample_n58(reg_sample_n58),
1008        .sample_27n(reg_sample_27n), .sample_59n(reg_sample_59n), .sample_n27(
               reg_sample_n27), .sample_n59(reg_sample_n59),
1009        .sample_28n(reg_sample_28n), .sample_60n(reg_sample_60n), .sample_n28(
               reg_sample_n28), .sample_n60(reg_sample_n60),
1010        .sample_29n(reg_sample_29n), .sample_61n(reg_sample_61n), .sample_n29(
               reg_sample_n29), .sample_n61(reg_sample_n61),
1011        .sample_30n(reg_sample_30n), .sample_62n(reg_sample_62n), .sample_n30(
               reg_sample_n30), .sample_n62(reg_sample_n62),
1012        .sample_31n(reg_sample_31n), .sample_63n(reg_sample_63n), .sample_n31(
               reg_sample_n31), .sample_n63(reg_sample_n63),
1013
1014        .sample_NN(reg_sample_NN),
1015
1016        .ref1(angular_mode7_sample1),
1017        .ref2(angular_mode7_sample2),
1018        .ref3(angular_mode7_sample3),
1019        .ref4(angular_mode7_sample4),
1020        .ref5(angular_mode7_sample5)
1021    );
1022
```

```
    angular_refs_selector #(WIDTH, 8) angular_mode8_selector(
       .clock(clock),
       .selector(angular_mode8_samples),
       .sample_0n(reg_sample_0n),  .sample_32n(reg_sample_32n), .sample_n0(
          reg_sample_n0),   .sample_n32(reg_sample_n32),
       .sample_1n(reg_sample_1n),  .sample_33n(reg_sample_33n), .sample_n1(
          reg_sample_n1),   .sample_n33(reg_sample_n33),
       .sample_2n(reg_sample_2n),  .sample_34n(reg_sample_34n), .sample_n2(
          reg_sample_n2),   .sample_n34(reg_sample_n34),
       .sample_3n(reg_sample_3n),  .sample_35n(reg_sample_35n), .sample_n3(
          reg_sample_n3),   .sample_n35(reg_sample_n35),
       .sample_4n(reg_sample_4n),  .sample_36n(reg_sample_36n), .sample_n4(
          reg_sample_n4),   .sample_n36(reg_sample_n36),
       .sample_5n(reg_sample_5n),  .sample_37n(reg_sample_37n), .sample_n5(
          reg_sample_n5),   .sample_n37(reg_sample_n37),
       .sample_6n(reg_sample_6n),  .sample_38n(reg_sample_38n), .sample_n6(
          reg_sample_n6),   .sample_n38(reg_sample_n38),
       .sample_7n(reg_sample_7n),  .sample_39n(reg_sample_39n), .sample_n7(
          reg_sample_n7),   .sample_n39(reg_sample_n39),
       .sample_8n(reg_sample_8n),  .sample_40n(reg_sample_40n), .sample_n8(
          reg_sample_n8),   .sample_n40(reg_sample_n40),
       .sample_9n(reg_sample_9n),  .sample_41n(reg_sample_41n), .sample_n9(
          reg_sample_n9),   .sample_n41(reg_sample_n41),
       .sample_10n(reg_sample_10n), .sample_42n(reg_sample_42n), .sample_n10(
          reg_sample_n10), .sample_n42(reg_sample_n42),
       .sample_11n(reg_sample_11n), .sample_43n(reg_sample_43n), .sample_n11(
          reg_sample_n11), .sample_n43(reg_sample_n43),
       .sample_12n(reg_sample_12n), .sample_44n(reg_sample_44n), .sample_n12(
          reg_sample_n12), .sample_n44(reg_sample_n44),
       .sample_13n(reg_sample_13n), .sample_45n(reg_sample_45n), .sample_n13(
          reg_sample_n13), .sample_n45(reg_sample_n45),
       .sample_14n(reg_sample_14n), .sample_46n(reg_sample_46n), .sample_n14(
          reg_sample_n14), .sample_n46(reg_sample_n46),
       .sample_15n(reg_sample_15n), .sample_47n(reg_sample_47n), .sample_n15(
          reg_sample_n15), .sample_n47(reg_sample_n47),
       .sample_16n(reg_sample_16n), .sample_48n(reg_sample_48n), .sample_n16(
          reg_sample_n16), .sample_n48(reg_sample_n48),
       .sample_17n(reg_sample_17n), .sample_49n(reg_sample_49n), .sample_n17(
          reg_sample_n17), .sample_n49(reg_sample_n49),
       .sample_18n(reg_sample_18n), .sample_50n(reg_sample_50n), .sample_n18(
          reg_sample_n18), .sample_n50(reg_sample_n50),
       .sample_19n(reg_sample_19n), .sample_51n(reg_sample_51n), .sample_n19(
          reg_sample_n19), .sample_n51(reg_sample_n51),
       .sample_20n(reg_sample_20n), .sample_52n(reg_sample_52n), .sample_n20(
          reg_sample_n20), .sample_n52(reg_sample_n52),
       .sample_21n(reg_sample_21n), .sample_53n(reg_sample_53n), .sample_n21(
          reg_sample_n21), .sample_n53(reg_sample_n53),
       .sample_22n(reg_sample_22n), .sample_54n(reg_sample_54n), .sample_n22(
          reg_sample_n22), .sample_n54(reg_sample_n54),
       .sample_23n(reg_sample_23n), .sample_55n(reg_sample_55n), .sample_n23(
          reg_sample_n23), .sample_n55(reg_sample_n55),
       .sample_24n(reg_sample_24n), .sample_56n(reg_sample_56n), .sample_n24(
          reg_sample_n24), .sample_n56(reg_sample_n56),
       .sample_25n(reg_sample_25n), .sample_57n(reg_sample_57n), .sample_n25(
          reg_sample_n25), .sample_n57(reg_sample_n57),
       .sample_26n(reg_sample_26n), .sample_58n(reg_sample_58n), .sample_n26(
          reg_sample_n26), .sample_n58(reg_sample_n58),
```

```
1053        .sample_27n(reg_sample_27n), .sample_59n(reg_sample_59n), .sample_n27(
                reg_sample_n27), .sample_n59(reg_sample_n59),
1054        .sample_28n(reg_sample_28n), .sample_60n(reg_sample_60n), .sample_n28(
                reg_sample_n28), .sample_n60(reg_sample_n60),
1055        .sample_29n(reg_sample_29n), .sample_61n(reg_sample_61n), .sample_n29(
                reg_sample_n29), .sample_n61(reg_sample_n61),
1056        .sample_30n(reg_sample_30n), .sample_62n(reg_sample_62n), .sample_n30(
                reg_sample_n30), .sample_n62(reg_sample_n62),
1057        .sample_31n(reg_sample_31n), .sample_63n(reg_sample_63n), .sample_n31(
                reg_sample_n31), .sample_n63(reg_sample_n63),
1058
1059        .sample_NN(reg_sample_NN),
1060
1061        .ref1(angular_mode8_sample1),
1062        .ref2(angular_mode8_sample2),
1063        .ref3(angular_mode8_sample3),
1064        .ref4(angular_mode8_sample4),
1065        .ref5(angular_mode8_sample5)
1066    );
1067
1068    angular_refs_selector #(WIDTH, 9) angular_mode9_selector(
1069        .clock(clock),
1070        .selector(angular_mode9_samples),
1071        .sample_0n(reg_sample_0n),  .sample_32n(reg_sample_32n), .sample_n0(
                reg_sample_n0),   .sample_n32(reg_sample_n32),
1072        .sample_1n(reg_sample_1n),  .sample_33n(reg_sample_33n), .sample_n1(
                reg_sample_n1),   .sample_n33(reg_sample_n33),
1073        .sample_2n(reg_sample_2n),  .sample_34n(reg_sample_34n), .sample_n2(
                reg_sample_n2),   .sample_n34(reg_sample_n34),
1074        .sample_3n(reg_sample_3n),  .sample_35n(reg_sample_35n), .sample_n3(
                reg_sample_n3),   .sample_n35(reg_sample_n35),
1075        .sample_4n(reg_sample_4n),  .sample_36n(reg_sample_36n), .sample_n4(
                reg_sample_n4),   .sample_n36(reg_sample_n36),
1076        .sample_5n(reg_sample_5n),  .sample_37n(reg_sample_37n), .sample_n5(
                reg_sample_n5),   .sample_n37(reg_sample_n37),
1077        .sample_6n(reg_sample_6n),  .sample_38n(reg_sample_38n), .sample_n6(
                reg_sample_n6),   .sample_n38(reg_sample_n38),
1078        .sample_7n(reg_sample_7n),  .sample_39n(reg_sample_39n), .sample_n7(
                reg_sample_n7),   .sample_n39(reg_sample_n39),
1079        .sample_8n(reg_sample_8n),  .sample_40n(reg_sample_40n), .sample_n8(
                reg_sample_n8),   .sample_n40(reg_sample_n40),
1080        .sample_9n(reg_sample_9n),  .sample_41n(reg_sample_41n), .sample_n9(
                reg_sample_n9),   .sample_n41(reg_sample_n41),
1081        .sample_10n(reg_sample_10n), .sample_42n(reg_sample_42n), .sample_n10(
                reg_sample_n10), .sample_n42(reg_sample_n42),
1082        .sample_11n(reg_sample_11n), .sample_43n(reg_sample_43n), .sample_n11(
                reg_sample_n11), .sample_n43(reg_sample_n43),
1083        .sample_12n(reg_sample_12n), .sample_44n(reg_sample_44n), .sample_n12(
                reg_sample_n12), .sample_n44(reg_sample_n44),
1084        .sample_13n(reg_sample_13n), .sample_45n(reg_sample_45n), .sample_n13(
                reg_sample_n13), .sample_n45(reg_sample_n45),
1085        .sample_14n(reg_sample_14n), .sample_46n(reg_sample_46n), .sample_n14(
                reg_sample_n14), .sample_n46(reg_sample_n46),
1086        .sample_15n(reg_sample_15n), .sample_47n(reg_sample_47n), .sample_n15(
                reg_sample_n15), .sample_n47(reg_sample_n47),
1087        .sample_16n(reg_sample_16n), .sample_48n(reg_sample_48n), .sample_n16(
                reg_sample_n16), .sample_n48(reg_sample_n48),
```

```
1088        .sample_17n(reg_sample_17n), .sample_49n(reg_sample_49n), .sample_n17(
                reg_sample_n17), .sample_n49(reg_sample_n49),
1089        .sample_18n(reg_sample_18n), .sample_50n(reg_sample_50n), .sample_n18(
                reg_sample_n18), .sample_n50(reg_sample_n50),
1090        .sample_19n(reg_sample_19n), .sample_51n(reg_sample_51n), .sample_n19(
                reg_sample_n19), .sample_n51(reg_sample_n51),
1091        .sample_20n(reg_sample_20n), .sample_52n(reg_sample_52n), .sample_n20(
                reg_sample_n20), .sample_n52(reg_sample_n52),
1092        .sample_21n(reg_sample_21n), .sample_53n(reg_sample_53n), .sample_n21(
                reg_sample_n21), .sample_n53(reg_sample_n53),
1093        .sample_22n(reg_sample_22n), .sample_54n(reg_sample_54n), .sample_n22(
                reg_sample_n22), .sample_n54(reg_sample_n54),
1094        .sample_23n(reg_sample_23n), .sample_55n(reg_sample_55n), .sample_n23(
                reg_sample_n23), .sample_n55(reg_sample_n55),
1095        .sample_24n(reg_sample_24n), .sample_56n(reg_sample_56n), .sample_n24(
                reg_sample_n24), .sample_n56(reg_sample_n56),
1096        .sample_25n(reg_sample_25n), .sample_57n(reg_sample_57n), .sample_n25(
                reg_sample_n25), .sample_n57(reg_sample_n57),
1097        .sample_26n(reg_sample_26n), .sample_58n(reg_sample_58n), .sample_n26(
                reg_sample_n26), .sample_n58(reg_sample_n58),
1098        .sample_27n(reg_sample_27n), .sample_59n(reg_sample_59n), .sample_n27(
                reg_sample_n27), .sample_n59(reg_sample_n59),
1099        .sample_28n(reg_sample_28n), .sample_60n(reg_sample_60n), .sample_n28(
                reg_sample_n28), .sample_n60(reg_sample_n60),
1100        .sample_29n(reg_sample_29n), .sample_61n(reg_sample_61n), .sample_n29(
                reg_sample_n29), .sample_n61(reg_sample_n61),
1101        .sample_30n(reg_sample_30n), .sample_62n(reg_sample_62n), .sample_n30(
                reg_sample_n30), .sample_n62(reg_sample_n62),
1102        .sample_31n(reg_sample_31n), .sample_63n(reg_sample_63n), .sample_n31(
                reg_sample_n31), .sample_n63(reg_sample_n63),
1103
1104        .sample_NN(reg_sample_NN),
1105
1106        .ref1(angular_mode9_sample1),
1107        .ref2(angular_mode9_sample2),
1108        .ref3(angular_mode9_sample3),
1109        .ref4(angular_mode9_sample4),
1110        .ref5(angular_mode9_sample5)
1111    );
1112
1113    angular_directly_refs_selector #(WIDTH, 10) angular_mode10_selector(
1114        .clock(clock),
1115        .index(filter_samples_mode10),
1116        .selector(angular_mode10_samples),
1117        .sample_0n(reg_sample_0n),  .sample_32n(reg_sample_32n), .sample_n0(
                reg_sample_n0),   .sample_n32(reg_sample_n32),
1118        .sample_1n(reg_sample_1n),  .sample_33n(reg_sample_33n), .sample_n1(
                reg_sample_n1),   .sample_n33(reg_sample_n33),
1119        .sample_2n(reg_sample_2n),  .sample_34n(reg_sample_34n), .sample_n2(
                reg_sample_n2),   .sample_n34(reg_sample_n34),
1120        .sample_3n(reg_sample_3n),  .sample_35n(reg_sample_35n), .sample_n3(
                reg_sample_n3),   .sample_n35(reg_sample_n35),
1121        .sample_4n(reg_sample_4n),  .sample_36n(reg_sample_36n), .sample_n4(
                reg_sample_n4),   .sample_n36(reg_sample_n36),
1122        .sample_5n(reg_sample_5n),  .sample_37n(reg_sample_37n), .sample_n5(
                reg_sample_n5),   .sample_n37(reg_sample_n37),
1123        .sample_6n(reg_sample_6n),  .sample_38n(reg_sample_38n), .sample_n6(
```

```
            reg_sample_n6),   .sample_n38(reg_sample_n38),
1124        .sample_7n(reg_sample_7n),  .sample_39n(reg_sample_39n), .sample_n7(
            reg_sample_n7),   .sample_n39(reg_sample_n39),
1125        .sample_8n(reg_sample_8n),  .sample_40n(reg_sample_40n), .sample_n8(
            reg_sample_n8),   .sample_n40(reg_sample_n40),
1126        .sample_9n(reg_sample_9n),  .sample_41n(reg_sample_41n), .sample_n9(
            reg_sample_n9),   .sample_n41(reg_sample_n41),
1127        .sample_10n(reg_sample_10n), .sample_42n(reg_sample_42n), .sample_n10(
            reg_sample_n10), .sample_n42(reg_sample_n42),
1128        .sample_11n(reg_sample_11n), .sample_43n(reg_sample_43n), .sample_n11(
            reg_sample_n11), .sample_n43(reg_sample_n43),
1129        .sample_12n(reg_sample_12n), .sample_44n(reg_sample_44n), .sample_n12(
            reg_sample_n12), .sample_n44(reg_sample_n44),
1130        .sample_13n(reg_sample_13n), .sample_45n(reg_sample_45n), .sample_n13(
            reg_sample_n13), .sample_n45(reg_sample_n45),
1131        .sample_14n(reg_sample_14n), .sample_46n(reg_sample_46n), .sample_n14(
            reg_sample_n14), .sample_n46(reg_sample_n46),
1132        .sample_15n(reg_sample_15n), .sample_47n(reg_sample_47n), .sample_n15(
            reg_sample_n15), .sample_n47(reg_sample_n47),
1133        .sample_16n(reg_sample_16n), .sample_48n(reg_sample_48n), .sample_n16(
            reg_sample_n16), .sample_n48(reg_sample_n48),
1134        .sample_17n(reg_sample_17n), .sample_49n(reg_sample_49n), .sample_n17(
            reg_sample_n17), .sample_n49(reg_sample_n49),
1135        .sample_18n(reg_sample_18n), .sample_50n(reg_sample_50n), .sample_n18(
            reg_sample_n18), .sample_n50(reg_sample_n50),
1136        .sample_19n(reg_sample_19n), .sample_51n(reg_sample_51n), .sample_n19(
            reg_sample_n19), .sample_n51(reg_sample_n51),
1137        .sample_20n(reg_sample_20n), .sample_52n(reg_sample_52n), .sample_n20(
            reg_sample_n20), .sample_n52(reg_sample_n52),
1138        .sample_21n(reg_sample_21n), .sample_53n(reg_sample_53n), .sample_n21(
            reg_sample_n21), .sample_n53(reg_sample_n53),
1139        .sample_22n(reg_sample_22n), .sample_54n(reg_sample_54n), .sample_n22(
            reg_sample_n22), .sample_n54(reg_sample_n54),
1140        .sample_23n(reg_sample_23n), .sample_55n(reg_sample_55n), .sample_n23(
            reg_sample_n23), .sample_n55(reg_sample_n55),
1141        .sample_24n(reg_sample_24n), .sample_56n(reg_sample_56n), .sample_n24(
            reg_sample_n24), .sample_n56(reg_sample_n56),
1142        .sample_25n(reg_sample_25n), .sample_57n(reg_sample_57n), .sample_n25(
            reg_sample_n25), .sample_n57(reg_sample_n57),
1143        .sample_26n(reg_sample_26n), .sample_58n(reg_sample_58n), .sample_n26(
            reg_sample_n26), .sample_n58(reg_sample_n58),
1144        .sample_27n(reg_sample_27n), .sample_59n(reg_sample_59n), .sample_n27(
            reg_sample_n27), .sample_n59(reg_sample_n59),
1145        .sample_28n(reg_sample_28n), .sample_60n(reg_sample_60n), .sample_n28(
            reg_sample_n28), .sample_n60(reg_sample_n60),
1146        .sample_29n(reg_sample_29n), .sample_61n(reg_sample_61n), .sample_n29(
            reg_sample_n29), .sample_n61(reg_sample_n61),
1147        .sample_30n(reg_sample_30n), .sample_62n(reg_sample_62n), .sample_n30(
            reg_sample_n30), .sample_n62(reg_sample_n62),
1148        .sample_31n(reg_sample_31n), .sample_63n(reg_sample_63n), .sample_n31(
            reg_sample_n31), .sample_n63(reg_sample_n63),
1149
1150        .sample_NN(reg_sample_NN),
1151
1152        .ref0(angular_mode10_sample0),
1153        .ref1(angular_mode10_sample1),
1154        .ref2(angular_mode10_sample2),
```

```
1155        .ref3(angular_mode10_sample3),
1156        .ref4(angular_mode10_sample4),
1157        .ref5(angular_mode10_sample5)
1158    );
1159
1160    angular_refs_selector #(WIDTH, 11) angular_mode11_selector(
1161        .clock(clock),
1162        .selector(angular_mode11_samples),
1163        .sample_0n(reg_sample_0n),  .sample_32n(reg_sample_32n), .sample_n0(
                reg_sample_n0),   .sample_n32(reg_sample_n32),
1164        .sample_1n(reg_sample_1n),  .sample_33n(reg_sample_33n), .sample_n1(
                reg_sample_n1),   .sample_n33(reg_sample_n33),
1165        .sample_2n(reg_sample_2n),  .sample_34n(reg_sample_34n), .sample_n2(
                reg_sample_n2),   .sample_n34(reg_sample_n34),
1166        .sample_3n(reg_sample_3n),  .sample_35n(reg_sample_35n), .sample_n3(
                reg_sample_n3),   .sample_n35(reg_sample_n35),
1167        .sample_4n(reg_sample_4n),  .sample_36n(reg_sample_36n), .sample_n4(
                reg_sample_n4),   .sample_n36(reg_sample_n36),
1168        .sample_5n(reg_sample_5n),  .sample_37n(reg_sample_37n), .sample_n5(
                reg_sample_n5),   .sample_n37(reg_sample_n37),
1169        .sample_6n(reg_sample_6n),  .sample_38n(reg_sample_38n), .sample_n6(
                reg_sample_n6),   .sample_n38(reg_sample_n38),
1170        .sample_7n(reg_sample_7n),  .sample_39n(reg_sample_39n), .sample_n7(
                reg_sample_n7),   .sample_n39(reg_sample_n39),
1171        .sample_8n(reg_sample_8n),  .sample_40n(reg_sample_40n), .sample_n8(
                reg_sample_n8),   .sample_n40(reg_sample_n40),
1172        .sample_9n(reg_sample_9n),  .sample_41n(reg_sample_41n), .sample_n9(
                reg_sample_n9),   .sample_n41(reg_sample_n41),
1173        .sample_10n(reg_sample_10n), .sample_42n(reg_sample_42n), .sample_n10(
                reg_sample_n10), .sample_n42(reg_sample_n42),
1174        .sample_11n(reg_sample_11n), .sample_43n(reg_sample_43n), .sample_n11(
                reg_sample_n11), .sample_n43(reg_sample_n43),
1175        .sample_12n(reg_sample_12n), .sample_44n(reg_sample_44n), .sample_n12(
                reg_sample_n12), .sample_n44(reg_sample_n44),
1176        .sample_13n(reg_sample_13n), .sample_45n(reg_sample_45n), .sample_n13(
                reg_sample_n13), .sample_n45(reg_sample_n45),
1177        .sample_14n(reg_sample_14n), .sample_46n(reg_sample_46n), .sample_n14(
                reg_sample_n14), .sample_n46(reg_sample_n46),
1178        .sample_15n(reg_sample_15n), .sample_47n(reg_sample_47n), .sample_n15(
                reg_sample_n15), .sample_n47(reg_sample_n47),
1179        .sample_16n(reg_sample_16n), .sample_48n(reg_sample_48n), .sample_n16(
                reg_sample_n16), .sample_n48(reg_sample_n48),
1180        .sample_17n(reg_sample_17n), .sample_49n(reg_sample_49n), .sample_n17(
                reg_sample_n17), .sample_n49(reg_sample_n49),
1181        .sample_18n(reg_sample_18n), .sample_50n(reg_sample_50n), .sample_n18(
                reg_sample_n18), .sample_n50(reg_sample_n50),
1182        .sample_19n(reg_sample_19n), .sample_51n(reg_sample_51n), .sample_n19(
                reg_sample_n19), .sample_n51(reg_sample_n51),
1183        .sample_20n(reg_sample_20n), .sample_52n(reg_sample_52n), .sample_n20(
                reg_sample_n20), .sample_n52(reg_sample_n52),
1184        .sample_21n(reg_sample_21n), .sample_53n(reg_sample_53n), .sample_n21(
                reg_sample_n21), .sample_n53(reg_sample_n53),
1185        .sample_22n(reg_sample_22n), .sample_54n(reg_sample_54n), .sample_n22(
                reg_sample_n22), .sample_n54(reg_sample_n54),
1186        .sample_23n(reg_sample_23n), .sample_55n(reg_sample_55n), .sample_n23(
                reg_sample_n23), .sample_n55(reg_sample_n55),
1187        .sample_24n(reg_sample_24n), .sample_56n(reg_sample_56n), .sample_n24(
```

```
            reg_sample_n24), .sample_n56(reg_sample_n56),
1188        .sample_25n(reg_sample_25n), .sample_57n(reg_sample_57n), .sample_n25(
            reg_sample_n25), .sample_n57(reg_sample_n57),
1189        .sample_26n(reg_sample_26n), .sample_58n(reg_sample_58n), .sample_n26(
            reg_sample_n26), .sample_n58(reg_sample_n58),
1190        .sample_27n(reg_sample_27n), .sample_59n(reg_sample_59n), .sample_n27(
            reg_sample_n27), .sample_n59(reg_sample_n59),
1191        .sample_28n(reg_sample_28n), .sample_60n(reg_sample_60n), .sample_n28(
            reg_sample_n28), .sample_n60(reg_sample_n60),
1192        .sample_29n(reg_sample_29n), .sample_61n(reg_sample_61n), .sample_n29(
            reg_sample_n29), .sample_n61(reg_sample_n61),
1193        .sample_30n(reg_sample_30n), .sample_62n(reg_sample_62n), .sample_n30(
            reg_sample_n30), .sample_n62(reg_sample_n62),
1194        .sample_31n(reg_sample_31n), .sample_63n(reg_sample_63n), .sample_n31(
            reg_sample_n31), .sample_n63(reg_sample_n63),
1195
1196        .sample_NN(reg_sample_NN),
1197
1198        .ref1(angular_mode11_sample1),
1199        .ref2(angular_mode11_sample2),
1200        .ref3(angular_mode11_sample3),
1201        .ref4(angular_mode11_sample4),
1202        .ref5(angular_mode11_sample5)
1203    );
1204
1205    angular_refs_selector #(WIDTH, 12) angular_mode12_selector(
1206        .clock(clock),
1207        .selector(angular_mode12_samples),
1208        .sample_0n(reg_sample_0n),  .sample_32n(reg_sample_32n), .sample_n0(
            reg_sample_n0),   .sample_n32(reg_sample_n32),
1209        .sample_1n(reg_sample_1n),  .sample_33n(reg_sample_33n), .sample_n1(
            reg_sample_n1),   .sample_n33(reg_sample_n33),
1210        .sample_2n(reg_sample_2n),  .sample_34n(reg_sample_34n), .sample_n2(
            reg_sample_n2),   .sample_n34(reg_sample_n34),
1211        .sample_3n(reg_sample_3n),  .sample_35n(reg_sample_35n), .sample_n3(
            reg_sample_n3),   .sample_n35(reg_sample_n35),
1212        .sample_4n(reg_sample_4n),  .sample_36n(reg_sample_36n), .sample_n4(
            reg_sample_n4),   .sample_n36(reg_sample_n36),
1213        .sample_5n(reg_sample_5n),  .sample_37n(reg_sample_37n), .sample_n5(
            reg_sample_n5),   .sample_n37(reg_sample_n37),
1214        .sample_6n(reg_sample_6n),  .sample_38n(reg_sample_38n), .sample_n6(
            reg_sample_n6),   .sample_n38(reg_sample_n38),
1215        .sample_7n(reg_sample_7n),  .sample_39n(reg_sample_39n), .sample_n7(
            reg_sample_n7),   .sample_n39(reg_sample_n39),
1216        .sample_8n(reg_sample_8n),  .sample_40n(reg_sample_40n), .sample_n8(
            reg_sample_n8),   .sample_n40(reg_sample_n40),
1217        .sample_9n(reg_sample_9n),  .sample_41n(reg_sample_41n), .sample_n9(
            reg_sample_n9),   .sample_n41(reg_sample_n41),
1218        .sample_10n(reg_sample_10n), .sample_42n(reg_sample_42n), .sample_n10(
            reg_sample_n10), .sample_n42(reg_sample_n42),
1219        .sample_11n(reg_sample_11n), .sample_43n(reg_sample_43n), .sample_n11(
            reg_sample_n11), .sample_n43(reg_sample_n43),
1220        .sample_12n(reg_sample_12n), .sample_44n(reg_sample_44n), .sample_n12(
            reg_sample_n12), .sample_n44(reg_sample_n44),
1221        .sample_13n(reg_sample_13n), .sample_45n(reg_sample_45n), .sample_n13(
            reg_sample_n13), .sample_n45(reg_sample_n45),
1222        .sample_14n(reg_sample_14n), .sample_46n(reg_sample_46n), .sample_n14(
```

```
            reg_sample_n14), .sample_n46(reg_sample_n46),
1223        .sample_15n(reg_sample_15n), .sample_47n(reg_sample_47n), .sample_n15(
            reg_sample_n15), .sample_n47(reg_sample_n47),
1224        .sample_16n(reg_sample_16n), .sample_48n(reg_sample_48n), .sample_n16(
            reg_sample_n16), .sample_n48(reg_sample_n48),
1225        .sample_17n(reg_sample_17n), .sample_49n(reg_sample_49n), .sample_n17(
            reg_sample_n17), .sample_n49(reg_sample_n49),
1226        .sample_18n(reg_sample_18n), .sample_50n(reg_sample_50n), .sample_n18(
            reg_sample_n18), .sample_n50(reg_sample_n50),
1227        .sample_19n(reg_sample_19n), .sample_51n(reg_sample_51n), .sample_n19(
            reg_sample_n19), .sample_n51(reg_sample_n51),
1228        .sample_20n(reg_sample_20n), .sample_52n(reg_sample_52n), .sample_n20(
            reg_sample_n20), .sample_n52(reg_sample_n52),
1229        .sample_21n(reg_sample_21n), .sample_53n(reg_sample_53n), .sample_n21(
            reg_sample_n21), .sample_n53(reg_sample_n53),
1230        .sample_22n(reg_sample_22n), .sample_54n(reg_sample_54n), .sample_n22(
            reg_sample_n22), .sample_n54(reg_sample_n54),
1231        .sample_23n(reg_sample_23n), .sample_55n(reg_sample_55n), .sample_n23(
            reg_sample_n23), .sample_n55(reg_sample_n55),
1232        .sample_24n(reg_sample_24n), .sample_56n(reg_sample_56n), .sample_n24(
            reg_sample_n24), .sample_n56(reg_sample_n56),
1233        .sample_25n(reg_sample_25n), .sample_57n(reg_sample_57n), .sample_n25(
            reg_sample_n25), .sample_n57(reg_sample_n57),
1234        .sample_26n(reg_sample_26n), .sample_58n(reg_sample_58n), .sample_n26(
            reg_sample_n26), .sample_n58(reg_sample_n58),
1235        .sample_27n(reg_sample_27n), .sample_59n(reg_sample_59n), .sample_n27(
            reg_sample_n27), .sample_n59(reg_sample_n59),
1236        .sample_28n(reg_sample_28n), .sample_60n(reg_sample_60n), .sample_n28(
            reg_sample_n28), .sample_n60(reg_sample_n60),
1237        .sample_29n(reg_sample_29n), .sample_61n(reg_sample_61n), .sample_n29(
            reg_sample_n29), .sample_n61(reg_sample_n61),
1238        .sample_30n(reg_sample_30n), .sample_62n(reg_sample_62n), .sample_n30(
            reg_sample_n30), .sample_n62(reg_sample_n62),
1239        .sample_31n(reg_sample_31n), .sample_63n(reg_sample_63n), .sample_n31(
            reg_sample_n31), .sample_n63(reg_sample_n63),
1240
1241        .sample_NN(reg_sample_NN),
1242
1243        .ref1(angular_mode12_sample1),
1244        .ref2(angular_mode12_sample2),
1245        .ref3(angular_mode12_sample3),
1246        .ref4(angular_mode12_sample4),
1247        .ref5(angular_mode12_sample5)
1248    );
1249
1250    angular_refs_selector #(WIDTH, 13) angular_mode13_selector(
1251        .clock(clock),
1252        .selector(angular_mode13_samples),
1253        .sample_0n(reg_sample_0n),  .sample_32n(reg_sample_32n), .sample_n0(
            reg_sample_n0),   .sample_n32(reg_sample_n32),
1254        .sample_1n(reg_sample_1n),  .sample_33n(reg_sample_33n), .sample_n1(
            reg_sample_n1),   .sample_n33(reg_sample_n33),
1255        .sample_2n(reg_sample_2n),  .sample_34n(reg_sample_34n), .sample_n2(
            reg_sample_n2),   .sample_n34(reg_sample_n34),
1256        .sample_3n(reg_sample_3n),  .sample_35n(reg_sample_35n), .sample_n3(
            reg_sample_n3),   .sample_n35(reg_sample_n35),
1257        .sample_4n(reg_sample_4n),  .sample_36n(reg_sample_36n), .sample_n4(
```

```
            reg_sample_n4),   .sample_n36(reg_sample_n36),
1258        .sample_5n(reg_sample_5n),  .sample_37n(reg_sample_37n), .sample_n5(
            reg_sample_n5),   .sample_n37(reg_sample_n37),
1259        .sample_6n(reg_sample_6n),  .sample_38n(reg_sample_38n), .sample_n6(
            reg_sample_n6),   .sample_n38(reg_sample_n38),
1260        .sample_7n(reg_sample_7n),  .sample_39n(reg_sample_39n), .sample_n7(
            reg_sample_n7),   .sample_n39(reg_sample_n39),
1261        .sample_8n(reg_sample_8n),  .sample_40n(reg_sample_40n), .sample_n8(
            reg_sample_n8),   .sample_n40(reg_sample_n40),
1262        .sample_9n(reg_sample_9n),  .sample_41n(reg_sample_41n), .sample_n9(
            reg_sample_n9),   .sample_n41(reg_sample_n41),
1263        .sample_10n(reg_sample_10n), .sample_42n(reg_sample_42n), .sample_n10(
            reg_sample_n10), .sample_n42(reg_sample_n42),
1264        .sample_11n(reg_sample_11n), .sample_43n(reg_sample_43n), .sample_n11(
            reg_sample_n11), .sample_n43(reg_sample_n43),
1265        .sample_12n(reg_sample_12n), .sample_44n(reg_sample_44n), .sample_n12(
            reg_sample_n12), .sample_n44(reg_sample_n44),
1266        .sample_13n(reg_sample_13n), .sample_45n(reg_sample_45n), .sample_n13(
            reg_sample_n13), .sample_n45(reg_sample_n45),
1267        .sample_14n(reg_sample_14n), .sample_46n(reg_sample_46n), .sample_n14(
            reg_sample_n14), .sample_n46(reg_sample_n46),
1268        .sample_15n(reg_sample_15n), .sample_47n(reg_sample_47n), .sample_n15(
            reg_sample_n15), .sample_n47(reg_sample_n47),
1269        .sample_16n(reg_sample_16n), .sample_48n(reg_sample_48n), .sample_n16(
            reg_sample_n16), .sample_n48(reg_sample_n48),
1270        .sample_17n(reg_sample_17n), .sample_49n(reg_sample_49n), .sample_n17(
            reg_sample_n17), .sample_n49(reg_sample_n49),
1271        .sample_18n(reg_sample_18n), .sample_50n(reg_sample_50n), .sample_n18(
            reg_sample_n18), .sample_n50(reg_sample_n50),
1272        .sample_19n(reg_sample_19n), .sample_51n(reg_sample_51n), .sample_n19(
            reg_sample_n19), .sample_n51(reg_sample_n51),
1273        .sample_20n(reg_sample_20n), .sample_52n(reg_sample_52n), .sample_n20(
            reg_sample_n20), .sample_n52(reg_sample_n52),
1274        .sample_21n(reg_sample_21n), .sample_53n(reg_sample_53n), .sample_n21(
            reg_sample_n21), .sample_n53(reg_sample_n53),
1275        .sample_22n(reg_sample_22n), .sample_54n(reg_sample_54n), .sample_n22(
            reg_sample_n22), .sample_n54(reg_sample_n54),
1276        .sample_23n(reg_sample_23n), .sample_55n(reg_sample_55n), .sample_n23(
            reg_sample_n23), .sample_n55(reg_sample_n55),
1277        .sample_24n(reg_sample_24n), .sample_56n(reg_sample_56n), .sample_n24(
            reg_sample_n24), .sample_n56(reg_sample_n56),
1278        .sample_25n(reg_sample_25n), .sample_57n(reg_sample_57n), .sample_n25(
            reg_sample_n25), .sample_n57(reg_sample_n57),
1279        .sample_26n(reg_sample_26n), .sample_58n(reg_sample_58n), .sample_n26(
            reg_sample_n26), .sample_n58(reg_sample_n58),
1280        .sample_27n(reg_sample_27n), .sample_59n(reg_sample_59n), .sample_n27(
            reg_sample_n27), .sample_n59(reg_sample_n59),
1281        .sample_28n(reg_sample_28n), .sample_60n(reg_sample_60n), .sample_n28(
            reg_sample_n28), .sample_n60(reg_sample_n60),
1282        .sample_29n(reg_sample_29n), .sample_61n(reg_sample_61n), .sample_n29(
            reg_sample_n29), .sample_n61(reg_sample_n61),
1283        .sample_30n(reg_sample_30n), .sample_62n(reg_sample_62n), .sample_n30(
            reg_sample_n30), .sample_n62(reg_sample_n62),
1284        .sample_31n(reg_sample_31n), .sample_63n(reg_sample_63n), .sample_n31(
            reg_sample_n31), .sample_n63(reg_sample_n63),
1285
1286        .sample_NN(reg_sample_NN),
```

```
1287
1288        .ref1(angular_mode13_sample1),
1289        .ref2(angular_mode13_sample2),
1290        .ref3(angular_mode13_sample3),
1291        .ref4(angular_mode13_sample4),
1292        .ref5(angular_mode13_sample5)
1293    );
1294
1295    angular_refs_selector #(WIDTH, 14) angular_mode14_selector(
1296        .clock(clock),
1297        .selector(angular_mode14_samples),
1298        .sample_0n(reg_sample_0n),  .sample_32n(reg_sample_32n), .sample_n0(
                reg_sample_n0),   .sample_n32(reg_sample_n32),
1299        .sample_1n(reg_sample_1n),  .sample_33n(reg_sample_33n), .sample_n1(
                reg_sample_n1),   .sample_n33(reg_sample_n33),
1300        .sample_2n(reg_sample_2n),  .sample_34n(reg_sample_34n), .sample_n2(
                reg_sample_n2),   .sample_n34(reg_sample_n34),
1301        .sample_3n(reg_sample_3n),  .sample_35n(reg_sample_35n), .sample_n3(
                reg_sample_n3),   .sample_n35(reg_sample_n35),
1302        .sample_4n(reg_sample_4n),  .sample_36n(reg_sample_36n), .sample_n4(
                reg_sample_n4),   .sample_n36(reg_sample_n36),
1303        .sample_5n(reg_sample_5n),  .sample_37n(reg_sample_37n), .sample_n5(
                reg_sample_n5),   .sample_n37(reg_sample_n37),
1304        .sample_6n(reg_sample_6n),  .sample_38n(reg_sample_38n), .sample_n6(
                reg_sample_n6),   .sample_n38(reg_sample_n38),
1305        .sample_7n(reg_sample_7n),  .sample_39n(reg_sample_39n), .sample_n7(
                reg_sample_n7),   .sample_n39(reg_sample_n39),
1306        .sample_8n(reg_sample_8n),  .sample_40n(reg_sample_40n), .sample_n8(
                reg_sample_n8),   .sample_n40(reg_sample_n40),
1307        .sample_9n(reg_sample_9n),  .sample_41n(reg_sample_41n), .sample_n9(
                reg_sample_n9),   .sample_n41(reg_sample_n41),
1308        .sample_10n(reg_sample_10n), .sample_42n(reg_sample_42n), .sample_n10(
                reg_sample_n10), .sample_n42(reg_sample_n42),
1309        .sample_11n(reg_sample_11n), .sample_43n(reg_sample_43n), .sample_n11(
                reg_sample_n11), .sample_n43(reg_sample_n43),
1310        .sample_12n(reg_sample_12n), .sample_44n(reg_sample_44n), .sample_n12(
                reg_sample_n12), .sample_n44(reg_sample_n44),
1311        .sample_13n(reg_sample_13n), .sample_45n(reg_sample_45n), .sample_n13(
                reg_sample_n13), .sample_n45(reg_sample_n45),
1312        .sample_14n(reg_sample_14n), .sample_46n(reg_sample_46n), .sample_n14(
                reg_sample_n14), .sample_n46(reg_sample_n46),
1313        .sample_15n(reg_sample_15n), .sample_47n(reg_sample_47n), .sample_n15(
                reg_sample_n15), .sample_n47(reg_sample_n47),
1314        .sample_16n(reg_sample_16n), .sample_48n(reg_sample_48n), .sample_n16(
                reg_sample_n16), .sample_n48(reg_sample_n48),
1315        .sample_17n(reg_sample_17n), .sample_49n(reg_sample_49n), .sample_n17(
                reg_sample_n17), .sample_n49(reg_sample_n49),
1316        .sample_18n(reg_sample_18n), .sample_50n(reg_sample_50n), .sample_n18(
                reg_sample_n18), .sample_n50(reg_sample_n50),
1317        .sample_19n(reg_sample_19n), .sample_51n(reg_sample_51n), .sample_n19(
                reg_sample_n19), .sample_n51(reg_sample_n51),
1318        .sample_20n(reg_sample_20n), .sample_52n(reg_sample_52n), .sample_n20(
                reg_sample_n20), .sample_n52(reg_sample_n52),
1319        .sample_21n(reg_sample_21n), .sample_53n(reg_sample_53n), .sample_n21(
                reg_sample_n21), .sample_n53(reg_sample_n53),
1320        .sample_22n(reg_sample_22n), .sample_54n(reg_sample_54n), .sample_n22(
                reg_sample_n22), .sample_n54(reg_sample_n54),
```

```
        .sample_23n(reg_sample_23n), .sample_55n(reg_sample_55n), .sample_n23(reg_sample_n23), .sample_n55(reg_sample_n55),
        .sample_24n(reg_sample_24n), .sample_56n(reg_sample_56n), .sample_n24(reg_sample_n24), .sample_n56(reg_sample_n56),
        .sample_25n(reg_sample_25n), .sample_57n(reg_sample_57n), .sample_n25(reg_sample_n25), .sample_n57(reg_sample_n57),
        .sample_26n(reg_sample_26n), .sample_58n(reg_sample_58n), .sample_n26(reg_sample_n26), .sample_n58(reg_sample_n58),
        .sample_27n(reg_sample_27n), .sample_59n(reg_sample_59n), .sample_n27(reg_sample_n27), .sample_n59(reg_sample_n59),
        .sample_28n(reg_sample_28n), .sample_60n(reg_sample_60n), .sample_n28(reg_sample_n28), .sample_n60(reg_sample_n60),
        .sample_29n(reg_sample_29n), .sample_61n(reg_sample_61n), .sample_n29(reg_sample_n29), .sample_n61(reg_sample_n61),
        .sample_30n(reg_sample_30n), .sample_62n(reg_sample_62n), .sample_n30(reg_sample_n30), .sample_n62(reg_sample_n62),
        .sample_31n(reg_sample_31n), .sample_63n(reg_sample_63n), .sample_n31(reg_sample_n31), .sample_n63(reg_sample_n63),

        .sample_NN(reg_sample_NN),

        .ref1(angular_mode14_sample1),
        .ref2(angular_mode14_sample2),
        .ref3(angular_mode14_sample3),
        .ref4(angular_mode14_sample4),
        .ref5(angular_mode14_sample5)
    );

    angular_refs_selector #(WIDTH, 15) angular_mode15_selector(
        .clock(clock),
        .selector(angular_mode15_samples),
        .sample_0n(reg_sample_0n),  .sample_32n(reg_sample_32n), .sample_n0(reg_sample_n0),   .sample_n32(reg_sample_n32),
        .sample_1n(reg_sample_1n),  .sample_33n(reg_sample_33n), .sample_n1(reg_sample_n1),   .sample_n33(reg_sample_n33),
        .sample_2n(reg_sample_2n),  .sample_34n(reg_sample_34n), .sample_n2(reg_sample_n2),   .sample_n34(reg_sample_n34),
        .sample_3n(reg_sample_3n),  .sample_35n(reg_sample_35n), .sample_n3(reg_sample_n3),   .sample_n35(reg_sample_n35),
        .sample_4n(reg_sample_4n),  .sample_36n(reg_sample_36n), .sample_n4(reg_sample_n4),   .sample_n36(reg_sample_n36),
        .sample_5n(reg_sample_5n),  .sample_37n(reg_sample_37n), .sample_n5(reg_sample_n5),   .sample_n37(reg_sample_n37),
        .sample_6n(reg_sample_6n),  .sample_38n(reg_sample_38n), .sample_n6(reg_sample_n6),   .sample_n38(reg_sample_n38),
        .sample_7n(reg_sample_7n),  .sample_39n(reg_sample_39n), .sample_n7(reg_sample_n7),   .sample_n39(reg_sample_n39),
        .sample_8n(reg_sample_8n),  .sample_40n(reg_sample_40n), .sample_n8(reg_sample_n8),   .sample_n40(reg_sample_n40),
        .sample_9n(reg_sample_9n),  .sample_41n(reg_sample_41n), .sample_n9(reg_sample_n9),   .sample_n41(reg_sample_n41),
        .sample_10n(reg_sample_10n), .sample_42n(reg_sample_42n), .sample_n10(reg_sample_n10), .sample_n42(reg_sample_n42),
        .sample_11n(reg_sample_11n), .sample_43n(reg_sample_43n), .sample_n11(reg_sample_n11), .sample_n43(reg_sample_n43),
        .sample_12n(reg_sample_12n), .sample_44n(reg_sample_44n), .sample_n12(reg_sample_n12), .sample_n44(reg_sample_n44),
```

```
1356        .sample_13n(reg_sample_13n), .sample_45n(reg_sample_45n), .sample_n13(
                reg_sample_n13), .sample_n45(reg_sample_n45),
1357        .sample_14n(reg_sample_14n), .sample_46n(reg_sample_46n), .sample_n14(
                reg_sample_n14), .sample_n46(reg_sample_n46),
1358        .sample_15n(reg_sample_15n), .sample_47n(reg_sample_47n), .sample_n15(
                reg_sample_n15), .sample_n47(reg_sample_n47),
1359        .sample_16n(reg_sample_16n), .sample_48n(reg_sample_48n), .sample_n16(
                reg_sample_n16), .sample_n48(reg_sample_n48),
1360        .sample_17n(reg_sample_17n), .sample_49n(reg_sample_49n), .sample_n17(
                reg_sample_n17), .sample_n49(reg_sample_n49),
1361        .sample_18n(reg_sample_18n), .sample_50n(reg_sample_50n), .sample_n18(
                reg_sample_n18), .sample_n50(reg_sample_n50),
1362        .sample_19n(reg_sample_19n), .sample_51n(reg_sample_51n), .sample_n19(
                reg_sample_n19), .sample_n51(reg_sample_n51),
1363        .sample_20n(reg_sample_20n), .sample_52n(reg_sample_52n), .sample_n20(
                reg_sample_n20), .sample_n52(reg_sample_n52),
1364        .sample_21n(reg_sample_21n), .sample_53n(reg_sample_53n), .sample_n21(
                reg_sample_n21), .sample_n53(reg_sample_n53),
1365        .sample_22n(reg_sample_22n), .sample_54n(reg_sample_54n), .sample_n22(
                reg_sample_n22), .sample_n54(reg_sample_n54),
1366        .sample_23n(reg_sample_23n), .sample_55n(reg_sample_55n), .sample_n23(
                reg_sample_n23), .sample_n55(reg_sample_n55),
1367        .sample_24n(reg_sample_24n), .sample_56n(reg_sample_56n), .sample_n24(
                reg_sample_n24), .sample_n56(reg_sample_n56),
1368        .sample_25n(reg_sample_25n), .sample_57n(reg_sample_57n), .sample_n25(
                reg_sample_n25), .sample_n57(reg_sample_n57),
1369        .sample_26n(reg_sample_26n), .sample_58n(reg_sample_58n), .sample_n26(
                reg_sample_n26), .sample_n58(reg_sample_n58),
1370        .sample_27n(reg_sample_27n), .sample_59n(reg_sample_59n), .sample_n27(
                reg_sample_n27), .sample_n59(reg_sample_n59),
1371        .sample_28n(reg_sample_28n), .sample_60n(reg_sample_60n), .sample_n28(
                reg_sample_n28), .sample_n60(reg_sample_n60),
1372        .sample_29n(reg_sample_29n), .sample_61n(reg_sample_61n), .sample_n29(
                reg_sample_n29), .sample_n61(reg_sample_n61),
1373        .sample_30n(reg_sample_30n), .sample_62n(reg_sample_62n), .sample_n30(
                reg_sample_n30), .sample_n62(reg_sample_n62),
1374        .sample_31n(reg_sample_31n), .sample_63n(reg_sample_63n), .sample_n31(
                reg_sample_n31), .sample_n63(reg_sample_n63),
1375
1376        .sample_NN(reg_sample_NN),
1377
1378        .ref1(angular_mode15_sample1),
1379        .ref2(angular_mode15_sample2),
1380        .ref3(angular_mode15_sample3),
1381        .ref4(angular_mode15_sample4),
1382        .ref5(angular_mode15_sample5)
1383    );
1384
1385    angular_refs_selector #(WIDTH, 16) angular_mode16_selector(
1386        .clock(clock),
1387        .selector(angular_mode16_samples),
1388        .sample_0n(reg_sample_0n),  .sample_32n(reg_sample_32n), .sample_n0(
                reg_sample_n0),   .sample_n32(reg_sample_n32),
1389        .sample_1n(reg_sample_1n),  .sample_33n(reg_sample_33n), .sample_n1(
                reg_sample_n1),   .sample_n33(reg_sample_n33),
1390        .sample_2n(reg_sample_2n),  .sample_34n(reg_sample_34n), .sample_n2(
                reg_sample_n2),   .sample_n34(reg_sample_n34),
```

```
1391        .sample_3n(reg_sample_3n),  .sample_35n(reg_sample_35n), .sample_n3(
               reg_sample_n3),   .sample_n35(reg_sample_n35),
1392        .sample_4n(reg_sample_4n),  .sample_36n(reg_sample_36n), .sample_n4(
               reg_sample_n4),   .sample_n36(reg_sample_n36),
1393        .sample_5n(reg_sample_5n),  .sample_37n(reg_sample_37n), .sample_n5(
               reg_sample_n5),   .sample_n37(reg_sample_n37),
1394        .sample_6n(reg_sample_6n),  .sample_38n(reg_sample_38n), .sample_n6(
               reg_sample_n6),   .sample_n38(reg_sample_n38),
1395        .sample_7n(reg_sample_7n),  .sample_39n(reg_sample_39n), .sample_n7(
               reg_sample_n7),   .sample_n39(reg_sample_n39),
1396        .sample_8n(reg_sample_8n),  .sample_40n(reg_sample_40n), .sample_n8(
               reg_sample_n8),   .sample_n40(reg_sample_n40),
1397        .sample_9n(reg_sample_9n),  .sample_41n(reg_sample_41n), .sample_n9(
               reg_sample_n9),   .sample_n41(reg_sample_n41),
1398        .sample_10n(reg_sample_10n), .sample_42n(reg_sample_42n), .sample_n10(
               reg_sample_n10), .sample_n42(reg_sample_n42),
1399        .sample_11n(reg_sample_11n), .sample_43n(reg_sample_43n), .sample_n11(
               reg_sample_n11), .sample_n43(reg_sample_n43),
1400        .sample_12n(reg_sample_12n), .sample_44n(reg_sample_44n), .sample_n12(
               reg_sample_n12), .sample_n44(reg_sample_n44),
1401        .sample_13n(reg_sample_13n), .sample_45n(reg_sample_45n), .sample_n13(
               reg_sample_n13), .sample_n45(reg_sample_n45),
1402        .sample_14n(reg_sample_14n), .sample_46n(reg_sample_46n), .sample_n14(
               reg_sample_n14), .sample_n46(reg_sample_n46),
1403        .sample_15n(reg_sample_15n), .sample_47n(reg_sample_47n), .sample_n15(
               reg_sample_n15), .sample_n47(reg_sample_n47),
1404        .sample_16n(reg_sample_16n), .sample_48n(reg_sample_48n), .sample_n16(
               reg_sample_n16), .sample_n48(reg_sample_n48),
1405        .sample_17n(reg_sample_17n), .sample_49n(reg_sample_49n), .sample_n17(
               reg_sample_n17), .sample_n49(reg_sample_n49),
1406        .sample_18n(reg_sample_18n), .sample_50n(reg_sample_50n), .sample_n18(
               reg_sample_n18), .sample_n50(reg_sample_n50),
1407        .sample_19n(reg_sample_19n), .sample_51n(reg_sample_51n), .sample_n19(
               reg_sample_n19), .sample_n51(reg_sample_n51),
1408        .sample_20n(reg_sample_20n), .sample_52n(reg_sample_52n), .sample_n20(
               reg_sample_n20), .sample_n52(reg_sample_n52),
1409        .sample_21n(reg_sample_21n), .sample_53n(reg_sample_53n), .sample_n21(
               reg_sample_n21), .sample_n53(reg_sample_n53),
1410        .sample_22n(reg_sample_22n), .sample_54n(reg_sample_54n), .sample_n22(
               reg_sample_n22), .sample_n54(reg_sample_n54),
1411        .sample_23n(reg_sample_23n), .sample_55n(reg_sample_55n), .sample_n23(
               reg_sample_n23), .sample_n55(reg_sample_n55),
1412        .sample_24n(reg_sample_24n), .sample_56n(reg_sample_56n), .sample_n24(
               reg_sample_n24), .sample_n56(reg_sample_n56),
1413        .sample_25n(reg_sample_25n), .sample_57n(reg_sample_57n), .sample_n25(
               reg_sample_n25), .sample_n57(reg_sample_n57),
1414        .sample_26n(reg_sample_26n), .sample_58n(reg_sample_58n), .sample_n26(
               reg_sample_n26), .sample_n58(reg_sample_n58),
1415        .sample_27n(reg_sample_27n), .sample_59n(reg_sample_59n), .sample_n27(
               reg_sample_n27), .sample_n59(reg_sample_n59),
1416        .sample_28n(reg_sample_28n), .sample_60n(reg_sample_60n), .sample_n28(
               reg_sample_n28), .sample_n60(reg_sample_n60),
1417        .sample_29n(reg_sample_29n), .sample_61n(reg_sample_61n), .sample_n29(
               reg_sample_n29), .sample_n61(reg_sample_n61),
1418        .sample_30n(reg_sample_30n), .sample_62n(reg_sample_62n), .sample_n30(
               reg_sample_n30), .sample_n62(reg_sample_n62),
1419        .sample_31n(reg_sample_31n), .sample_63n(reg_sample_63n), .sample_n31(
```

```
            reg_sample_n31), .sample_n63(reg_sample_n63),
1420
1421        .sample_NN(reg_sample_NN),
1422
1423        .ref1(angular_mode16_sample1),
1424        .ref2(angular_mode16_sample2),
1425        .ref3(angular_mode16_sample3),
1426        .ref4(angular_mode16_sample4),
1427        .ref5(angular_mode16_sample5)
1428    );
1429
1430    angular_refs_selector #(WIDTH, 17) angular_mode17_selector(
1431        .clock(clock),
1432        .selector(angular_mode17_samples),
1433        .sample_0n(reg_sample_0n),  .sample_32n(reg_sample_32n), .sample_n0(
                reg_sample_n0),   .sample_n32(reg_sample_n32),
1434        .sample_1n(reg_sample_1n),  .sample_33n(reg_sample_33n), .sample_n1(
                reg_sample_n1),   .sample_n33(reg_sample_n33),
1435        .sample_2n(reg_sample_2n),  .sample_34n(reg_sample_34n), .sample_n2(
                reg_sample_n2),   .sample_n34(reg_sample_n34),
1436        .sample_3n(reg_sample_3n),  .sample_35n(reg_sample_35n), .sample_n3(
                reg_sample_n3),   .sample_n35(reg_sample_n35),
1437        .sample_4n(reg_sample_4n),  .sample_36n(reg_sample_36n), .sample_n4(
                reg_sample_n4),   .sample_n36(reg_sample_n36),
1438        .sample_5n(reg_sample_5n),  .sample_37n(reg_sample_37n), .sample_n5(
                reg_sample_n5),   .sample_n37(reg_sample_n37),
1439        .sample_6n(reg_sample_6n),  .sample_38n(reg_sample_38n), .sample_n6(
                reg_sample_n6),   .sample_n38(reg_sample_n38),
1440        .sample_7n(reg_sample_7n),  .sample_39n(reg_sample_39n), .sample_n7(
                reg_sample_n7),   .sample_n39(reg_sample_n39),
1441        .sample_8n(reg_sample_8n),  .sample_40n(reg_sample_40n), .sample_n8(
                reg_sample_n8),   .sample_n40(reg_sample_n40),
1442        .sample_9n(reg_sample_9n),  .sample_41n(reg_sample_41n), .sample_n9(
                reg_sample_n9),   .sample_n41(reg_sample_n41),
1443        .sample_10n(reg_sample_10n), .sample_42n(reg_sample_42n), .sample_n10(
                reg_sample_n10), .sample_n42(reg_sample_n42),
1444        .sample_11n(reg_sample_11n), .sample_43n(reg_sample_43n), .sample_n11(
                reg_sample_n11), .sample_n43(reg_sample_n43),
1445        .sample_12n(reg_sample_12n), .sample_44n(reg_sample_44n), .sample_n12(
                reg_sample_n12), .sample_n44(reg_sample_n44),
1446        .sample_13n(reg_sample_13n), .sample_45n(reg_sample_45n), .sample_n13(
                reg_sample_n13), .sample_n45(reg_sample_n45),
1447        .sample_14n(reg_sample_14n), .sample_46n(reg_sample_46n), .sample_n14(
                reg_sample_n14), .sample_n46(reg_sample_n46),
1448        .sample_15n(reg_sample_15n), .sample_47n(reg_sample_47n), .sample_n15(
                reg_sample_n15), .sample_n47(reg_sample_n47),
1449        .sample_16n(reg_sample_16n), .sample_48n(reg_sample_48n), .sample_n16(
                reg_sample_n16), .sample_n48(reg_sample_n48),
1450        .sample_17n(reg_sample_17n), .sample_49n(reg_sample_49n), .sample_n17(
                reg_sample_n17), .sample_n49(reg_sample_n49),
1451        .sample_18n(reg_sample_18n), .sample_50n(reg_sample_50n), .sample_n18(
                reg_sample_n18), .sample_n50(reg_sample_n50),
1452        .sample_19n(reg_sample_19n), .sample_51n(reg_sample_51n), .sample_n19(
                reg_sample_n19), .sample_n51(reg_sample_n51),
1453        .sample_20n(reg_sample_20n), .sample_52n(reg_sample_52n), .sample_n20(
                reg_sample_n20), .sample_n52(reg_sample_n52),
1454        .sample_21n(reg_sample_21n), .sample_53n(reg_sample_53n), .sample_n21(
```

```
            reg_sample_n21), .sample_n53(reg_sample_n53),
1455        .sample_22n(reg_sample_22n), .sample_54n(reg_sample_54n), .sample_n22(
               reg_sample_n22), .sample_n54(reg_sample_n54),
1456        .sample_23n(reg_sample_23n), .sample_55n(reg_sample_55n), .sample_n23(
               reg_sample_n23), .sample_n55(reg_sample_n55),
1457        .sample_24n(reg_sample_24n), .sample_56n(reg_sample_56n), .sample_n24(
               reg_sample_n24), .sample_n56(reg_sample_n56),
1458        .sample_25n(reg_sample_25n), .sample_57n(reg_sample_57n), .sample_n25(
               reg_sample_n25), .sample_n57(reg_sample_n57),
1459        .sample_26n(reg_sample_26n), .sample_58n(reg_sample_58n), .sample_n26(
               reg_sample_n26), .sample_n58(reg_sample_n58),
1460        .sample_27n(reg_sample_27n), .sample_59n(reg_sample_59n), .sample_n27(
               reg_sample_n27), .sample_n59(reg_sample_n59),
1461        .sample_28n(reg_sample_28n), .sample_60n(reg_sample_60n), .sample_n28(
               reg_sample_n28), .sample_n60(reg_sample_n60),
1462        .sample_29n(reg_sample_29n), .sample_61n(reg_sample_61n), .sample_n29(
               reg_sample_n29), .sample_n61(reg_sample_n61),
1463        .sample_30n(reg_sample_30n), .sample_62n(reg_sample_62n), .sample_n30(
               reg_sample_n30), .sample_n62(reg_sample_n62),
1464        .sample_31n(reg_sample_31n), .sample_63n(reg_sample_63n), .sample_n31(
               reg_sample_n31), .sample_n63(reg_sample_n63),
1465
1466        .sample_NN(reg_sample_NN),
1467
1468        .ref1(angular_mode17_sample1),
1469        .ref2(angular_mode17_sample2),
1470        .ref3(angular_mode17_sample3),
1471        .ref4(angular_mode17_sample4),
1472        .ref5(angular_mode17_sample5)
1473    );
1474
1475    angular_refs_selector #(WIDTH, 18) angular_mode18_selector(
1476        .clock(clock),
1477        .selector(angular_mode18_samples),
1478        .sample_0n(reg_sample_0n),  .sample_32n(reg_sample_32n), .sample_n0(
               reg_sample_n0),   .sample_n32(reg_sample_n32),
1479        .sample_1n(reg_sample_1n),  .sample_33n(reg_sample_33n), .sample_n1(
               reg_sample_n1),   .sample_n33(reg_sample_n33),
1480        .sample_2n(reg_sample_2n),  .sample_34n(reg_sample_34n), .sample_n2(
               reg_sample_n2),   .sample_n34(reg_sample_n34),
1481        .sample_3n(reg_sample_3n),  .sample_35n(reg_sample_35n), .sample_n3(
               reg_sample_n3),   .sample_n35(reg_sample_n35),
1482        .sample_4n(reg_sample_4n),  .sample_36n(reg_sample_36n), .sample_n4(
               reg_sample_n4),   .sample_n36(reg_sample_n36),
1483        .sample_5n(reg_sample_5n),  .sample_37n(reg_sample_37n), .sample_n5(
               reg_sample_n5),   .sample_n37(reg_sample_n37),
1484        .sample_6n(reg_sample_6n),  .sample_38n(reg_sample_38n), .sample_n6(
               reg_sample_n6),   .sample_n38(reg_sample_n38),
1485        .sample_7n(reg_sample_7n),  .sample_39n(reg_sample_39n), .sample_n7(
               reg_sample_n7),   .sample_n39(reg_sample_n39),
1486        .sample_8n(reg_sample_8n),  .sample_40n(reg_sample_40n), .sample_n8(
               reg_sample_n8),   .sample_n40(reg_sample_n40),
1487        .sample_9n(reg_sample_9n),  .sample_41n(reg_sample_41n), .sample_n9(
               reg_sample_n9),   .sample_n41(reg_sample_n41),
1488        .sample_10n(reg_sample_10n), .sample_42n(reg_sample_42n), .sample_n10(
               reg_sample_n10), .sample_n42(reg_sample_n42),
1489        .sample_11n(reg_sample_11n), .sample_43n(reg_sample_43n), .sample_n11(
```

```
            reg_sample_n11), .sample_n43(reg_sample_n43),
1490        .sample_12n(reg_sample_12n), .sample_44n(reg_sample_44n), .sample_n12(
            reg_sample_n12), .sample_n44(reg_sample_n44),
1491        .sample_13n(reg_sample_13n), .sample_45n(reg_sample_45n), .sample_n13(
            reg_sample_n13), .sample_n45(reg_sample_n45),
1492        .sample_14n(reg_sample_14n), .sample_46n(reg_sample_46n), .sample_n14(
            reg_sample_n14), .sample_n46(reg_sample_n46),
1493        .sample_15n(reg_sample_15n), .sample_47n(reg_sample_47n), .sample_n15(
            reg_sample_n15), .sample_n47(reg_sample_n47),
1494        .sample_16n(reg_sample_16n), .sample_48n(reg_sample_48n), .sample_n16(
            reg_sample_n16), .sample_n48(reg_sample_n48),
1495        .sample_17n(reg_sample_17n), .sample_49n(reg_sample_49n), .sample_n17(
            reg_sample_n17), .sample_n49(reg_sample_n49),
1496        .sample_18n(reg_sample_18n), .sample_50n(reg_sample_50n), .sample_n18(
            reg_sample_n18), .sample_n50(reg_sample_n50),
1497        .sample_19n(reg_sample_19n), .sample_51n(reg_sample_51n), .sample_n19(
            reg_sample_n19), .sample_n51(reg_sample_n51),
1498        .sample_20n(reg_sample_20n), .sample_52n(reg_sample_52n), .sample_n20(
            reg_sample_n20), .sample_n52(reg_sample_n52),
1499        .sample_21n(reg_sample_21n), .sample_53n(reg_sample_53n), .sample_n21(
            reg_sample_n21), .sample_n53(reg_sample_n53),
1500        .sample_22n(reg_sample_22n), .sample_54n(reg_sample_54n), .sample_n22(
            reg_sample_n22), .sample_n54(reg_sample_n54),
1501        .sample_23n(reg_sample_23n), .sample_55n(reg_sample_55n), .sample_n23(
            reg_sample_n23), .sample_n55(reg_sample_n55),
1502        .sample_24n(reg_sample_24n), .sample_56n(reg_sample_56n), .sample_n24(
            reg_sample_n24), .sample_n56(reg_sample_n56),
1503        .sample_25n(reg_sample_25n), .sample_57n(reg_sample_57n), .sample_n25(
            reg_sample_n25), .sample_n57(reg_sample_n57),
1504        .sample_26n(reg_sample_26n), .sample_58n(reg_sample_58n), .sample_n26(
            reg_sample_n26), .sample_n58(reg_sample_n58),
1505        .sample_27n(reg_sample_27n), .sample_59n(reg_sample_59n), .sample_n27(
            reg_sample_n27), .sample_n59(reg_sample_n59),
1506        .sample_28n(reg_sample_28n), .sample_60n(reg_sample_60n), .sample_n28(
            reg_sample_n28), .sample_n60(reg_sample_n60),
1507        .sample_29n(reg_sample_29n), .sample_61n(reg_sample_61n), .sample_n29(
            reg_sample_n29), .sample_n61(reg_sample_n61),
1508        .sample_30n(reg_sample_30n), .sample_62n(reg_sample_62n), .sample_n30(
            reg_sample_n30), .sample_n62(reg_sample_n62),
1509        .sample_31n(reg_sample_31n), .sample_63n(reg_sample_63n), .sample_n31(
            reg_sample_n31), .sample_n63(reg_sample_n63),
1510
1511        .sample_NN(reg_sample_NN),
1512
1513        .ref1(angular_mode18_sample1),
1514        .ref2(angular_mode18_sample2),
1515        .ref3(angular_mode18_sample3),
1516        .ref4(angular_mode18_sample4),
1517        .ref5(angular_mode18_sample5)
1518    );
1519
1520    angular_refs_selector #(WIDTH, 19) angular_mode19_selector(
1521        .clock(clock),
1522        .selector(angular_mode19_samples),
1523        .sample_0n(reg_sample_0n),  .sample_32n(reg_sample_32n), .sample_n0(
            reg_sample_n0),   .sample_n32(reg_sample_n32),
1524        .sample_1n(reg_sample_1n),  .sample_33n(reg_sample_33n), .sample_n1(
```

```
            reg_sample_n1),   .sample_n33(reg_sample_n33),
1525        .sample_2n(reg_sample_2n),  .sample_34n(reg_sample_34n), .sample_n2(
            reg_sample_n2),   .sample_n34(reg_sample_n34),
1526        .sample_3n(reg_sample_3n),  .sample_35n(reg_sample_35n), .sample_n3(
            reg_sample_n3),   .sample_n35(reg_sample_n35),
1527        .sample_4n(reg_sample_4n),  .sample_36n(reg_sample_36n), .sample_n4(
            reg_sample_n4),   .sample_n36(reg_sample_n36),
1528        .sample_5n(reg_sample_5n),  .sample_37n(reg_sample_37n), .sample_n5(
            reg_sample_n5),   .sample_n37(reg_sample_n37),
1529        .sample_6n(reg_sample_6n),  .sample_38n(reg_sample_38n), .sample_n6(
            reg_sample_n6),   .sample_n38(reg_sample_n38),
1530        .sample_7n(reg_sample_7n),  .sample_39n(reg_sample_39n), .sample_n7(
            reg_sample_n7),   .sample_n39(reg_sample_n39),
1531        .sample_8n(reg_sample_8n),  .sample_40n(reg_sample_40n), .sample_n8(
            reg_sample_n8),   .sample_n40(reg_sample_n40),
1532        .sample_9n(reg_sample_9n),  .sample_41n(reg_sample_41n), .sample_n9(
            reg_sample_n9),   .sample_n41(reg_sample_n41),
1533        .sample_10n(reg_sample_10n), .sample_42n(reg_sample_42n), .sample_n10(
            reg_sample_n10), .sample_n42(reg_sample_n42),
1534        .sample_11n(reg_sample_11n), .sample_43n(reg_sample_43n), .sample_n11(
            reg_sample_n11), .sample_n43(reg_sample_n43),
1535        .sample_12n(reg_sample_12n), .sample_44n(reg_sample_44n), .sample_n12(
            reg_sample_n12), .sample_n44(reg_sample_n44),
1536        .sample_13n(reg_sample_13n), .sample_45n(reg_sample_45n), .sample_n13(
            reg_sample_n13), .sample_n45(reg_sample_n45),
1537        .sample_14n(reg_sample_14n), .sample_46n(reg_sample_46n), .sample_n14(
            reg_sample_n14), .sample_n46(reg_sample_n46),
1538        .sample_15n(reg_sample_15n), .sample_47n(reg_sample_47n), .sample_n15(
            reg_sample_n15), .sample_n47(reg_sample_n47),
1539        .sample_16n(reg_sample_16n), .sample_48n(reg_sample_48n), .sample_n16(
            reg_sample_n16), .sample_n48(reg_sample_n48),
1540        .sample_17n(reg_sample_17n), .sample_49n(reg_sample_49n), .sample_n17(
            reg_sample_n17), .sample_n49(reg_sample_n49),
1541        .sample_18n(reg_sample_18n), .sample_50n(reg_sample_50n), .sample_n18(
            reg_sample_n18), .sample_n50(reg_sample_n50),
1542        .sample_19n(reg_sample_19n), .sample_51n(reg_sample_51n), .sample_n19(
            reg_sample_n19), .sample_n51(reg_sample_n51),
1543        .sample_20n(reg_sample_20n), .sample_52n(reg_sample_52n), .sample_n20(
            reg_sample_n20), .sample_n52(reg_sample_n52),
1544        .sample_21n(reg_sample_21n), .sample_53n(reg_sample_53n), .sample_n21(
            reg_sample_n21), .sample_n53(reg_sample_n53),
1545        .sample_22n(reg_sample_22n), .sample_54n(reg_sample_54n), .sample_n22(
            reg_sample_n22), .sample_n54(reg_sample_n54),
1546        .sample_23n(reg_sample_23n), .sample_55n(reg_sample_55n), .sample_n23(
            reg_sample_n23), .sample_n55(reg_sample_n55),
1547        .sample_24n(reg_sample_24n), .sample_56n(reg_sample_56n), .sample_n24(
            reg_sample_n24), .sample_n56(reg_sample_n56),
1548        .sample_25n(reg_sample_25n), .sample_57n(reg_sample_57n), .sample_n25(
            reg_sample_n25), .sample_n57(reg_sample_n57),
1549        .sample_26n(reg_sample_26n), .sample_58n(reg_sample_58n), .sample_n26(
            reg_sample_n26), .sample_n58(reg_sample_n58),
1550        .sample_27n(reg_sample_27n), .sample_59n(reg_sample_59n), .sample_n27(
            reg_sample_n27), .sample_n59(reg_sample_n59),
1551        .sample_28n(reg_sample_28n), .sample_60n(reg_sample_60n), .sample_n28(
            reg_sample_n28), .sample_n60(reg_sample_n60),
1552        .sample_29n(reg_sample_29n), .sample_61n(reg_sample_61n), .sample_n29(
            reg_sample_n29), .sample_n61(reg_sample_n61),
```

```
1553        .sample_30n(reg_sample_30n), .sample_62n(reg_sample_62n), .sample_n30(
                reg_sample_n30), .sample_n62(reg_sample_n62),
1554        .sample_31n(reg_sample_31n), .sample_63n(reg_sample_63n), .sample_n31(
                reg_sample_n31), .sample_n63(reg_sample_n63),
1555
1556        .sample_NN(reg_sample_NN),
1557
1558        .ref1(angular_mode19_sample1),
1559        .ref2(angular_mode19_sample2),
1560        .ref3(angular_mode19_sample3),
1561        .ref4(angular_mode19_sample4),
1562        .ref5(angular_mode19_sample5)
1563    );
1564
1565    angular_refs_selector #(WIDTH, 20) angular_mode20_selector(
1566        .clock(clock),
1567        .selector(angular_mode20_samples),
1568        .sample_0n(reg_sample_0n),  .sample_32n(reg_sample_32n), .sample_n0(
                reg_sample_n0),   .sample_n32(reg_sample_n32),
1569        .sample_1n(reg_sample_1n),  .sample_33n(reg_sample_33n), .sample_n1(
                reg_sample_n1),   .sample_n33(reg_sample_n33),
1570        .sample_2n(reg_sample_2n),  .sample_34n(reg_sample_34n), .sample_n2(
                reg_sample_n2),   .sample_n34(reg_sample_n34),
1571        .sample_3n(reg_sample_3n),  .sample_35n(reg_sample_35n), .sample_n3(
                reg_sample_n3),   .sample_n35(reg_sample_n35),
1572        .sample_4n(reg_sample_4n),  .sample_36n(reg_sample_36n), .sample_n4(
                reg_sample_n4),   .sample_n36(reg_sample_n36),
1573        .sample_5n(reg_sample_5n),  .sample_37n(reg_sample_37n), .sample_n5(
                reg_sample_n5),   .sample_n37(reg_sample_n37),
1574        .sample_6n(reg_sample_6n),  .sample_38n(reg_sample_38n), .sample_n6(
                reg_sample_n6),   .sample_n38(reg_sample_n38),
1575        .sample_7n(reg_sample_7n),  .sample_39n(reg_sample_39n), .sample_n7(
                reg_sample_n7),   .sample_n39(reg_sample_n39),
1576        .sample_8n(reg_sample_8n),  .sample_40n(reg_sample_40n), .sample_n8(
                reg_sample_n8),   .sample_n40(reg_sample_n40),
1577        .sample_9n(reg_sample_9n),  .sample_41n(reg_sample_41n), .sample_n9(
                reg_sample_n9),   .sample_n41(reg_sample_n41),
1578        .sample_10n(reg_sample_10n), .sample_42n(reg_sample_42n), .sample_n10(
                reg_sample_n10), .sample_n42(reg_sample_n42),
1579        .sample_11n(reg_sample_11n), .sample_43n(reg_sample_43n), .sample_n11(
                reg_sample_n11), .sample_n43(reg_sample_n43),
1580        .sample_12n(reg_sample_12n), .sample_44n(reg_sample_44n), .sample_n12(
                reg_sample_n12), .sample_n44(reg_sample_n44),
1581        .sample_13n(reg_sample_13n), .sample_45n(reg_sample_45n), .sample_n13(
                reg_sample_n13), .sample_n45(reg_sample_n45),
1582        .sample_14n(reg_sample_14n), .sample_46n(reg_sample_46n), .sample_n14(
                reg_sample_n14), .sample_n46(reg_sample_n46),
1583        .sample_15n(reg_sample_15n), .sample_47n(reg_sample_47n), .sample_n15(
                reg_sample_n15), .sample_n47(reg_sample_n47),
1584        .sample_16n(reg_sample_16n), .sample_48n(reg_sample_48n), .sample_n16(
                reg_sample_n16), .sample_n48(reg_sample_n48),
1585        .sample_17n(reg_sample_17n), .sample_49n(reg_sample_49n), .sample_n17(
                reg_sample_n17), .sample_n49(reg_sample_n49),
1586        .sample_18n(reg_sample_18n), .sample_50n(reg_sample_50n), .sample_n18(
                reg_sample_n18), .sample_n50(reg_sample_n50),
1587        .sample_19n(reg_sample_19n), .sample_51n(reg_sample_51n), .sample_n19(
                reg_sample_n19), .sample_n51(reg_sample_n51),
```

```
1588        .sample_20n(reg_sample_20n), .sample_52n(reg_sample_52n), .sample_n20(
               reg_sample_n20), .sample_n52(reg_sample_n52),
1589        .sample_21n(reg_sample_21n), .sample_53n(reg_sample_53n), .sample_n21(
               reg_sample_n21), .sample_n53(reg_sample_n53),
1590        .sample_22n(reg_sample_22n), .sample_54n(reg_sample_54n), .sample_n22(
               reg_sample_n22), .sample_n54(reg_sample_n54),
1591        .sample_23n(reg_sample_23n), .sample_55n(reg_sample_55n), .sample_n23(
               reg_sample_n23), .sample_n55(reg_sample_n55),
1592        .sample_24n(reg_sample_24n), .sample_56n(reg_sample_56n), .sample_n24(
               reg_sample_n24), .sample_n56(reg_sample_n56),
1593        .sample_25n(reg_sample_25n), .sample_57n(reg_sample_57n), .sample_n25(
               reg_sample_n25), .sample_n57(reg_sample_n57),
1594        .sample_26n(reg_sample_26n), .sample_58n(reg_sample_58n), .sample_n26(
               reg_sample_n26), .sample_n58(reg_sample_n58),
1595        .sample_27n(reg_sample_27n), .sample_59n(reg_sample_59n), .sample_n27(
               reg_sample_n27), .sample_n59(reg_sample_n59),
1596        .sample_28n(reg_sample_28n), .sample_60n(reg_sample_60n), .sample_n28(
               reg_sample_n28), .sample_n60(reg_sample_n60),
1597        .sample_29n(reg_sample_29n), .sample_61n(reg_sample_61n), .sample_n29(
               reg_sample_n29), .sample_n61(reg_sample_n61),
1598        .sample_30n(reg_sample_30n), .sample_62n(reg_sample_62n), .sample_n30(
               reg_sample_n30), .sample_n62(reg_sample_n62),
1599        .sample_31n(reg_sample_31n), .sample_63n(reg_sample_63n), .sample_n31(
               reg_sample_n31), .sample_n63(reg_sample_n63),
1600
1601        .sample_NN(reg_sample_NN),
1602
1603        .ref1(angular_mode20_sample1),
1604        .ref2(angular_mode20_sample2),
1605        .ref3(angular_mode20_sample3),
1606        .ref4(angular_mode20_sample4),
1607        .ref5(angular_mode20_sample5)
1608    );
1609
1610    angular_refs_selector #(WIDTH, 21) angular_mode21_selector(
1611        .clock(clock),
1612        .selector(angular_mode21_samples),
1613        .sample_0n(reg_sample_0n),  .sample_32n(reg_sample_32n), .sample_n0(
               reg_sample_n0),   .sample_n32(reg_sample_n32),
1614        .sample_1n(reg_sample_1n),  .sample_33n(reg_sample_33n), .sample_n1(
               reg_sample_n1),   .sample_n33(reg_sample_n33),
1615        .sample_2n(reg_sample_2n),  .sample_34n(reg_sample_34n), .sample_n2(
               reg_sample_n2),   .sample_n34(reg_sample_n34),
1616        .sample_3n(reg_sample_3n),  .sample_35n(reg_sample_35n), .sample_n3(
               reg_sample_n3),   .sample_n35(reg_sample_n35),
1617        .sample_4n(reg_sample_4n),  .sample_36n(reg_sample_36n), .sample_n4(
               reg_sample_n4),   .sample_n36(reg_sample_n36),
1618        .sample_5n(reg_sample_5n),  .sample_37n(reg_sample_37n), .sample_n5(
               reg_sample_n5),   .sample_n37(reg_sample_n37),
1619        .sample_6n(reg_sample_6n),  .sample_38n(reg_sample_38n), .sample_n6(
               reg_sample_n6),   .sample_n38(reg_sample_n38),
1620        .sample_7n(reg_sample_7n),  .sample_39n(reg_sample_39n), .sample_n7(
               reg_sample_n7),   .sample_n39(reg_sample_n39),
1621        .sample_8n(reg_sample_8n),  .sample_40n(reg_sample_40n), .sample_n8(
               reg_sample_n8),   .sample_n40(reg_sample_n40),
1622        .sample_9n(reg_sample_9n),  .sample_41n(reg_sample_41n), .sample_n9(
               reg_sample_n9),   .sample_n41(reg_sample_n41),
```

```
1623        .sample_10n(reg_sample_10n), .sample_42n(reg_sample_42n), .sample_n10(
                reg_sample_n10), .sample_n42(reg_sample_n42),
1624        .sample_11n(reg_sample_11n), .sample_43n(reg_sample_43n), .sample_n11(
                reg_sample_n11), .sample_n43(reg_sample_n43),
1625        .sample_12n(reg_sample_12n), .sample_44n(reg_sample_44n), .sample_n12(
                reg_sample_n12), .sample_n44(reg_sample_n44),
1626        .sample_13n(reg_sample_13n), .sample_45n(reg_sample_45n), .sample_n13(
                reg_sample_n13), .sample_n45(reg_sample_n45),
1627        .sample_14n(reg_sample_14n), .sample_46n(reg_sample_46n), .sample_n14(
                reg_sample_n14), .sample_n46(reg_sample_n46),
1628        .sample_15n(reg_sample_15n), .sample_47n(reg_sample_47n), .sample_n15(
                reg_sample_n15), .sample_n47(reg_sample_n47),
1629        .sample_16n(reg_sample_16n), .sample_48n(reg_sample_48n), .sample_n16(
                reg_sample_n16), .sample_n48(reg_sample_n48),
1630        .sample_17n(reg_sample_17n), .sample_49n(reg_sample_49n), .sample_n17(
                reg_sample_n17), .sample_n49(reg_sample_n49),
1631        .sample_18n(reg_sample_18n), .sample_50n(reg_sample_50n), .sample_n18(
                reg_sample_n18), .sample_n50(reg_sample_n50),
1632        .sample_19n(reg_sample_19n), .sample_51n(reg_sample_51n), .sample_n19(
                reg_sample_n19), .sample_n51(reg_sample_n51),
1633        .sample_20n(reg_sample_20n), .sample_52n(reg_sample_52n), .sample_n20(
                reg_sample_n20), .sample_n52(reg_sample_n52),
1634        .sample_21n(reg_sample_21n), .sample_53n(reg_sample_53n), .sample_n21(
                reg_sample_n21), .sample_n53(reg_sample_n53),
1635        .sample_22n(reg_sample_22n), .sample_54n(reg_sample_54n), .sample_n22(
                reg_sample_n22), .sample_n54(reg_sample_n54),
1636        .sample_23n(reg_sample_23n), .sample_55n(reg_sample_55n), .sample_n23(
                reg_sample_n23), .sample_n55(reg_sample_n55),
1637        .sample_24n(reg_sample_24n), .sample_56n(reg_sample_56n), .sample_n24(
                reg_sample_n24), .sample_n56(reg_sample_n56),
1638        .sample_25n(reg_sample_25n), .sample_57n(reg_sample_57n), .sample_n25(
                reg_sample_n25), .sample_n57(reg_sample_n57),
1639        .sample_26n(reg_sample_26n), .sample_58n(reg_sample_58n), .sample_n26(
                reg_sample_n26), .sample_n58(reg_sample_n58),
1640        .sample_27n(reg_sample_27n), .sample_59n(reg_sample_59n), .sample_n27(
                reg_sample_n27), .sample_n59(reg_sample_n59),
1641        .sample_28n(reg_sample_28n), .sample_60n(reg_sample_60n), .sample_n28(
                reg_sample_n28), .sample_n60(reg_sample_n60),
1642        .sample_29n(reg_sample_29n), .sample_61n(reg_sample_61n), .sample_n29(
                reg_sample_n29), .sample_n61(reg_sample_n61),
1643        .sample_30n(reg_sample_30n), .sample_62n(reg_sample_62n), .sample_n30(
                reg_sample_n30), .sample_n62(reg_sample_n62),
1644        .sample_31n(reg_sample_31n), .sample_63n(reg_sample_63n), .sample_n31(
                reg_sample_n31), .sample_n63(reg_sample_n63),
1645
1646        .sample_NN(reg_sample_NN),
1647
1648        .ref1(angular_mode21_sample1),
1649        .ref2(angular_mode21_sample2),
1650        .ref3(angular_mode21_sample3),
1651        .ref4(angular_mode21_sample4),
1652        .ref5(angular_mode21_sample5)
1653    );
1654
1655    angular_refs_selector #(WIDTH, 22) angular_mode22_selector(
1656        .clock(clock),
1657        .selector(angular_mode22_samples),
```

```
1658        .sample_0n(reg_sample_0n),  .sample_32n(reg_sample_32n), .sample_n0(
               reg_sample_n0),   .sample_n32(reg_sample_n32),
1659        .sample_1n(reg_sample_1n),  .sample_33n(reg_sample_33n), .sample_n1(
               reg_sample_n1),   .sample_n33(reg_sample_n33),
1660        .sample_2n(reg_sample_2n),  .sample_34n(reg_sample_34n), .sample_n2(
               reg_sample_n2),   .sample_n34(reg_sample_n34),
1661        .sample_3n(reg_sample_3n),  .sample_35n(reg_sample_35n), .sample_n3(
               reg_sample_n3),   .sample_n35(reg_sample_n35),
1662        .sample_4n(reg_sample_4n),  .sample_36n(reg_sample_36n), .sample_n4(
               reg_sample_n4),   .sample_n36(reg_sample_n36),
1663        .sample_5n(reg_sample_5n),  .sample_37n(reg_sample_37n), .sample_n5(
               reg_sample_n5),   .sample_n37(reg_sample_n37),
1664        .sample_6n(reg_sample_6n),  .sample_38n(reg_sample_38n), .sample_n6(
               reg_sample_n6),   .sample_n38(reg_sample_n38),
1665        .sample_7n(reg_sample_7n),  .sample_39n(reg_sample_39n), .sample_n7(
               reg_sample_n7),   .sample_n39(reg_sample_n39),
1666        .sample_8n(reg_sample_8n),  .sample_40n(reg_sample_40n), .sample_n8(
               reg_sample_n8),   .sample_n40(reg_sample_n40),
1667        .sample_9n(reg_sample_9n),  .sample_41n(reg_sample_41n), .sample_n9(
               reg_sample_n9),   .sample_n41(reg_sample_n41),
1668        .sample_10n(reg_sample_10n), .sample_42n(reg_sample_42n), .sample_n10(
               reg_sample_n10), .sample_n42(reg_sample_n42),
1669        .sample_11n(reg_sample_11n), .sample_43n(reg_sample_43n), .sample_n11(
               reg_sample_n11), .sample_n43(reg_sample_n43),
1670        .sample_12n(reg_sample_12n), .sample_44n(reg_sample_44n), .sample_n12(
               reg_sample_n12), .sample_n44(reg_sample_n44),
1671        .sample_13n(reg_sample_13n), .sample_45n(reg_sample_45n), .sample_n13(
               reg_sample_n13), .sample_n45(reg_sample_n45),
1672        .sample_14n(reg_sample_14n), .sample_46n(reg_sample_46n), .sample_n14(
               reg_sample_n14), .sample_n46(reg_sample_n46),
1673        .sample_15n(reg_sample_15n), .sample_47n(reg_sample_47n), .sample_n15(
               reg_sample_n15), .sample_n47(reg_sample_n47),
1674        .sample_16n(reg_sample_16n), .sample_48n(reg_sample_48n), .sample_n16(
               reg_sample_n16), .sample_n48(reg_sample_n48),
1675        .sample_17n(reg_sample_17n), .sample_49n(reg_sample_49n), .sample_n17(
               reg_sample_n17), .sample_n49(reg_sample_n49),
1676        .sample_18n(reg_sample_18n), .sample_50n(reg_sample_50n), .sample_n18(
               reg_sample_n18), .sample_n50(reg_sample_n50),
1677        .sample_19n(reg_sample_19n), .sample_51n(reg_sample_51n), .sample_n19(
               reg_sample_n19), .sample_n51(reg_sample_n51),
1678        .sample_20n(reg_sample_20n), .sample_52n(reg_sample_52n), .sample_n20(
               reg_sample_n20), .sample_n52(reg_sample_n52),
1679        .sample_21n(reg_sample_21n), .sample_53n(reg_sample_53n), .sample_n21(
               reg_sample_n21), .sample_n53(reg_sample_n53),
1680        .sample_22n(reg_sample_22n), .sample_54n(reg_sample_54n), .sample_n22(
               reg_sample_n22), .sample_n54(reg_sample_n54),
1681        .sample_23n(reg_sample_23n), .sample_55n(reg_sample_55n), .sample_n23(
               reg_sample_n23), .sample_n55(reg_sample_n55),
1682        .sample_24n(reg_sample_24n), .sample_56n(reg_sample_56n), .sample_n24(
               reg_sample_n24), .sample_n56(reg_sample_n56),
1683        .sample_25n(reg_sample_25n), .sample_57n(reg_sample_57n), .sample_n25(
               reg_sample_n25), .sample_n57(reg_sample_n57),
1684        .sample_26n(reg_sample_26n), .sample_58n(reg_sample_58n), .sample_n26(
               reg_sample_n26), .sample_n58(reg_sample_n58),
1685        .sample_27n(reg_sample_27n), .sample_59n(reg_sample_59n), .sample_n27(
               reg_sample_n27), .sample_n59(reg_sample_n59),
1686        .sample_28n(reg_sample_28n), .sample_60n(reg_sample_60n), .sample_n28(
```

```
            reg_sample_n28), .sample_n60(reg_sample_n60),
1687        .sample_29n(reg_sample_29n), .sample_61n(reg_sample_61n), .sample_n29(
            reg_sample_n29), .sample_n61(reg_sample_n61),
1688        .sample_30n(reg_sample_30n), .sample_62n(reg_sample_62n), .sample_n30(
            reg_sample_n30), .sample_n62(reg_sample_n62),
1689        .sample_31n(reg_sample_31n), .sample_63n(reg_sample_63n), .sample_n31(
            reg_sample_n31), .sample_n63(reg_sample_n63),
1690
1691        .sample_NN(reg_sample_NN),
1692
1693        .ref1(angular_mode22_sample1),
1694        .ref2(angular_mode22_sample2),
1695        .ref3(angular_mode22_sample3),
1696        .ref4(angular_mode22_sample4),
1697        .ref5(angular_mode22_sample5)
1698    );
1699
1700    angular_refs_selector #(WIDTH, 23) angular_mode23_selector(
1701        .clock(clock),
1702        .selector(angular_mode23_samples),
1703        .sample_0n(reg_sample_0n),  .sample_32n(reg_sample_32n), .sample_n0(
            reg_sample_n0),   .sample_n32(reg_sample_n32),
1704        .sample_1n(reg_sample_1n),  .sample_33n(reg_sample_33n), .sample_n1(
            reg_sample_n1),   .sample_n33(reg_sample_n33),
1705        .sample_2n(reg_sample_2n),  .sample_34n(reg_sample_34n), .sample_n2(
            reg_sample_n2),   .sample_n34(reg_sample_n34),
1706        .sample_3n(reg_sample_3n),  .sample_35n(reg_sample_35n), .sample_n3(
            reg_sample_n3),   .sample_n35(reg_sample_n35),
1707        .sample_4n(reg_sample_4n),  .sample_36n(reg_sample_36n), .sample_n4(
            reg_sample_n4),   .sample_n36(reg_sample_n36),
1708        .sample_5n(reg_sample_5n),  .sample_37n(reg_sample_37n), .sample_n5(
            reg_sample_n5),   .sample_n37(reg_sample_n37),
1709        .sample_6n(reg_sample_6n),  .sample_38n(reg_sample_38n), .sample_n6(
            reg_sample_n6),   .sample_n38(reg_sample_n38),
1710        .sample_7n(reg_sample_7n),  .sample_39n(reg_sample_39n), .sample_n7(
            reg_sample_n7),   .sample_n39(reg_sample_n39),
1711        .sample_8n(reg_sample_8n),  .sample_40n(reg_sample_40n), .sample_n8(
            reg_sample_n8),   .sample_n40(reg_sample_n40),
1712        .sample_9n(reg_sample_9n),  .sample_41n(reg_sample_41n), .sample_n9(
            reg_sample_n9),   .sample_n41(reg_sample_n41),
1713        .sample_10n(reg_sample_10n), .sample_42n(reg_sample_42n), .sample_n10(
            reg_sample_n10), .sample_n42(reg_sample_n42),
1714        .sample_11n(reg_sample_11n), .sample_43n(reg_sample_43n), .sample_n11(
            reg_sample_n11), .sample_n43(reg_sample_n43),
1715        .sample_12n(reg_sample_12n), .sample_44n(reg_sample_44n), .sample_n12(
            reg_sample_n12), .sample_n44(reg_sample_n44),
1716        .sample_13n(reg_sample_13n), .sample_45n(reg_sample_45n), .sample_n13(
            reg_sample_n13), .sample_n45(reg_sample_n45),
1717        .sample_14n(reg_sample_14n), .sample_46n(reg_sample_46n), .sample_n14(
            reg_sample_n14), .sample_n46(reg_sample_n46),
1718        .sample_15n(reg_sample_15n), .sample_47n(reg_sample_47n), .sample_n15(
            reg_sample_n15), .sample_n47(reg_sample_n47),
1719        .sample_16n(reg_sample_16n), .sample_48n(reg_sample_48n), .sample_n16(
            reg_sample_n16), .sample_n48(reg_sample_n48),
1720        .sample_17n(reg_sample_17n), .sample_49n(reg_sample_49n), .sample_n17(
            reg_sample_n17), .sample_n49(reg_sample_n49),
1721        .sample_18n(reg_sample_18n), .sample_50n(reg_sample_50n), .sample_n18(
```

```
            reg_sample_n18), .sample_n50(reg_sample_n50),
1722        .sample_19n(reg_sample_19n), .sample_51n(reg_sample_51n), .sample_n19(
            reg_sample_n19), .sample_n51(reg_sample_n51),
1723        .sample_20n(reg_sample_20n), .sample_52n(reg_sample_52n), .sample_n20(
            reg_sample_n20), .sample_n52(reg_sample_n52),
1724        .sample_21n(reg_sample_21n), .sample_53n(reg_sample_53n), .sample_n21(
            reg_sample_n21), .sample_n53(reg_sample_n53),
1725        .sample_22n(reg_sample_22n), .sample_54n(reg_sample_54n), .sample_n22(
            reg_sample_n22), .sample_n54(reg_sample_n54),
1726        .sample_23n(reg_sample_23n), .sample_55n(reg_sample_55n), .sample_n23(
            reg_sample_n23), .sample_n55(reg_sample_n55),
1727        .sample_24n(reg_sample_24n), .sample_56n(reg_sample_56n), .sample_n24(
            reg_sample_n24), .sample_n56(reg_sample_n56),
1728        .sample_25n(reg_sample_25n), .sample_57n(reg_sample_57n), .sample_n25(
            reg_sample_n25), .sample_n57(reg_sample_n57),
1729        .sample_26n(reg_sample_26n), .sample_58n(reg_sample_58n), .sample_n26(
            reg_sample_n26), .sample_n58(reg_sample_n58),
1730        .sample_27n(reg_sample_27n), .sample_59n(reg_sample_59n), .sample_n27(
            reg_sample_n27), .sample_n59(reg_sample_n59),
1731        .sample_28n(reg_sample_28n), .sample_60n(reg_sample_60n), .sample_n28(
            reg_sample_n28), .sample_n60(reg_sample_n60),
1732        .sample_29n(reg_sample_29n), .sample_61n(reg_sample_61n), .sample_n29(
            reg_sample_n29), .sample_n61(reg_sample_n61),
1733        .sample_30n(reg_sample_30n), .sample_62n(reg_sample_62n), .sample_n30(
            reg_sample_n30), .sample_n62(reg_sample_n62),
1734        .sample_31n(reg_sample_31n), .sample_63n(reg_sample_63n), .sample_n31(
            reg_sample_n31), .sample_n63(reg_sample_n63),
1735
1736        .sample_NN(reg_sample_NN),
1737
1738        .ref1(angular_mode23_sample1),
1739        .ref2(angular_mode23_sample2),
1740        .ref3(angular_mode23_sample3),
1741        .ref4(angular_mode23_sample4),
1742        .ref5(angular_mode23_sample5)
1743    );
1744
1745    angular_refs_selector #(WIDTH, 24) angular_mode24_selector(
1746        .clock(clock),
1747        .selector(angular_mode24_samples),
1748        .sample_0n(reg_sample_0n),  .sample_32n(reg_sample_32n), .sample_n0(
            reg_sample_n0),   .sample_n32(reg_sample_n32),
1749        .sample_1n(reg_sample_1n),  .sample_33n(reg_sample_33n), .sample_n1(
            reg_sample_n1),   .sample_n33(reg_sample_n33),
1750        .sample_2n(reg_sample_2n),  .sample_34n(reg_sample_34n), .sample_n2(
            reg_sample_n2),   .sample_n34(reg_sample_n34),
1751        .sample_3n(reg_sample_3n),  .sample_35n(reg_sample_35n), .sample_n3(
            reg_sample_n3),   .sample_n35(reg_sample_n35),
1752        .sample_4n(reg_sample_4n),  .sample_36n(reg_sample_36n), .sample_n4(
            reg_sample_n4),   .sample_n36(reg_sample_n36),
1753        .sample_5n(reg_sample_5n),  .sample_37n(reg_sample_37n), .sample_n5(
            reg_sample_n5),   .sample_n37(reg_sample_n37),
1754        .sample_6n(reg_sample_6n),  .sample_38n(reg_sample_38n), .sample_n6(
            reg_sample_n6),   .sample_n38(reg_sample_n38),
1755        .sample_7n(reg_sample_7n),  .sample_39n(reg_sample_39n), .sample_n7(
            reg_sample_n7),   .sample_n39(reg_sample_n39),
1756        .sample_8n(reg_sample_8n),  .sample_40n(reg_sample_40n), .sample_n8(
```

```
            reg_sample_n8),   .sample_n40(reg_sample_n40),
1757        .sample_9n(reg_sample_9n),  .sample_41n(reg_sample_41n), .sample_n9(
            reg_sample_n9),   .sample_n41(reg_sample_n41),
1758        .sample_10n(reg_sample_10n), .sample_42n(reg_sample_42n), .sample_n10(
            reg_sample_n10), .sample_n42(reg_sample_n42),
1759        .sample_11n(reg_sample_11n), .sample_43n(reg_sample_43n), .sample_n11(
            reg_sample_n11), .sample_n43(reg_sample_n43),
1760        .sample_12n(reg_sample_12n), .sample_44n(reg_sample_44n), .sample_n12(
            reg_sample_n12), .sample_n44(reg_sample_n44),
1761        .sample_13n(reg_sample_13n), .sample_45n(reg_sample_45n), .sample_n13(
            reg_sample_n13), .sample_n45(reg_sample_n45),
1762        .sample_14n(reg_sample_14n), .sample_46n(reg_sample_46n), .sample_n14(
            reg_sample_n14), .sample_n46(reg_sample_n46),
1763        .sample_15n(reg_sample_15n), .sample_47n(reg_sample_47n), .sample_n15(
            reg_sample_n15), .sample_n47(reg_sample_n47),
1764        .sample_16n(reg_sample_16n), .sample_48n(reg_sample_48n), .sample_n16(
            reg_sample_n16), .sample_n48(reg_sample_n48),
1765        .sample_17n(reg_sample_17n), .sample_49n(reg_sample_49n), .sample_n17(
            reg_sample_n17), .sample_n49(reg_sample_n49),
1766        .sample_18n(reg_sample_18n), .sample_50n(reg_sample_50n), .sample_n18(
            reg_sample_n18), .sample_n50(reg_sample_n50),
1767        .sample_19n(reg_sample_19n), .sample_51n(reg_sample_51n), .sample_n19(
            reg_sample_n19), .sample_n51(reg_sample_n51),
1768        .sample_20n(reg_sample_20n), .sample_52n(reg_sample_52n), .sample_n20(
            reg_sample_n20), .sample_n52(reg_sample_n52),
1769        .sample_21n(reg_sample_21n), .sample_53n(reg_sample_53n), .sample_n21(
            reg_sample_n21), .sample_n53(reg_sample_n53),
1770        .sample_22n(reg_sample_22n), .sample_54n(reg_sample_54n), .sample_n22(
            reg_sample_n22), .sample_n54(reg_sample_n54),
1771        .sample_23n(reg_sample_23n), .sample_55n(reg_sample_55n), .sample_n23(
            reg_sample_n23), .sample_n55(reg_sample_n55),
1772        .sample_24n(reg_sample_24n), .sample_56n(reg_sample_56n), .sample_n24(
            reg_sample_n24), .sample_n56(reg_sample_n56),
1773        .sample_25n(reg_sample_25n), .sample_57n(reg_sample_57n), .sample_n25(
            reg_sample_n25), .sample_n57(reg_sample_n57),
1774        .sample_26n(reg_sample_26n), .sample_58n(reg_sample_58n), .sample_n26(
            reg_sample_n26), .sample_n58(reg_sample_n58),
1775        .sample_27n(reg_sample_27n), .sample_59n(reg_sample_59n), .sample_n27(
            reg_sample_n27), .sample_n59(reg_sample_n59),
1776        .sample_28n(reg_sample_28n), .sample_60n(reg_sample_60n), .sample_n28(
            reg_sample_n28), .sample_n60(reg_sample_n60),
1777        .sample_29n(reg_sample_29n), .sample_61n(reg_sample_61n), .sample_n29(
            reg_sample_n29), .sample_n61(reg_sample_n61),
1778        .sample_30n(reg_sample_30n), .sample_62n(reg_sample_62n), .sample_n30(
            reg_sample_n30), .sample_n62(reg_sample_n62),
1779        .sample_31n(reg_sample_31n), .sample_63n(reg_sample_63n), .sample_n31(
            reg_sample_n31), .sample_n63(reg_sample_n63),
1780
1781        .sample_NN(reg_sample_NN),
1782
1783        .ref1(angular_mode24_sample1),
1784        .ref2(angular_mode24_sample2),
1785        .ref3(angular_mode24_sample3),
1786        .ref4(angular_mode24_sample4),
1787        .ref5(angular_mode24_sample5)
1788    );
1789
```

```
    angular_refs_selector #(WIDTH, 25) angular_mode25_selector(
       .clock(clock),
       .selector(angular_mode25_samples),
       .sample_0n(reg_sample_0n),  .sample_32n(reg_sample_32n), .sample_n0(
          reg_sample_n0),   .sample_n32(reg_sample_n32),
       .sample_1n(reg_sample_1n),  .sample_33n(reg_sample_33n), .sample_n1(
          reg_sample_n1),   .sample_n33(reg_sample_n33),
       .sample_2n(reg_sample_2n),  .sample_34n(reg_sample_34n), .sample_n2(
          reg_sample_n2),   .sample_n34(reg_sample_n34),
       .sample_3n(reg_sample_3n),  .sample_35n(reg_sample_35n), .sample_n3(
          reg_sample_n3),   .sample_n35(reg_sample_n35),
       .sample_4n(reg_sample_4n),  .sample_36n(reg_sample_36n), .sample_n4(
          reg_sample_n4),   .sample_n36(reg_sample_n36),
       .sample_5n(reg_sample_5n),  .sample_37n(reg_sample_37n), .sample_n5(
          reg_sample_n5),   .sample_n37(reg_sample_n37),
       .sample_6n(reg_sample_6n),  .sample_38n(reg_sample_38n), .sample_n6(
          reg_sample_n6),   .sample_n38(reg_sample_n38),
       .sample_7n(reg_sample_7n),  .sample_39n(reg_sample_39n), .sample_n7(
          reg_sample_n7),   .sample_n39(reg_sample_n39),
       .sample_8n(reg_sample_8n),  .sample_40n(reg_sample_40n), .sample_n8(
          reg_sample_n8),   .sample_n40(reg_sample_n40),
       .sample_9n(reg_sample_9n),  .sample_41n(reg_sample_41n), .sample_n9(
          reg_sample_n9),   .sample_n41(reg_sample_n41),
       .sample_10n(reg_sample_10n), .sample_42n(reg_sample_42n), .sample_n10(
          reg_sample_n10), .sample_n42(reg_sample_n42),
       .sample_11n(reg_sample_11n), .sample_43n(reg_sample_43n), .sample_n11(
          reg_sample_n11), .sample_n43(reg_sample_n43),
       .sample_12n(reg_sample_12n), .sample_44n(reg_sample_44n), .sample_n12(
          reg_sample_n12), .sample_n44(reg_sample_n44),
       .sample_13n(reg_sample_13n), .sample_45n(reg_sample_45n), .sample_n13(
          reg_sample_n13), .sample_n45(reg_sample_n45),
       .sample_14n(reg_sample_14n), .sample_46n(reg_sample_46n), .sample_n14(
          reg_sample_n14), .sample_n46(reg_sample_n46),
       .sample_15n(reg_sample_15n), .sample_47n(reg_sample_47n), .sample_n15(
          reg_sample_n15), .sample_n47(reg_sample_n47),
       .sample_16n(reg_sample_16n), .sample_48n(reg_sample_48n), .sample_n16(
          reg_sample_n16), .sample_n48(reg_sample_n48),
       .sample_17n(reg_sample_17n), .sample_49n(reg_sample_49n), .sample_n17(
          reg_sample_n17), .sample_n49(reg_sample_n49),
       .sample_18n(reg_sample_18n), .sample_50n(reg_sample_50n), .sample_n18(
          reg_sample_n18), .sample_n50(reg_sample_n50),
       .sample_19n(reg_sample_19n), .sample_51n(reg_sample_51n), .sample_n19(
          reg_sample_n19), .sample_n51(reg_sample_n51),
       .sample_20n(reg_sample_20n), .sample_52n(reg_sample_52n), .sample_n20(
          reg_sample_n20), .sample_n52(reg_sample_n52),
       .sample_21n(reg_sample_21n), .sample_53n(reg_sample_53n), .sample_n21(
          reg_sample_n21), .sample_n53(reg_sample_n53),
       .sample_22n(reg_sample_22n), .sample_54n(reg_sample_54n), .sample_n22(
          reg_sample_n22), .sample_n54(reg_sample_n54),
       .sample_23n(reg_sample_23n), .sample_55n(reg_sample_55n), .sample_n23(
          reg_sample_n23), .sample_n55(reg_sample_n55),
       .sample_24n(reg_sample_24n), .sample_56n(reg_sample_56n), .sample_n24(
          reg_sample_n24), .sample_n56(reg_sample_n56),
       .sample_25n(reg_sample_25n), .sample_57n(reg_sample_57n), .sample_n25(
          reg_sample_n25), .sample_n57(reg_sample_n57),
       .sample_26n(reg_sample_26n), .sample_58n(reg_sample_58n), .sample_n26(
          reg_sample_n26), .sample_n58(reg_sample_n58),
```

```
1820        .sample_27n(reg_sample_27n), .sample_59n(reg_sample_59n), .sample_n27(
                reg_sample_n27), .sample_n59(reg_sample_n59),
1821        .sample_28n(reg_sample_28n), .sample_60n(reg_sample_60n), .sample_n28(
                reg_sample_n28), .sample_n60(reg_sample_n60),
1822        .sample_29n(reg_sample_29n), .sample_61n(reg_sample_61n), .sample_n29(
                reg_sample_n29), .sample_n61(reg_sample_n61),
1823        .sample_30n(reg_sample_30n), .sample_62n(reg_sample_62n), .sample_n30(
                reg_sample_n30), .sample_n62(reg_sample_n62),
1824        .sample_31n(reg_sample_31n), .sample_63n(reg_sample_63n), .sample_n31(
                reg_sample_n31), .sample_n63(reg_sample_n63),
1825
1826        .sample_NN(reg_sample_NN),
1827
1828        .ref1(angular_mode25_sample1),
1829        .ref2(angular_mode25_sample2),
1830        .ref3(angular_mode25_sample3),
1831        .ref4(angular_mode25_sample4),
1832        .ref5(angular_mode25_sample5)
1833    );
1834
1835    angular_directly_refs_selector #(WIDTH, 26) angular_mode26_selector(
1836        .clock(clock),
1837        .index(filter_samples_mode26),
1838        .selector(angular_mode26_samples),
1839        .sample_0n(reg_sample_0n),  .sample_32n(reg_sample_32n), .sample_n0(
                reg_sample_n0),   .sample_n32(reg_sample_n32),
1840        .sample_1n(reg_sample_1n),  .sample_33n(reg_sample_33n), .sample_n1(
                reg_sample_n1),   .sample_n33(reg_sample_n33),
1841        .sample_2n(reg_sample_2n),  .sample_34n(reg_sample_34n), .sample_n2(
                reg_sample_n2),   .sample_n34(reg_sample_n34),
1842        .sample_3n(reg_sample_3n),  .sample_35n(reg_sample_35n), .sample_n3(
                reg_sample_n3),   .sample_n35(reg_sample_n35),
1843        .sample_4n(reg_sample_4n),  .sample_36n(reg_sample_36n), .sample_n4(
                reg_sample_n4),   .sample_n36(reg_sample_n36),
1844        .sample_5n(reg_sample_5n),  .sample_37n(reg_sample_37n), .sample_n5(
                reg_sample_n5),   .sample_n37(reg_sample_n37),
1845        .sample_6n(reg_sample_6n),  .sample_38n(reg_sample_38n), .sample_n6(
                reg_sample_n6),   .sample_n38(reg_sample_n38),
1846        .sample_7n(reg_sample_7n),  .sample_39n(reg_sample_39n), .sample_n7(
                reg_sample_n7),   .sample_n39(reg_sample_n39),
1847        .sample_8n(reg_sample_8n),  .sample_40n(reg_sample_40n), .sample_n8(
                reg_sample_n8),   .sample_n40(reg_sample_n40),
1848        .sample_9n(reg_sample_9n),  .sample_41n(reg_sample_41n), .sample_n9(
                reg_sample_n9),   .sample_n41(reg_sample_n41),
1849        .sample_10n(reg_sample_10n), .sample_42n(reg_sample_42n), .sample_n10(
                reg_sample_n10), .sample_n42(reg_sample_n42),
1850        .sample_11n(reg_sample_11n), .sample_43n(reg_sample_43n), .sample_n11(
                reg_sample_n11), .sample_n43(reg_sample_n43),
1851        .sample_12n(reg_sample_12n), .sample_44n(reg_sample_44n), .sample_n12(
                reg_sample_n12), .sample_n44(reg_sample_n44),
1852        .sample_13n(reg_sample_13n), .sample_45n(reg_sample_45n), .sample_n13(
                reg_sample_n13), .sample_n45(reg_sample_n45),
1853        .sample_14n(reg_sample_14n), .sample_46n(reg_sample_46n), .sample_n14(
                reg_sample_n14), .sample_n46(reg_sample_n46),
1854        .sample_15n(reg_sample_15n), .sample_47n(reg_sample_47n), .sample_n15(
                reg_sample_n15), .sample_n47(reg_sample_n47),
1855        .sample_16n(reg_sample_16n), .sample_48n(reg_sample_48n), .sample_n16(
```

```
            reg_sample_n16), .sample_n48(reg_sample_n48),
1856        .sample_17n(reg_sample_17n), .sample_49n(reg_sample_49n), .sample_n17(
            reg_sample_n17), .sample_n49(reg_sample_n49),
1857        .sample_18n(reg_sample_18n), .sample_50n(reg_sample_50n), .sample_n18(
            reg_sample_n18), .sample_n50(reg_sample_n50),
1858        .sample_19n(reg_sample_19n), .sample_51n(reg_sample_51n), .sample_n19(
            reg_sample_n19), .sample_n51(reg_sample_n51),
1859        .sample_20n(reg_sample_20n), .sample_52n(reg_sample_52n), .sample_n20(
            reg_sample_n20), .sample_n52(reg_sample_n52),
1860        .sample_21n(reg_sample_21n), .sample_53n(reg_sample_53n), .sample_n21(
            reg_sample_n21), .sample_n53(reg_sample_n53),
1861        .sample_22n(reg_sample_22n), .sample_54n(reg_sample_54n), .sample_n22(
            reg_sample_n22), .sample_n54(reg_sample_n54),
1862        .sample_23n(reg_sample_23n), .sample_55n(reg_sample_55n), .sample_n23(
            reg_sample_n23), .sample_n55(reg_sample_n55),
1863        .sample_24n(reg_sample_24n), .sample_56n(reg_sample_56n), .sample_n24(
            reg_sample_n24), .sample_n56(reg_sample_n56),
1864        .sample_25n(reg_sample_25n), .sample_57n(reg_sample_57n), .sample_n25(
            reg_sample_n25), .sample_n57(reg_sample_n57),
1865        .sample_26n(reg_sample_26n), .sample_58n(reg_sample_58n), .sample_n26(
            reg_sample_n26), .sample_n58(reg_sample_n58),
1866        .sample_27n(reg_sample_27n), .sample_59n(reg_sample_59n), .sample_n27(
            reg_sample_n27), .sample_n59(reg_sample_n59),
1867        .sample_28n(reg_sample_28n), .sample_60n(reg_sample_60n), .sample_n28(
            reg_sample_n28), .sample_n60(reg_sample_n60),
1868        .sample_29n(reg_sample_29n), .sample_61n(reg_sample_61n), .sample_n29(
            reg_sample_n29), .sample_n61(reg_sample_n61),
1869        .sample_30n(reg_sample_30n), .sample_62n(reg_sample_62n), .sample_n30(
            reg_sample_n30), .sample_n62(reg_sample_n62),
1870        .sample_31n(reg_sample_31n), .sample_63n(reg_sample_63n), .sample_n31(
            reg_sample_n31), .sample_n63(reg_sample_n63),
1871
1872        .sample_NN(reg_sample_NN),
1873
1874        .ref0(angular_mode26_sample0),
1875        .ref1(angular_mode26_sample1),
1876        .ref2(angular_mode26_sample2),
1877        .ref3(angular_mode26_sample3),
1878        .ref4(angular_mode26_sample4),
1879        .ref5(angular_mode26_sample5)
1880    );
1881
1882    angular_refs_selector #(WIDTH, 27) angular_mode27_selector(
1883        .clock(clock),
1884        .selector(angular_mode27_samples),
1885        .sample_0n(reg_sample_0n),  .sample_32n(reg_sample_32n), .sample_n0(
            reg_sample_n0),   .sample_n32(reg_sample_n32),
1886        .sample_1n(reg_sample_1n),  .sample_33n(reg_sample_33n), .sample_n1(
            reg_sample_n1),   .sample_n33(reg_sample_n33),
1887        .sample_2n(reg_sample_2n),  .sample_34n(reg_sample_34n), .sample_n2(
            reg_sample_n2),   .sample_n34(reg_sample_n34),
1888        .sample_3n(reg_sample_3n),  .sample_35n(reg_sample_35n), .sample_n3(
            reg_sample_n3),   .sample_n35(reg_sample_n35),
1889        .sample_4n(reg_sample_4n),  .sample_36n(reg_sample_36n), .sample_n4(
            reg_sample_n4),   .sample_n36(reg_sample_n36),
1890        .sample_5n(reg_sample_5n),  .sample_37n(reg_sample_37n), .sample_n5(
            reg_sample_n5),   .sample_n37(reg_sample_n37),
```

```
1891        .sample_6n(reg_sample_6n),  .sample_38n(reg_sample_38n), .sample_n6(
                reg_sample_n6),   .sample_n38(reg_sample_n38),
1892        .sample_7n(reg_sample_7n),  .sample_39n(reg_sample_39n), .sample_n7(
                reg_sample_n7),   .sample_n39(reg_sample_n39),
1893        .sample_8n(reg_sample_8n),  .sample_40n(reg_sample_40n), .sample_n8(
                reg_sample_n8),   .sample_n40(reg_sample_n40),
1894        .sample_9n(reg_sample_9n),  .sample_41n(reg_sample_41n), .sample_n9(
                reg_sample_n9),   .sample_n41(reg_sample_n41),
1895        .sample_10n(reg_sample_10n), .sample_42n(reg_sample_42n), .sample_n10(
                reg_sample_n10), .sample_n42(reg_sample_n42),
1896        .sample_11n(reg_sample_11n), .sample_43n(reg_sample_43n), .sample_n11(
                reg_sample_n11), .sample_n43(reg_sample_n43),
1897        .sample_12n(reg_sample_12n), .sample_44n(reg_sample_44n), .sample_n12(
                reg_sample_n12), .sample_n44(reg_sample_n44),
1898        .sample_13n(reg_sample_13n), .sample_45n(reg_sample_45n), .sample_n13(
                reg_sample_n13), .sample_n45(reg_sample_n45),
1899        .sample_14n(reg_sample_14n), .sample_46n(reg_sample_46n), .sample_n14(
                reg_sample_n14), .sample_n46(reg_sample_n46),
1900        .sample_15n(reg_sample_15n), .sample_47n(reg_sample_47n), .sample_n15(
                reg_sample_n15), .sample_n47(reg_sample_n47),
1901        .sample_16n(reg_sample_16n), .sample_48n(reg_sample_48n), .sample_n16(
                reg_sample_n16), .sample_n48(reg_sample_n48),
1902        .sample_17n(reg_sample_17n), .sample_49n(reg_sample_49n), .sample_n17(
                reg_sample_n17), .sample_n49(reg_sample_n49),
1903        .sample_18n(reg_sample_18n), .sample_50n(reg_sample_50n), .sample_n18(
                reg_sample_n18), .sample_n50(reg_sample_n50),
1904        .sample_19n(reg_sample_19n), .sample_51n(reg_sample_51n), .sample_n19(
                reg_sample_n19), .sample_n51(reg_sample_n51),
1905        .sample_20n(reg_sample_20n), .sample_52n(reg_sample_52n), .sample_n20(
                reg_sample_n20), .sample_n52(reg_sample_n52),
1906        .sample_21n(reg_sample_21n), .sample_53n(reg_sample_53n), .sample_n21(
                reg_sample_n21), .sample_n53(reg_sample_n53),
1907        .sample_22n(reg_sample_22n), .sample_54n(reg_sample_54n), .sample_n22(
                reg_sample_n22), .sample_n54(reg_sample_n54),
1908        .sample_23n(reg_sample_23n), .sample_55n(reg_sample_55n), .sample_n23(
                reg_sample_n23), .sample_n55(reg_sample_n55),
1909        .sample_24n(reg_sample_24n), .sample_56n(reg_sample_56n), .sample_n24(
                reg_sample_n24), .sample_n56(reg_sample_n56),
1910        .sample_25n(reg_sample_25n), .sample_57n(reg_sample_57n), .sample_n25(
                reg_sample_n25), .sample_n57(reg_sample_n57),
1911        .sample_26n(reg_sample_26n), .sample_58n(reg_sample_58n), .sample_n26(
                reg_sample_n26), .sample_n58(reg_sample_n58),
1912        .sample_27n(reg_sample_27n), .sample_59n(reg_sample_59n), .sample_n27(
                reg_sample_n27), .sample_n59(reg_sample_n59),
1913        .sample_28n(reg_sample_28n), .sample_60n(reg_sample_60n), .sample_n28(
                reg_sample_n28), .sample_n60(reg_sample_n60),
1914        .sample_29n(reg_sample_29n), .sample_61n(reg_sample_61n), .sample_n29(
                reg_sample_n29), .sample_n61(reg_sample_n61),
1915        .sample_30n(reg_sample_30n), .sample_62n(reg_sample_62n), .sample_n30(
                reg_sample_n30), .sample_n62(reg_sample_n62),
1916        .sample_31n(reg_sample_31n), .sample_63n(reg_sample_63n), .sample_n31(
                reg_sample_n31), .sample_n63(reg_sample_n63),
1917
1918        .sample_NN(reg_sample_NN),
1919
1920        .ref1(angular_mode27_sample1),
1921        .ref2(angular_mode27_sample2),
```

```
1922        .ref3(angular_mode27_sample3),
1923        .ref4(angular_mode27_sample4),
1924        .ref5(angular_mode27_sample5)
1925    );
1926
1927    angular_refs_selector #(WIDTH, 28) angular_mode28_selector(
1928        .clock(clock),
1929        .selector(angular_mode28_samples),
1930        .sample_0n(reg_sample_0n),  .sample_32n(reg_sample_32n), .sample_n0(
               reg_sample_n0),   .sample_n32(reg_sample_n32),
1931        .sample_1n(reg_sample_1n),  .sample_33n(reg_sample_33n), .sample_n1(
               reg_sample_n1),   .sample_n33(reg_sample_n33),
1932        .sample_2n(reg_sample_2n),  .sample_34n(reg_sample_34n), .sample_n2(
               reg_sample_n2),   .sample_n34(reg_sample_n34),
1933        .sample_3n(reg_sample_3n),  .sample_35n(reg_sample_35n), .sample_n3(
               reg_sample_n3),   .sample_n35(reg_sample_n35),
1934        .sample_4n(reg_sample_4n),  .sample_36n(reg_sample_36n), .sample_n4(
               reg_sample_n4),   .sample_n36(reg_sample_n36),
1935        .sample_5n(reg_sample_5n),  .sample_37n(reg_sample_37n), .sample_n5(
               reg_sample_n5),   .sample_n37(reg_sample_n37),
1936        .sample_6n(reg_sample_6n),  .sample_38n(reg_sample_38n), .sample_n6(
               reg_sample_n6),   .sample_n38(reg_sample_n38),
1937        .sample_7n(reg_sample_7n),  .sample_39n(reg_sample_39n), .sample_n7(
               reg_sample_n7),   .sample_n39(reg_sample_n39),
1938        .sample_8n(reg_sample_8n),  .sample_40n(reg_sample_40n), .sample_n8(
               reg_sample_n8),   .sample_n40(reg_sample_n40),
1939        .sample_9n(reg_sample_9n),  .sample_41n(reg_sample_41n), .sample_n9(
               reg_sample_n9),   .sample_n41(reg_sample_n41),
1940        .sample_10n(reg_sample_10n), .sample_42n(reg_sample_42n), .sample_n10(
               reg_sample_n10), .sample_n42(reg_sample_n42),
1941        .sample_11n(reg_sample_11n), .sample_43n(reg_sample_43n), .sample_n11(
               reg_sample_n11), .sample_n43(reg_sample_n43),
1942        .sample_12n(reg_sample_12n), .sample_44n(reg_sample_44n), .sample_n12(
               reg_sample_n12), .sample_n44(reg_sample_n44),
1943        .sample_13n(reg_sample_13n), .sample_45n(reg_sample_45n), .sample_n13(
               reg_sample_n13), .sample_n45(reg_sample_n45),
1944        .sample_14n(reg_sample_14n), .sample_46n(reg_sample_46n), .sample_n14(
               reg_sample_n14), .sample_n46(reg_sample_n46),
1945        .sample_15n(reg_sample_15n), .sample_47n(reg_sample_47n), .sample_n15(
               reg_sample_n15), .sample_n47(reg_sample_n47),
1946        .sample_16n(reg_sample_16n), .sample_48n(reg_sample_48n), .sample_n16(
               reg_sample_n16), .sample_n48(reg_sample_n48),
1947        .sample_17n(reg_sample_17n), .sample_49n(reg_sample_49n), .sample_n17(
               reg_sample_n17), .sample_n49(reg_sample_n49),
1948        .sample_18n(reg_sample_18n), .sample_50n(reg_sample_50n), .sample_n18(
               reg_sample_n18), .sample_n50(reg_sample_n50),
1949        .sample_19n(reg_sample_19n), .sample_51n(reg_sample_51n), .sample_n19(
               reg_sample_n19), .sample_n51(reg_sample_n51),
1950        .sample_20n(reg_sample_20n), .sample_52n(reg_sample_52n), .sample_n20(
               reg_sample_n20), .sample_n52(reg_sample_n52),
1951        .sample_21n(reg_sample_21n), .sample_53n(reg_sample_53n), .sample_n21(
               reg_sample_n21), .sample_n53(reg_sample_n53),
1952        .sample_22n(reg_sample_22n), .sample_54n(reg_sample_54n), .sample_n22(
               reg_sample_n22), .sample_n54(reg_sample_n54),
1953        .sample_23n(reg_sample_23n), .sample_55n(reg_sample_55n), .sample_n23(
               reg_sample_n23), .sample_n55(reg_sample_n55),
1954        .sample_24n(reg_sample_24n), .sample_56n(reg_sample_56n), .sample_n24(
```

```
            reg_sample_n24), .sample_n56(reg_sample_n56),
1955        .sample_25n(reg_sample_25n), .sample_57n(reg_sample_57n), .sample_n25(
            reg_sample_n25), .sample_n57(reg_sample_n57),
1956        .sample_26n(reg_sample_26n), .sample_58n(reg_sample_58n), .sample_n26(
            reg_sample_n26), .sample_n58(reg_sample_n58),
1957        .sample_27n(reg_sample_27n), .sample_59n(reg_sample_59n), .sample_n27(
            reg_sample_n27), .sample_n59(reg_sample_n59),
1958        .sample_28n(reg_sample_28n), .sample_60n(reg_sample_60n), .sample_n28(
            reg_sample_n28), .sample_n60(reg_sample_n60),
1959        .sample_29n(reg_sample_29n), .sample_61n(reg_sample_61n), .sample_n29(
            reg_sample_n29), .sample_n61(reg_sample_n61),
1960        .sample_30n(reg_sample_30n), .sample_62n(reg_sample_62n), .sample_n30(
            reg_sample_n30), .sample_n62(reg_sample_n62),
1961        .sample_31n(reg_sample_31n), .sample_63n(reg_sample_63n), .sample_n31(
            reg_sample_n31), .sample_n63(reg_sample_n63),
1962
1963        .sample_NN(reg_sample_NN),
1964
1965        .ref1(angular_mode28_sample1),
1966        .ref2(angular_mode28_sample2),
1967        .ref3(angular_mode28_sample3),
1968        .ref4(angular_mode28_sample4),
1969        .ref5(angular_mode28_sample5)
1970    );
1971
1972    angular_refs_selector #(WIDTH, 29) angular_mode29_selector(
1973        .clock(clock),
1974        .selector(angular_mode29_samples),
1975        .sample_0n(reg_sample_0n),  .sample_32n(reg_sample_32n), .sample_n0(
            reg_sample_n0),   .sample_n32(reg_sample_n32),
1976        .sample_1n(reg_sample_1n),  .sample_33n(reg_sample_33n), .sample_n1(
            reg_sample_n1),   .sample_n33(reg_sample_n33),
1977        .sample_2n(reg_sample_2n),  .sample_34n(reg_sample_34n), .sample_n2(
            reg_sample_n2),   .sample_n34(reg_sample_n34),
1978        .sample_3n(reg_sample_3n),  .sample_35n(reg_sample_35n), .sample_n3(
            reg_sample_n3),   .sample_n35(reg_sample_n35),
1979        .sample_4n(reg_sample_4n),  .sample_36n(reg_sample_36n), .sample_n4(
            reg_sample_n4),   .sample_n36(reg_sample_n36),
1980        .sample_5n(reg_sample_5n),  .sample_37n(reg_sample_37n), .sample_n5(
            reg_sample_n5),   .sample_n37(reg_sample_n37),
1981        .sample_6n(reg_sample_6n),  .sample_38n(reg_sample_38n), .sample_n6(
            reg_sample_n6),   .sample_n38(reg_sample_n38),
1982        .sample_7n(reg_sample_7n),  .sample_39n(reg_sample_39n), .sample_n7(
            reg_sample_n7),   .sample_n39(reg_sample_n39),
1983        .sample_8n(reg_sample_8n),  .sample_40n(reg_sample_40n), .sample_n8(
            reg_sample_n8),   .sample_n40(reg_sample_n40),
1984        .sample_9n(reg_sample_9n),  .sample_41n(reg_sample_41n), .sample_n9(
            reg_sample_n9),   .sample_n41(reg_sample_n41),
1985        .sample_10n(reg_sample_10n), .sample_42n(reg_sample_42n), .sample_n10(
            reg_sample_n10), .sample_n42(reg_sample_n42),
1986        .sample_11n(reg_sample_11n), .sample_43n(reg_sample_43n), .sample_n11(
            reg_sample_n11), .sample_n43(reg_sample_n43),
1987        .sample_12n(reg_sample_12n), .sample_44n(reg_sample_44n), .sample_n12(
            reg_sample_n12), .sample_n44(reg_sample_n44),
1988        .sample_13n(reg_sample_13n), .sample_45n(reg_sample_45n), .sample_n13(
            reg_sample_n13), .sample_n45(reg_sample_n45),
1989        .sample_14n(reg_sample_14n), .sample_46n(reg_sample_46n), .sample_n14(
```

```
            reg_sample_n14), .sample_n46(reg_sample_n46),
1990        .sample_15n(reg_sample_15n), .sample_47n(reg_sample_47n), .sample_n15(
            reg_sample_n15), .sample_n47(reg_sample_n47),
1991        .sample_16n(reg_sample_16n), .sample_48n(reg_sample_48n), .sample_n16(
            reg_sample_n16), .sample_n48(reg_sample_n48),
1992        .sample_17n(reg_sample_17n), .sample_49n(reg_sample_49n), .sample_n17(
            reg_sample_n17), .sample_n49(reg_sample_n49),
1993        .sample_18n(reg_sample_18n), .sample_50n(reg_sample_50n), .sample_n18(
            reg_sample_n18), .sample_n50(reg_sample_n50),
1994        .sample_19n(reg_sample_19n), .sample_51n(reg_sample_51n), .sample_n19(
            reg_sample_n19), .sample_n51(reg_sample_n51),
1995        .sample_20n(reg_sample_20n), .sample_52n(reg_sample_52n), .sample_n20(
            reg_sample_n20), .sample_n52(reg_sample_n52),
1996        .sample_21n(reg_sample_21n), .sample_53n(reg_sample_53n), .sample_n21(
            reg_sample_n21), .sample_n53(reg_sample_n53),
1997        .sample_22n(reg_sample_22n), .sample_54n(reg_sample_54n), .sample_n22(
            reg_sample_n22), .sample_n54(reg_sample_n54),
1998        .sample_23n(reg_sample_23n), .sample_55n(reg_sample_55n), .sample_n23(
            reg_sample_n23), .sample_n55(reg_sample_n55),
1999        .sample_24n(reg_sample_24n), .sample_56n(reg_sample_56n), .sample_n24(
            reg_sample_n24), .sample_n56(reg_sample_n56),
2000        .sample_25n(reg_sample_25n), .sample_57n(reg_sample_57n), .sample_n25(
            reg_sample_n25), .sample_n57(reg_sample_n57),
2001        .sample_26n(reg_sample_26n), .sample_58n(reg_sample_58n), .sample_n26(
            reg_sample_n26), .sample_n58(reg_sample_n58),
2002        .sample_27n(reg_sample_27n), .sample_59n(reg_sample_59n), .sample_n27(
            reg_sample_n27), .sample_n59(reg_sample_n59),
2003        .sample_28n(reg_sample_28n), .sample_60n(reg_sample_60n), .sample_n28(
            reg_sample_n28), .sample_n60(reg_sample_n60),
2004        .sample_29n(reg_sample_29n), .sample_61n(reg_sample_61n), .sample_n29(
            reg_sample_n29), .sample_n61(reg_sample_n61),
2005        .sample_30n(reg_sample_30n), .sample_62n(reg_sample_62n), .sample_n30(
            reg_sample_n30), .sample_n62(reg_sample_n62),
2006        .sample_31n(reg_sample_31n), .sample_63n(reg_sample_63n), .sample_n31(
            reg_sample_n31), .sample_n63(reg_sample_n63),
2007
2008        .sample_NN(reg_sample_NN),
2009
2010        .ref1(angular_mode29_sample1),
2011        .ref2(angular_mode29_sample2),
2012        .ref3(angular_mode29_sample3),
2013        .ref4(angular_mode29_sample4),
2014        .ref5(angular_mode29_sample5)
2015    );
2016
2017    angular_refs_selector #(WIDTH, 30) angular_mode30_selector(
2018        .clock(clock),
2019        .selector(angular_mode30_samples),
2020        .sample_0n(reg_sample_0n),  .sample_32n(reg_sample_32n), .sample_n0(
            reg_sample_n0),   .sample_n32(reg_sample_n32),
2021        .sample_1n(reg_sample_1n),  .sample_33n(reg_sample_33n), .sample_n1(
            reg_sample_n1),   .sample_n33(reg_sample_n33),
2022        .sample_2n(reg_sample_2n),  .sample_34n(reg_sample_34n), .sample_n2(
            reg_sample_n2),   .sample_n34(reg_sample_n34),
2023        .sample_3n(reg_sample_3n),  .sample_35n(reg_sample_35n), .sample_n3(
            reg_sample_n3),   .sample_n35(reg_sample_n35),
2024        .sample_4n(reg_sample_4n),  .sample_36n(reg_sample_36n), .sample_n4(
```

```
            reg_sample_n4),   .sample_n36(reg_sample_n36),
2025        .sample_5n(reg_sample_5n),  .sample_37n(reg_sample_37n), .sample_n5(
            reg_sample_n5),   .sample_n37(reg_sample_n37),
2026        .sample_6n(reg_sample_6n),  .sample_38n(reg_sample_38n), .sample_n6(
            reg_sample_n6),   .sample_n38(reg_sample_n38),
2027        .sample_7n(reg_sample_7n),  .sample_39n(reg_sample_39n), .sample_n7(
            reg_sample_n7),   .sample_n39(reg_sample_n39),
2028        .sample_8n(reg_sample_8n),  .sample_40n(reg_sample_40n), .sample_n8(
            reg_sample_n8),   .sample_n40(reg_sample_n40),
2029        .sample_9n(reg_sample_9n),  .sample_41n(reg_sample_41n), .sample_n9(
            reg_sample_n9),   .sample_n41(reg_sample_n41),
2030        .sample_10n(reg_sample_10n), .sample_42n(reg_sample_42n), .sample_n10(
            reg_sample_n10), .sample_n42(reg_sample_n42),
2031        .sample_11n(reg_sample_11n), .sample_43n(reg_sample_43n), .sample_n11(
            reg_sample_n11), .sample_n43(reg_sample_n43),
2032        .sample_12n(reg_sample_12n), .sample_44n(reg_sample_44n), .sample_n12(
            reg_sample_n12), .sample_n44(reg_sample_n44),
2033        .sample_13n(reg_sample_13n), .sample_45n(reg_sample_45n), .sample_n13(
            reg_sample_n13), .sample_n45(reg_sample_n45),
2034        .sample_14n(reg_sample_14n), .sample_46n(reg_sample_46n), .sample_n14(
            reg_sample_n14), .sample_n46(reg_sample_n46),
2035        .sample_15n(reg_sample_15n), .sample_47n(reg_sample_47n), .sample_n15(
            reg_sample_n15), .sample_n47(reg_sample_n47),
2036        .sample_16n(reg_sample_16n), .sample_48n(reg_sample_48n), .sample_n16(
            reg_sample_n16), .sample_n48(reg_sample_n48),
2037        .sample_17n(reg_sample_17n), .sample_49n(reg_sample_49n), .sample_n17(
            reg_sample_n17), .sample_n49(reg_sample_n49),
2038        .sample_18n(reg_sample_18n), .sample_50n(reg_sample_50n), .sample_n18(
            reg_sample_n18), .sample_n50(reg_sample_n50),
2039        .sample_19n(reg_sample_19n), .sample_51n(reg_sample_51n), .sample_n19(
            reg_sample_n19), .sample_n51(reg_sample_n51),
2040        .sample_20n(reg_sample_20n), .sample_52n(reg_sample_52n), .sample_n20(
            reg_sample_n20), .sample_n52(reg_sample_n52),
2041        .sample_21n(reg_sample_21n), .sample_53n(reg_sample_53n), .sample_n21(
            reg_sample_n21), .sample_n53(reg_sample_n53),
2042        .sample_22n(reg_sample_22n), .sample_54n(reg_sample_54n), .sample_n22(
            reg_sample_n22), .sample_n54(reg_sample_n54),
2043        .sample_23n(reg_sample_23n), .sample_55n(reg_sample_55n), .sample_n23(
            reg_sample_n23), .sample_n55(reg_sample_n55),
2044        .sample_24n(reg_sample_24n), .sample_56n(reg_sample_56n), .sample_n24(
            reg_sample_n24), .sample_n56(reg_sample_n56),
2045        .sample_25n(reg_sample_25n), .sample_57n(reg_sample_57n), .sample_n25(
            reg_sample_n25), .sample_n57(reg_sample_n57),
2046        .sample_26n(reg_sample_26n), .sample_58n(reg_sample_58n), .sample_n26(
            reg_sample_n26), .sample_n58(reg_sample_n58),
2047        .sample_27n(reg_sample_27n), .sample_59n(reg_sample_59n), .sample_n27(
            reg_sample_n27), .sample_n59(reg_sample_n59),
2048        .sample_28n(reg_sample_28n), .sample_60n(reg_sample_60n), .sample_n28(
            reg_sample_n28), .sample_n60(reg_sample_n60),
2049        .sample_29n(reg_sample_29n), .sample_61n(reg_sample_61n), .sample_n29(
            reg_sample_n29), .sample_n61(reg_sample_n61),
2050        .sample_30n(reg_sample_30n), .sample_62n(reg_sample_62n), .sample_n30(
            reg_sample_n30), .sample_n62(reg_sample_n62),
2051        .sample_31n(reg_sample_31n), .sample_63n(reg_sample_63n), .sample_n31(
            reg_sample_n31), .sample_n63(reg_sample_n63),
2052
2053        .sample_NN(reg_sample_NN),
```

```
2054
2055        .ref1(angular_mode30_sample1),
2056        .ref2(angular_mode30_sample2),
2057        .ref3(angular_mode30_sample3),
2058        .ref4(angular_mode30_sample4),
2059        .ref5(angular_mode30_sample5)
2060    );
2061
2062    angular_refs_selector #(WIDTH, 31) angular_mode31_selector(
2063        .clock(clock),
2064        .selector(angular_mode31_samples),
2065        .sample_0n(reg_sample_0n),  .sample_32n(reg_sample_32n), .sample_n0(
               reg_sample_n0),   .sample_n32(reg_sample_n32),
2066        .sample_1n(reg_sample_1n),  .sample_33n(reg_sample_33n), .sample_n1(
               reg_sample_n1),   .sample_n33(reg_sample_n33),
2067        .sample_2n(reg_sample_2n),  .sample_34n(reg_sample_34n), .sample_n2(
               reg_sample_n2),   .sample_n34(reg_sample_n34),
2068        .sample_3n(reg_sample_3n),  .sample_35n(reg_sample_35n), .sample_n3(
               reg_sample_n3),   .sample_n35(reg_sample_n35),
2069        .sample_4n(reg_sample_4n),  .sample_36n(reg_sample_36n), .sample_n4(
               reg_sample_n4),   .sample_n36(reg_sample_n36),
2070        .sample_5n(reg_sample_5n),  .sample_37n(reg_sample_37n), .sample_n5(
               reg_sample_n5),   .sample_n37(reg_sample_n37),
2071        .sample_6n(reg_sample_6n),  .sample_38n(reg_sample_38n), .sample_n6(
               reg_sample_n6),   .sample_n38(reg_sample_n38),
2072        .sample_7n(reg_sample_7n),  .sample_39n(reg_sample_39n), .sample_n7(
               reg_sample_n7),   .sample_n39(reg_sample_n39),
2073        .sample_8n(reg_sample_8n),  .sample_40n(reg_sample_40n), .sample_n8(
               reg_sample_n8),   .sample_n40(reg_sample_n40),
2074        .sample_9n(reg_sample_9n),  .sample_41n(reg_sample_41n), .sample_n9(
               reg_sample_n9),   .sample_n41(reg_sample_n41),
2075        .sample_10n(reg_sample_10n), .sample_42n(reg_sample_42n), .sample_n10(
               reg_sample_n10), .sample_n42(reg_sample_n42),
2076        .sample_11n(reg_sample_11n), .sample_43n(reg_sample_43n), .sample_n11(
               reg_sample_n11), .sample_n43(reg_sample_n43),
2077        .sample_12n(reg_sample_12n), .sample_44n(reg_sample_44n), .sample_n12(
               reg_sample_n12), .sample_n44(reg_sample_n44),
2078        .sample_13n(reg_sample_13n), .sample_45n(reg_sample_45n), .sample_n13(
               reg_sample_n13), .sample_n45(reg_sample_n45),
2079        .sample_14n(reg_sample_14n), .sample_46n(reg_sample_46n), .sample_n14(
               reg_sample_n14), .sample_n46(reg_sample_n46),
2080        .sample_15n(reg_sample_15n), .sample_47n(reg_sample_47n), .sample_n15(
               reg_sample_n15), .sample_n47(reg_sample_n47),
2081        .sample_16n(reg_sample_16n), .sample_48n(reg_sample_48n), .sample_n16(
               reg_sample_n16), .sample_n48(reg_sample_n48),
2082        .sample_17n(reg_sample_17n), .sample_49n(reg_sample_49n), .sample_n17(
               reg_sample_n17), .sample_n49(reg_sample_n49),
2083        .sample_18n(reg_sample_18n), .sample_50n(reg_sample_50n), .sample_n18(
               reg_sample_n18), .sample_n50(reg_sample_n50),
2084        .sample_19n(reg_sample_19n), .sample_51n(reg_sample_51n), .sample_n19(
               reg_sample_n19), .sample_n51(reg_sample_n51),
2085        .sample_20n(reg_sample_20n), .sample_52n(reg_sample_52n), .sample_n20(
               reg_sample_n20), .sample_n52(reg_sample_n52),
2086        .sample_21n(reg_sample_21n), .sample_53n(reg_sample_53n), .sample_n21(
               reg_sample_n21), .sample_n53(reg_sample_n53),
2087        .sample_22n(reg_sample_22n), .sample_54n(reg_sample_54n), .sample_n22(
               reg_sample_n22), .sample_n54(reg_sample_n54),
```

```
2088        .sample_23n(reg_sample_23n), .sample_55n(reg_sample_55n), .sample_n23(
                reg_sample_n23), .sample_n55(reg_sample_n55),
2089        .sample_24n(reg_sample_24n), .sample_56n(reg_sample_56n), .sample_n24(
                reg_sample_n24), .sample_n56(reg_sample_n56),
2090        .sample_25n(reg_sample_25n), .sample_57n(reg_sample_57n), .sample_n25(
                reg_sample_n25), .sample_n57(reg_sample_n57),
2091        .sample_26n(reg_sample_26n), .sample_58n(reg_sample_58n), .sample_n26(
                reg_sample_n26), .sample_n58(reg_sample_n58),
2092        .sample_27n(reg_sample_27n), .sample_59n(reg_sample_59n), .sample_n27(
                reg_sample_n27), .sample_n59(reg_sample_n59),
2093        .sample_28n(reg_sample_28n), .sample_60n(reg_sample_60n), .sample_n28(
                reg_sample_n28), .sample_n60(reg_sample_n60),
2094        .sample_29n(reg_sample_29n), .sample_61n(reg_sample_61n), .sample_n29(
                reg_sample_n29), .sample_n61(reg_sample_n61),
2095        .sample_30n(reg_sample_30n), .sample_62n(reg_sample_62n), .sample_n30(
                reg_sample_n30), .sample_n62(reg_sample_n62),
2096        .sample_31n(reg_sample_31n), .sample_63n(reg_sample_63n), .sample_n31(
                reg_sample_n31), .sample_n63(reg_sample_n63),
2097
2098        .sample_NN(reg_sample_NN),
2099
2100        .ref1(angular_mode31_sample1),
2101        .ref2(angular_mode31_sample2),
2102        .ref3(angular_mode31_sample3),
2103        .ref4(angular_mode31_sample4),
2104        .ref5(angular_mode31_sample5)
2105    );
2106
2107    angular_refs_selector #(WIDTH, 32) angular_mode32_selector(
2108        .clock(clock),
2109        .selector(angular_mode32_samples),
2110        .sample_0n(reg_sample_0n),  .sample_32n(reg_sample_32n), .sample_n0(
                reg_sample_n0),   .sample_n32(reg_sample_n32),
2111        .sample_1n(reg_sample_1n),  .sample_33n(reg_sample_33n), .sample_n1(
                reg_sample_n1),   .sample_n33(reg_sample_n33),
2112        .sample_2n(reg_sample_2n),  .sample_34n(reg_sample_34n), .sample_n2(
                reg_sample_n2),   .sample_n34(reg_sample_n34),
2113        .sample_3n(reg_sample_3n),  .sample_35n(reg_sample_35n), .sample_n3(
                reg_sample_n3),   .sample_n35(reg_sample_n35),
2114        .sample_4n(reg_sample_4n),  .sample_36n(reg_sample_36n), .sample_n4(
                reg_sample_n4),   .sample_n36(reg_sample_n36),
2115        .sample_5n(reg_sample_5n),  .sample_37n(reg_sample_37n), .sample_n5(
                reg_sample_n5),   .sample_n37(reg_sample_n37),
2116        .sample_6n(reg_sample_6n),  .sample_38n(reg_sample_38n), .sample_n6(
                reg_sample_n6),   .sample_n38(reg_sample_n38),
2117        .sample_7n(reg_sample_7n),  .sample_39n(reg_sample_39n), .sample_n7(
                reg_sample_n7),   .sample_n39(reg_sample_n39),
2118        .sample_8n(reg_sample_8n),  .sample_40n(reg_sample_40n), .sample_n8(
                reg_sample_n8),   .sample_n40(reg_sample_n40),
2119        .sample_9n(reg_sample_9n),  .sample_41n(reg_sample_41n), .sample_n9(
                reg_sample_n9),   .sample_n41(reg_sample_n41),
2120        .sample_10n(reg_sample_10n), .sample_42n(reg_sample_42n), .sample_n10(
                reg_sample_n10), .sample_n42(reg_sample_n42),
2121        .sample_11n(reg_sample_11n), .sample_43n(reg_sample_43n), .sample_n11(
                reg_sample_n11), .sample_n43(reg_sample_n43),
2122        .sample_12n(reg_sample_12n), .sample_44n(reg_sample_44n), .sample_n12(
                reg_sample_n12), .sample_n44(reg_sample_n44),
```

```
2123        .sample_13n(reg_sample_13n), .sample_45n(reg_sample_45n), .sample_n13(
               reg_sample_n13), .sample_n45(reg_sample_n45),
2124        .sample_14n(reg_sample_14n), .sample_46n(reg_sample_46n), .sample_n14(
               reg_sample_n14), .sample_n46(reg_sample_n46),
2125        .sample_15n(reg_sample_15n), .sample_47n(reg_sample_47n), .sample_n15(
               reg_sample_n15), .sample_n47(reg_sample_n47),
2126        .sample_16n(reg_sample_16n), .sample_48n(reg_sample_48n), .sample_n16(
               reg_sample_n16), .sample_n48(reg_sample_n48),
2127        .sample_17n(reg_sample_17n), .sample_49n(reg_sample_49n), .sample_n17(
               reg_sample_n17), .sample_n49(reg_sample_n49),
2128        .sample_18n(reg_sample_18n), .sample_50n(reg_sample_50n), .sample_n18(
               reg_sample_n18), .sample_n50(reg_sample_n50),
2129        .sample_19n(reg_sample_19n), .sample_51n(reg_sample_51n), .sample_n19(
               reg_sample_n19), .sample_n51(reg_sample_n51),
2130        .sample_20n(reg_sample_20n), .sample_52n(reg_sample_52n), .sample_n20(
               reg_sample_n20), .sample_n52(reg_sample_n52),
2131        .sample_21n(reg_sample_21n), .sample_53n(reg_sample_53n), .sample_n21(
               reg_sample_n21), .sample_n53(reg_sample_n53),
2132        .sample_22n(reg_sample_22n), .sample_54n(reg_sample_54n), .sample_n22(
               reg_sample_n22), .sample_n54(reg_sample_n54),
2133        .sample_23n(reg_sample_23n), .sample_55n(reg_sample_55n), .sample_n23(
               reg_sample_n23), .sample_n55(reg_sample_n55),
2134        .sample_24n(reg_sample_24n), .sample_56n(reg_sample_56n), .sample_n24(
               reg_sample_n24), .sample_n56(reg_sample_n56),
2135        .sample_25n(reg_sample_25n), .sample_57n(reg_sample_57n), .sample_n25(
               reg_sample_n25), .sample_n57(reg_sample_n57),
2136        .sample_26n(reg_sample_26n), .sample_58n(reg_sample_58n), .sample_n26(
               reg_sample_n26), .sample_n58(reg_sample_n58),
2137        .sample_27n(reg_sample_27n), .sample_59n(reg_sample_59n), .sample_n27(
               reg_sample_n27), .sample_n59(reg_sample_n59),
2138        .sample_28n(reg_sample_28n), .sample_60n(reg_sample_60n), .sample_n28(
               reg_sample_n28), .sample_n60(reg_sample_n60),
2139        .sample_29n(reg_sample_29n), .sample_61n(reg_sample_61n), .sample_n29(
               reg_sample_n29), .sample_n61(reg_sample_n61),
2140        .sample_30n(reg_sample_30n), .sample_62n(reg_sample_62n), .sample_n30(
               reg_sample_n30), .sample_n62(reg_sample_n62),
2141        .sample_31n(reg_sample_31n), .sample_63n(reg_sample_63n), .sample_n31(
               reg_sample_n31), .sample_n63(reg_sample_n63),
2142
2143        .sample_NN(reg_sample_NN),
2144
2145        .ref1(angular_mode32_sample1),
2146        .ref2(angular_mode32_sample2),
2147        .ref3(angular_mode32_sample3),
2148        .ref4(angular_mode32_sample4),
2149        .ref5(angular_mode32_sample5)
2150    );
2151
2152    angular_refs_selector #(WIDTH, 33) angular_mode33_selector(
2153        .clock(clock),
2154        .selector(angular_mode33_samples),
2155        .sample_0n(reg_sample_0n),  .sample_32n(reg_sample_32n), .sample_n0(
               reg_sample_n0),   .sample_n32(reg_sample_n32),
2156        .sample_1n(reg_sample_1n),  .sample_33n(reg_sample_33n), .sample_n1(
               reg_sample_n1),   .sample_n33(reg_sample_n33),
2157        .sample_2n(reg_sample_2n),  .sample_34n(reg_sample_34n), .sample_n2(
               reg_sample_n2),   .sample_n34(reg_sample_n34),
```

```
2158        .sample_3n(reg_sample_3n),  .sample_35n(reg_sample_35n), .sample_n3(
                reg_sample_n3),   .sample_n35(reg_sample_n35),
2159        .sample_4n(reg_sample_4n),  .sample_36n(reg_sample_36n), .sample_n4(
                reg_sample_n4),   .sample_n36(reg_sample_n36),
2160        .sample_5n(reg_sample_5n),  .sample_37n(reg_sample_37n), .sample_n5(
                reg_sample_n5),   .sample_n37(reg_sample_n37),
2161        .sample_6n(reg_sample_6n),  .sample_38n(reg_sample_38n), .sample_n6(
                reg_sample_n6),   .sample_n38(reg_sample_n38),
2162        .sample_7n(reg_sample_7n),  .sample_39n(reg_sample_39n), .sample_n7(
                reg_sample_n7),   .sample_n39(reg_sample_n39),
2163        .sample_8n(reg_sample_8n),  .sample_40n(reg_sample_40n), .sample_n8(
                reg_sample_n8),   .sample_n40(reg_sample_n40),
2164        .sample_9n(reg_sample_9n),  .sample_41n(reg_sample_41n), .sample_n9(
                reg_sample_n9),   .sample_n41(reg_sample_n41),
2165        .sample_10n(reg_sample_10n), .sample_42n(reg_sample_42n), .sample_n10(
                reg_sample_n10), .sample_n42(reg_sample_n42),
2166        .sample_11n(reg_sample_11n), .sample_43n(reg_sample_43n), .sample_n11(
                reg_sample_n11), .sample_n43(reg_sample_n43),
2167        .sample_12n(reg_sample_12n), .sample_44n(reg_sample_44n), .sample_n12(
                reg_sample_n12), .sample_n44(reg_sample_n44),
2168        .sample_13n(reg_sample_13n), .sample_45n(reg_sample_45n), .sample_n13(
                reg_sample_n13), .sample_n45(reg_sample_n45),
2169        .sample_14n(reg_sample_14n), .sample_46n(reg_sample_46n), .sample_n14(
                reg_sample_n14), .sample_n46(reg_sample_n46),
2170        .sample_15n(reg_sample_15n), .sample_47n(reg_sample_47n), .sample_n15(
                reg_sample_n15), .sample_n47(reg_sample_n47),
2171        .sample_16n(reg_sample_16n), .sample_48n(reg_sample_48n), .sample_n16(
                reg_sample_n16), .sample_n48(reg_sample_n48),
2172        .sample_17n(reg_sample_17n), .sample_49n(reg_sample_49n), .sample_n17(
                reg_sample_n17), .sample_n49(reg_sample_n49),
2173        .sample_18n(reg_sample_18n), .sample_50n(reg_sample_50n), .sample_n18(
                reg_sample_n18), .sample_n50(reg_sample_n50),
2174        .sample_19n(reg_sample_19n), .sample_51n(reg_sample_51n), .sample_n19(
                reg_sample_n19), .sample_n51(reg_sample_n51),
2175        .sample_20n(reg_sample_20n), .sample_52n(reg_sample_52n), .sample_n20(
                reg_sample_n20), .sample_n52(reg_sample_n52),
2176        .sample_21n(reg_sample_21n), .sample_53n(reg_sample_53n), .sample_n21(
                reg_sample_n21), .sample_n53(reg_sample_n53),
2177        .sample_22n(reg_sample_22n), .sample_54n(reg_sample_54n), .sample_n22(
                reg_sample_n22), .sample_n54(reg_sample_n54),
2178        .sample_23n(reg_sample_23n), .sample_55n(reg_sample_55n), .sample_n23(
                reg_sample_n23), .sample_n55(reg_sample_n55),
2179        .sample_24n(reg_sample_24n), .sample_56n(reg_sample_56n), .sample_n24(
                reg_sample_n24), .sample_n56(reg_sample_n56),
2180        .sample_25n(reg_sample_25n), .sample_57n(reg_sample_57n), .sample_n25(
                reg_sample_n25), .sample_n57(reg_sample_n57),
2181        .sample_26n(reg_sample_26n), .sample_58n(reg_sample_58n), .sample_n26(
                reg_sample_n26), .sample_n58(reg_sample_n58),
2182        .sample_27n(reg_sample_27n), .sample_59n(reg_sample_59n), .sample_n27(
                reg_sample_n27), .sample_n59(reg_sample_n59),
2183        .sample_28n(reg_sample_28n), .sample_60n(reg_sample_60n), .sample_n28(
                reg_sample_n28), .sample_n60(reg_sample_n60),
2184        .sample_29n(reg_sample_29n), .sample_61n(reg_sample_61n), .sample_n29(
                reg_sample_n29), .sample_n61(reg_sample_n61),
2185        .sample_30n(reg_sample_30n), .sample_62n(reg_sample_62n), .sample_n30(
                reg_sample_n30), .sample_n62(reg_sample_n62),
2186        .sample_31n(reg_sample_31n), .sample_63n(reg_sample_63n), .sample_n31(
```

```
            reg_sample_n31), .sample_n63(reg_sample_n63),
2187
2188        .sample_NN(reg_sample_NN),
2189
2190        .ref1(angular_mode33_sample1),
2191        .ref2(angular_mode33_sample2),
2192        .ref3(angular_mode33_sample3),
2193        .ref4(angular_mode33_sample4),
2194        .ref5(angular_mode33_sample5)
2195    );
2196
2197    angular_refs_selector #(WIDTH, 34) angular_mode34_selector(
2198        .clock(clock),
2199        .selector(angular_mode34_samples),
2200        .sample_0n(reg_sample_0n),  .sample_32n(reg_sample_32n), .sample_n0(
                reg_sample_n0),   .sample_n32(reg_sample_n32),
2201        .sample_1n(reg_sample_1n),  .sample_33n(reg_sample_33n), .sample_n1(
                reg_sample_n1),   .sample_n33(reg_sample_n33),
2202        .sample_2n(reg_sample_2n),  .sample_34n(reg_sample_34n), .sample_n2(
                reg_sample_n2),   .sample_n34(reg_sample_n34),
2203        .sample_3n(reg_sample_3n),  .sample_35n(reg_sample_35n), .sample_n3(
                reg_sample_n3),   .sample_n35(reg_sample_n35),
2204        .sample_4n(reg_sample_4n),  .sample_36n(reg_sample_36n), .sample_n4(
                reg_sample_n4),   .sample_n36(reg_sample_n36),
2205        .sample_5n(reg_sample_5n),  .sample_37n(reg_sample_37n), .sample_n5(
                reg_sample_n5),   .sample_n37(reg_sample_n37),
2206        .sample_6n(reg_sample_6n),  .sample_38n(reg_sample_38n), .sample_n6(
                reg_sample_n6),   .sample_n38(reg_sample_n38),
2207        .sample_7n(reg_sample_7n),  .sample_39n(reg_sample_39n), .sample_n7(
                reg_sample_n7),   .sample_n39(reg_sample_n39),
2208        .sample_8n(reg_sample_8n),  .sample_40n(reg_sample_40n), .sample_n8(
                reg_sample_n8),   .sample_n40(reg_sample_n40),
2209        .sample_9n(reg_sample_9n),  .sample_41n(reg_sample_41n), .sample_n9(
                reg_sample_n9),   .sample_n41(reg_sample_n41),
2210        .sample_10n(reg_sample_10n), .sample_42n(reg_sample_42n), .sample_n10(
                reg_sample_n10), .sample_n42(reg_sample_n42),
2211        .sample_11n(reg_sample_11n), .sample_43n(reg_sample_43n), .sample_n11(
                reg_sample_n11), .sample_n43(reg_sample_n43),
2212        .sample_12n(reg_sample_12n), .sample_44n(reg_sample_44n), .sample_n12(
                reg_sample_n12), .sample_n44(reg_sample_n44),
2213        .sample_13n(reg_sample_13n), .sample_45n(reg_sample_45n), .sample_n13(
                reg_sample_n13), .sample_n45(reg_sample_n45),
2214        .sample_14n(reg_sample_14n), .sample_46n(reg_sample_46n), .sample_n14(
                reg_sample_n14), .sample_n46(reg_sample_n46),
2215        .sample_15n(reg_sample_15n), .sample_47n(reg_sample_47n), .sample_n15(
                reg_sample_n15), .sample_n47(reg_sample_n47),
2216        .sample_16n(reg_sample_16n), .sample_48n(reg_sample_48n), .sample_n16(
                reg_sample_n16), .sample_n48(reg_sample_n48),
2217        .sample_17n(reg_sample_17n), .sample_49n(reg_sample_49n), .sample_n17(
                reg_sample_n17), .sample_n49(reg_sample_n49),
2218        .sample_18n(reg_sample_18n), .sample_50n(reg_sample_50n), .sample_n18(
                reg_sample_n18), .sample_n50(reg_sample_n50),
2219        .sample_19n(reg_sample_19n), .sample_51n(reg_sample_51n), .sample_n19(
                reg_sample_n19), .sample_n51(reg_sample_n51),
2220        .sample_20n(reg_sample_20n), .sample_52n(reg_sample_52n), .sample_n20(
                reg_sample_n20), .sample_n52(reg_sample_n52),
2221        .sample_21n(reg_sample_21n), .sample_53n(reg_sample_53n), .sample_n21(
```

```
            reg_sample_n21), .sample_n53(reg_sample_n53),
2222        .sample_22n(reg_sample_22n), .sample_54n(reg_sample_54n), .sample_n22(
            reg_sample_n22), .sample_n54(reg_sample_n54),
2223        .sample_23n(reg_sample_23n), .sample_55n(reg_sample_55n), .sample_n23(
            reg_sample_n23), .sample_n55(reg_sample_n55),
2224        .sample_24n(reg_sample_24n), .sample_56n(reg_sample_56n), .sample_n24(
            reg_sample_n24), .sample_n56(reg_sample_n56),
2225        .sample_25n(reg_sample_25n), .sample_57n(reg_sample_57n), .sample_n25(
            reg_sample_n25), .sample_n57(reg_sample_n57),
2226        .sample_26n(reg_sample_26n), .sample_58n(reg_sample_58n), .sample_n26(
            reg_sample_n26), .sample_n58(reg_sample_n58),
2227        .sample_27n(reg_sample_27n), .sample_59n(reg_sample_59n), .sample_n27(
            reg_sample_n27), .sample_n59(reg_sample_n59),
2228        .sample_28n(reg_sample_28n), .sample_60n(reg_sample_60n), .sample_n28(
            reg_sample_n28), .sample_n60(reg_sample_n60),
2229        .sample_29n(reg_sample_29n), .sample_61n(reg_sample_61n), .sample_n29(
            reg_sample_n29), .sample_n61(reg_sample_n61),
2230        .sample_30n(reg_sample_30n), .sample_62n(reg_sample_62n), .sample_n30(
            reg_sample_n30), .sample_n62(reg_sample_n62),
2231        .sample_31n(reg_sample_31n), .sample_63n(reg_sample_63n), .sample_n31(
            reg_sample_n31), .sample_n63(reg_sample_n63),
2232
2233        .sample_NN(reg_sample_NN),
2234
2235        .ref1(angular_mode34_sample1),
2236        .ref2(angular_mode34_sample2),
2237        .ref3(angular_mode34_sample3),
2238        .ref4(angular_mode34_sample4),
2239        .ref5(angular_mode34_sample5)
2240    );
2241
2242    always @(posedge clock)
2243    begin
2244
2245        if (reset)
2246        begin
2247            reg_i <= 5'b00000;
2248            reg_iterations <= 5'b00000;
2249
2250            reg_sample_0n <= 8'b00000000; reg_sample_32n <= 8'b00000000;
                reg_sample_n0 <= 8'b00000000; reg_sample_n32 <= 8'b00000000;
2251            reg_sample_1n <= 8'b00000000; reg_sample_33n <= 8'b00000000;
                reg_sample_n1 <= 8'b00000000; reg_sample_n33 <= 8'b00000000;
2252            reg_sample_2n <= 8'b00000000; reg_sample_34n <= 8'b00000000;
                reg_sample_n2 <= 8'b00000000; reg_sample_n34 <= 8'b00000000;
2253            reg_sample_3n <= 8'b00000000; reg_sample_35n <= 8'b00000000;
                reg_sample_n3 <= 8'b00000000; reg_sample_n35 <= 8'b00000000;
2254            reg_sample_4n <= 8'b00000000; reg_sample_36n <= 8'b00000000;
                reg_sample_n4 <= 8'b00000000; reg_sample_n36 <= 8'b00000000;
2255            reg_sample_5n <= 8'b00000000; reg_sample_37n <= 8'b00000000;
                reg_sample_n5 <= 8'b00000000; reg_sample_n37 <= 8'b00000000;
2256            reg_sample_6n <= 8'b00000000; reg_sample_38n <= 8'b00000000;
                reg_sample_n6 <= 8'b00000000; reg_sample_n38 <= 8'b00000000;
2257            reg_sample_7n <= 8'b00000000; reg_sample_39n <= 8'b00000000;
                reg_sample_n7 <= 8'b00000000; reg_sample_n39 <= 8'b00000000;
2258            reg_sample_8n <= 8'b00000000; reg_sample_40n <= 8'b00000000;
                reg_sample_n8 <= 8'b00000000; reg_sample_n40 <= 8'b00000000;
```

```
2259        reg_sample_9n <= 8'b00000000; reg_sample_41n <= 8'b00000000;
                reg_sample_n9 <= 8'b00000000; reg_sample_n41 <= 8'b00000000;
2260        reg_sample_10n <= 8'b00000000;   reg_sample_42n <= 8'b00000000;
                reg_sample_n10 <= 8'b00000000;   reg_sample_n42 <= 8'b00000000;
2261        reg_sample_11n <= 8'b00000000;   reg_sample_43n <= 8'b00000000;
                reg_sample_n11 <= 8'b00000000;   reg_sample_n43 <= 8'b00000000;
2262        reg_sample_12n <= 8'b00000000;   reg_sample_44n <= 8'b00000000;
                reg_sample_n12 <= 8'b00000000;   reg_sample_n44 <= 8'b00000000;
2263        reg_sample_13n <= 8'b00000000;   reg_sample_45n <= 8'b00000000;
                reg_sample_n13 <= 8'b00000000;   reg_sample_n45 <= 8'b00000000;
2264        reg_sample_14n <= 8'b00000000;   reg_sample_46n <= 8'b00000000;
                reg_sample_n14 <= 8'b00000000;   reg_sample_n46 <= 8'b00000000;
2265        reg_sample_15n <= 8'b00000000;   reg_sample_47n <= 8'b00000000;
                reg_sample_n15 <= 8'b00000000;   reg_sample_n47 <= 8'b00000000;
2266        reg_sample_16n <= 8'b00000000;   reg_sample_48n <= 8'b00000000;
                reg_sample_n16 <= 8'b00000000;   reg_sample_n48 <= 8'b00000000;
2267        reg_sample_17n <= 8'b00000000;   reg_sample_49n <= 8'b00000000;
                reg_sample_n17 <= 8'b00000000;   reg_sample_n49 <= 8'b00000000;
2268        reg_sample_18n <= 8'b00000000;   reg_sample_50n <= 8'b00000000;
                reg_sample_n18 <= 8'b00000000;   reg_sample_n50 <= 8'b00000000;
2269        reg_sample_19n <= 8'b00000000;   reg_sample_51n <= 8'b00000000;
                reg_sample_n19 <= 8'b00000000;   reg_sample_n51 <= 8'b00000000;
2270        reg_sample_20n <= 8'b00000000;   reg_sample_52n <= 8'b00000000;
                reg_sample_n20 <= 8'b00000000;   reg_sample_n52 <= 8'b00000000;
2271        reg_sample_21n <= 8'b00000000;   reg_sample_53n <= 8'b00000000;
                reg_sample_n21 <= 8'b00000000;   reg_sample_n53 <= 8'b00000000;
2272        reg_sample_22n <= 8'b00000000;   reg_sample_54n <= 8'b00000000;
                reg_sample_n22 <= 8'b00000000;   reg_sample_n54 <= 8'b00000000;
2273        reg_sample_23n <= 8'b00000000;   reg_sample_55n <= 8'b00000000;
                reg_sample_n23 <= 8'b00000000;   reg_sample_n55 <= 8'b00000000;
2274        reg_sample_24n <= 8'b00000000;   reg_sample_56n <= 8'b00000000;
                reg_sample_n24 <= 8'b00000000;   reg_sample_n56 <= 8'b00000000;
2275        reg_sample_25n <= 8'b00000000;   reg_sample_57n <= 8'b00000000;
                reg_sample_n25 <= 8'b00000000;   reg_sample_n57 <= 8'b00000000;
2276        reg_sample_26n <= 8'b00000000;   reg_sample_58n <= 8'b00000000;
                reg_sample_n26 <= 8'b00000000;   reg_sample_n58 <= 8'b00000000;
2277        reg_sample_27n <= 8'b00000000;   reg_sample_59n <= 8'b00000000;
                reg_sample_n27 <= 8'b00000000;   reg_sample_n59 <= 8'b00000000;
2278        reg_sample_28n <= 8'b00000000;   reg_sample_60n <= 8'b00000000;
                reg_sample_n28 <= 8'b00000000;   reg_sample_n60 <= 8'b00000000;
2279        reg_sample_29n <= 8'b00000000;   reg_sample_61n <= 8'b00000000;
                reg_sample_n29 <= 8'b00000000;   reg_sample_n61 <= 8'b00000000;
2280        reg_sample_30n <= 8'b00000000;   reg_sample_62n <= 8'b00000000;
                reg_sample_n30 <= 8'b00000000;   reg_sample_n62 <= 8'b00000000;
2281        reg_sample_31n <= 8'b00000000;   reg_sample_63n <= 8'b00000000;
                reg_sample_n31 <= 8'b00000000;   reg_sample_n63 <= 8'b00000000;
2282
2283        reg_sample_Nn <= 8'b00000000; reg_sample_nN <= 8'b00000000;
                reg_sample_NN <= 8'b00000000;
2284
2285        reg_planar_sample_0n <= 8'b00000000;
2286        reg_planar_sample_1n <= 8'b00000000;
2287        reg_planar_sample_2n <= 8'b00000000;
2288        reg_planar_sample_3n <= 8'b00000000;
2289        reg_planar_sample_n0 <= 8'b00000000;
2290        reg_planar_sample_n1 <= 8'b00000000;
2291        reg_planar_sample_n2 <= 8'b00000000;
```

```
2292            reg_planar_sample_n3 <= 8'b00000000;
2293
2294            reg_dc_sample_0n <= 8'b00000000;
2295            reg_dc_sample_1n <= 8'b00000000;
2296            reg_dc_sample_2n <= 8'b00000000;
2297            reg_dc_sample_3n <= 8'b00000000;
2298            reg_dc_sample_n0 <= 8'b00000000;
2299            reg_dc_sample_n1 <= 8'b00000000;
2300            reg_dc_sample_n2 <= 8'b00000000;
2301            reg_dc_sample_n3 <= 8'b00000000;
2302
2303            reg_angular_mode2_ref1 <= 8'b00000000;
2304            reg_angular_mode2_ref2 <= 8'b00000000;
2305            reg_angular_mode2_ref3 <= 8'b00000000;
2306            reg_angular_mode2_ref4 <= 8'b00000000;
2307            reg_angular_mode2_ref5 <= 8'b00000000;
2308
2309            reg_angular_mode3_ref1 <= 8'b00000000;
2310            reg_angular_mode3_ref2 <= 8'b00000000;
2311            reg_angular_mode3_ref3 <= 8'b00000000;
2312            reg_angular_mode3_ref4 <= 8'b00000000;
2313            reg_angular_mode3_ref5 <= 8'b00000000;
2314
2315            reg_angular_mode4_ref1 <= 8'b00000000;
2316            reg_angular_mode4_ref2 <= 8'b00000000;
2317            reg_angular_mode4_ref3 <= 8'b00000000;
2318            reg_angular_mode4_ref4 <= 8'b00000000;
2319            reg_angular_mode4_ref5 <= 8'b00000000;
2320
2321            reg_angular_mode5_ref1 <= 8'b00000000;
2322            reg_angular_mode5_ref2 <= 8'b00000000;
2323            reg_angular_mode5_ref3 <= 8'b00000000;
2324            reg_angular_mode5_ref4 <= 8'b00000000;
2325            reg_angular_mode5_ref5 <= 8'b00000000;
2326
2327            reg_angular_mode6_ref1 <= 8'b00000000;
2328            reg_angular_mode6_ref2 <= 8'b00000000;
2329            reg_angular_mode6_ref3 <= 8'b00000000;
2330            reg_angular_mode6_ref4 <= 8'b00000000;
2331            reg_angular_mode6_ref5 <= 8'b00000000;
2332
2333            reg_angular_mode7_ref1 <= 8'b00000000;
2334            reg_angular_mode7_ref2 <= 8'b00000000;
2335            reg_angular_mode7_ref3 <= 8'b00000000;
2336            reg_angular_mode7_ref4 <= 8'b00000000;
2337            reg_angular_mode7_ref5 <= 8'b00000000;
2338
2339            reg_angular_mode8_ref1 <= 8'b00000000;
2340            reg_angular_mode8_ref2 <= 8'b00000000;
2341            reg_angular_mode8_ref3 <= 8'b00000000;
2342            reg_angular_mode8_ref4 <= 8'b00000000;
2343            reg_angular_mode8_ref5 <= 8'b00000000;
2344
2345            reg_angular_mode9_ref1 <= 8'b00000000;
2346            reg_angular_mode9_ref2 <= 8'b00000000;
2347            reg_angular_mode9_ref3 <= 8'b00000000;
2348            reg_angular_mode9_ref4 <= 8'b00000000;
```

```
2349            reg_angular_mode9_ref5 <= 8'b00000000;
2350
2351            reg_angular_mode10_ref0 <= 8'b00000000;
2352            reg_angular_mode10_ref1 <= 8'b00000000;
2353            reg_angular_mode10_ref2 <= 8'b00000000;
2354            reg_angular_mode10_ref3 <= 8'b00000000;
2355            reg_angular_mode10_ref4 <= 8'b00000000;
2356            reg_angular_mode10_ref5 <= 8'b00000000;
2357
2358            reg_angular_mode11_ref1 <= 8'b00000000;
2359            reg_angular_mode11_ref2 <= 8'b00000000;
2360            reg_angular_mode11_ref3 <= 8'b00000000;
2361            reg_angular_mode11_ref4 <= 8'b00000000;
2362            reg_angular_mode11_ref5 <= 8'b00000000;
2363
2364            reg_angular_mode12_ref1 <= 8'b00000000;
2365            reg_angular_mode12_ref2 <= 8'b00000000;
2366            reg_angular_mode12_ref3 <= 8'b00000000;
2367            reg_angular_mode12_ref4 <= 8'b00000000;
2368            reg_angular_mode12_ref5 <= 8'b00000000;
2369
2370            reg_angular_mode13_ref1 <= 8'b00000000;
2371            reg_angular_mode13_ref2 <= 8'b00000000;
2372            reg_angular_mode13_ref3 <= 8'b00000000;
2373            reg_angular_mode13_ref4 <= 8'b00000000;
2374            reg_angular_mode13_ref5 <= 8'b00000000;
2375
2376            reg_angular_mode14_ref1 <= 8'b00000000;
2377            reg_angular_mode14_ref2 <= 8'b00000000;
2378            reg_angular_mode14_ref3 <= 8'b00000000;
2379            reg_angular_mode14_ref4 <= 8'b00000000;
2380            reg_angular_mode14_ref5 <= 8'b00000000;
2381
2382            reg_angular_mode15_ref1 <= 8'b00000000;
2383            reg_angular_mode15_ref2 <= 8'b00000000;
2384            reg_angular_mode15_ref3 <= 8'b00000000;
2385            reg_angular_mode15_ref4 <= 8'b00000000;
2386            reg_angular_mode15_ref5 <= 8'b00000000;
2387
2388            reg_angular_mode16_ref1 <= 8'b00000000;
2389            reg_angular_mode16_ref2 <= 8'b00000000;
2390            reg_angular_mode16_ref3 <= 8'b00000000;
2391            reg_angular_mode16_ref4 <= 8'b00000000;
2392            reg_angular_mode16_ref5 <= 8'b00000000;
2393
2394            reg_angular_mode17_ref1 <= 8'b00000000;
2395            reg_angular_mode17_ref2 <= 8'b00000000;
2396            reg_angular_mode17_ref3 <= 8'b00000000;
2397            reg_angular_mode17_ref4 <= 8'b00000000;
2398            reg_angular_mode17_ref5 <= 8'b00000000;
2399
2400            reg_angular_mode18_ref1 <= 8'b00000000;
2401            reg_angular_mode18_ref2 <= 8'b00000000;
2402            reg_angular_mode18_ref3 <= 8'b00000000;
2403            reg_angular_mode18_ref4 <= 8'b00000000;
2404            reg_angular_mode18_ref5 <= 8'b00000000;
2405
```

```
2406            reg_angular_mode19_ref1 <= 8'b00000000;
2407            reg_angular_mode19_ref2 <= 8'b00000000;
2408            reg_angular_mode19_ref3 <= 8'b00000000;
2409            reg_angular_mode19_ref4 <= 8'b00000000;
2410            reg_angular_mode19_ref5 <= 8'b00000000;
2411
2412            reg_angular_mode20_ref1 <= 8'b00000000;
2413            reg_angular_mode20_ref2 <= 8'b00000000;
2414            reg_angular_mode20_ref3 <= 8'b00000000;
2415            reg_angular_mode20_ref4 <= 8'b00000000;
2416            reg_angular_mode20_ref5 <= 8'b00000000;
2417
2418            reg_angular_mode21_ref1 <= 8'b00000000;
2419            reg_angular_mode21_ref2 <= 8'b00000000;
2420            reg_angular_mode21_ref3 <= 8'b00000000;
2421            reg_angular_mode21_ref4 <= 8'b00000000;
2422            reg_angular_mode21_ref5 <= 8'b00000000;
2423
2424            reg_angular_mode22_ref1 <= 8'b00000000;
2425            reg_angular_mode22_ref2 <= 8'b00000000;
2426            reg_angular_mode22_ref3 <= 8'b00000000;
2427            reg_angular_mode22_ref4 <= 8'b00000000;
2428            reg_angular_mode22_ref5 <= 8'b00000000;
2429
2430            reg_angular_mode23_ref1 <= 8'b00000000;
2431            reg_angular_mode23_ref2 <= 8'b00000000;
2432            reg_angular_mode23_ref3 <= 8'b00000000;
2433            reg_angular_mode23_ref4 <= 8'b00000000;
2434            reg_angular_mode23_ref5 <= 8'b00000000;
2435
2436            reg_angular_mode24_ref1 <= 8'b00000000;
2437            reg_angular_mode24_ref2 <= 8'b00000000;
2438            reg_angular_mode24_ref3 <= 8'b00000000;
2439            reg_angular_mode24_ref4 <= 8'b00000000;
2440            reg_angular_mode24_ref5 <= 8'b00000000;
2441
2442            reg_angular_mode25_ref1 <= 8'b00000000;
2443            reg_angular_mode25_ref2 <= 8'b00000000;
2444            reg_angular_mode25_ref3 <= 8'b00000000;
2445            reg_angular_mode25_ref4 <= 8'b00000000;
2446            reg_angular_mode25_ref5 <= 8'b00000000;
2447
2448            reg_angular_mode26_ref0 <= 8'b00000000;
2449            reg_angular_mode26_ref1 <= 8'b00000000;
2450            reg_angular_mode26_ref2 <= 8'b00000000;
2451            reg_angular_mode26_ref3 <= 8'b00000000;
2452            reg_angular_mode26_ref4 <= 8'b00000000;
2453            reg_angular_mode26_ref5 <= 8'b00000000;
2454
2455            reg_angular_mode27_ref1 <= 8'b00000000;
2456            reg_angular_mode27_ref2 <= 8'b00000000;
2457            reg_angular_mode27_ref3 <= 8'b00000000;
2458            reg_angular_mode27_ref4 <= 8'b00000000;
2459            reg_angular_mode27_ref5 <= 8'b00000000;
2460
2461            reg_angular_mode28_ref1 <= 8'b00000000;
2462            reg_angular_mode28_ref2 <= 8'b00000000;
```

```
2463            reg_angular_mode28_ref3 <= 8'b00000000;
2464            reg_angular_mode28_ref4 <= 8'b00000000;
2465            reg_angular_mode28_ref5 <= 8'b00000000;
2466
2467            reg_angular_mode29_ref1 <= 8'b00000000;
2468            reg_angular_mode29_ref2 <= 8'b00000000;
2469            reg_angular_mode29_ref3 <= 8'b00000000;
2470            reg_angular_mode29_ref4 <= 8'b00000000;
2471            reg_angular_mode29_ref5 <= 8'b00000000;
2472
2473            reg_angular_mode30_ref1 <= 8'b00000000;
2474            reg_angular_mode30_ref2 <= 8'b00000000;
2475            reg_angular_mode30_ref3 <= 8'b00000000;
2476            reg_angular_mode30_ref4 <= 8'b00000000;
2477            reg_angular_mode30_ref5 <= 8'b00000000;
2478
2479            reg_angular_mode31_ref1 <= 8'b00000000;
2480            reg_angular_mode31_ref2 <= 8'b00000000;
2481            reg_angular_mode31_ref3 <= 8'b00000000;
2482            reg_angular_mode31_ref4 <= 8'b00000000;
2483            reg_angular_mode31_ref5 <= 8'b00000000;
2484
2485            reg_angular_mode32_ref1 <= 8'b00000000;
2486            reg_angular_mode32_ref2 <= 8'b00000000;
2487            reg_angular_mode32_ref3 <= 8'b00000000;
2488            reg_angular_mode32_ref4 <= 8'b00000000;
2489            reg_angular_mode32_ref5 <= 8'b00000000;
2490
2491            reg_angular_mode33_ref1 <= 8'b00000000;
2492            reg_angular_mode33_ref2 <= 8'b00000000;
2493            reg_angular_mode33_ref3 <= 8'b00000000;
2494            reg_angular_mode33_ref4 <= 8'b00000000;
2495            reg_angular_mode33_ref5 <= 8'b00000000;
2496
2497            reg_angular_mode34_ref1 <= 8'b00000000;
2498            reg_angular_mode34_ref2 <= 8'b00000000;
2499            reg_angular_mode34_ref3 <= 8'b00000000;
2500            reg_angular_mode34_ref4 <= 8'b00000000;
2501            reg_angular_mode34_ref5 <= 8'b00000000;
2502        end
2503        else
2504        begin
2505
2506            if (def_enable)
2507            begin
2508                reg_sample_Nn <= sample_Nn;
2509                reg_sample_nN <= sample_nN;
2510                reg_sample_NN <= sample_NN;
2511
2512                case (N)
2513                    4:       begin reg_iterations <= 5'b00010; end
2514                    8:       begin reg_iterations <= 5'b00100; end
2515                    16:         begin reg_iterations <= 5'b01000; end
2516                    32:         begin reg_iterations <= 5'b10000; end
2517                    default: begin reg_iterations <= 5'b00010; end
2518                endcase
2519            end
```

```
2520
2521            if (samples_enable)
2522            begin
2523               case (reg_i)
2524                  1: begin
2525                        reg_sample_0n <= sample_0n;
2526                        reg_sample_1n <= sample_1n;
2527                        reg_sample_2n <= sample_2n;
2528                        reg_sample_3n <= sample_3n;
2529                        reg_sample_n0 <= sample_n0;
2530                        reg_sample_n1 <= sample_n1;
2531                        reg_sample_n2 <= sample_n2;
2532                        reg_sample_n3 <= sample_n3;
2533                     end
2534
2535                  2: begin
2536                        reg_sample_4n <= sample_0n;
2537                        reg_sample_5n <= sample_1n;
2538                        reg_sample_6n <= sample_2n;
2539                        reg_sample_7n <= sample_3n;
2540                        reg_sample_n4 <= sample_n0;
2541                        reg_sample_n5 <= sample_n1;
2542                        reg_sample_n6 <= sample_n2;
2543                        reg_sample_n7 <= sample_n3;
2544                     end
2545
2546                  3: begin
2547                        reg_sample_8n  <= sample_0n;
2548                        reg_sample_9n  <= sample_1n;
2549                        reg_sample_10n <= sample_2n;
2550                        reg_sample_11n <= sample_3n;
2551                        reg_sample_n8  <= sample_n0;
2552                        reg_sample_n9  <= sample_n1;
2553                        reg_sample_n10 <= sample_n2;
2554                        reg_sample_n11 <= sample_n3;
2555                     end
2556
2557                  4: begin
2558                        reg_sample_12n <= sample_0n;
2559                        reg_sample_13n <= sample_1n;
2560                        reg_sample_14n <= sample_2n;
2561                        reg_sample_15n <= sample_3n;
2562                        reg_sample_n12 <= sample_n0;
2563                        reg_sample_n13 <= sample_n1;
2564                        reg_sample_n14 <= sample_n2;
2565                        reg_sample_n15 <= sample_n3;
2566                     end
2567
2568                  5: begin
2569                        reg_sample_16n <= sample_0n;
2570                        reg_sample_17n <= sample_1n;
2571                        reg_sample_18n <= sample_2n;
2572                        reg_sample_19n <= sample_3n;
2573                        reg_sample_n16 <= sample_n0;
2574                        reg_sample_n17 <= sample_n1;
2575                        reg_sample_n18 <= sample_n2;
2576                        reg_sample_n19 <= sample_n3;
```

```
2577                    end
2578
2579                6: begin
2580                       reg_sample_20n <= sample_0n;
2581                       reg_sample_21n <= sample_1n;
2582                       reg_sample_22n <= sample_2n;
2583                       reg_sample_23n <= sample_3n;
2584                       reg_sample_n20 <= sample_n0;
2585                       reg_sample_n21 <= sample_n1;
2586                       reg_sample_n22 <= sample_n2;
2587                       reg_sample_n23 <= sample_n3;
2588                    end
2589
2590                7: begin
2591                       reg_sample_24n <= sample_0n;
2592                       reg_sample_25n <= sample_1n;
2593                       reg_sample_26n <= sample_2n;
2594                       reg_sample_27n <= sample_3n;
2595                       reg_sample_n24 <= sample_n0;
2596                       reg_sample_n25 <= sample_n1;
2597                       reg_sample_n26 <= sample_n2;
2598                       reg_sample_n27 <= sample_n3;
2599                    end
2600
2601                8: begin
2602                       reg_sample_28n <= sample_0n;
2603                       reg_sample_29n <= sample_1n;
2604                       reg_sample_30n <= sample_2n;
2605                       reg_sample_31n <= sample_3n;
2606                       reg_sample_n28 <= sample_n0;
2607                       reg_sample_n29 <= sample_n1;
2608                       reg_sample_n30 <= sample_n2;
2609                       reg_sample_n31 <= sample_n3;
2610                    end
2611
2612                9: begin
2613                       reg_sample_32n <= sample_0n;
2614                       reg_sample_33n <= sample_1n;
2615                       reg_sample_34n <= sample_2n;
2616                       reg_sample_35n <= sample_3n;
2617                       reg_sample_n32 <= sample_n0;
2618                       reg_sample_n33 <= sample_n1;
2619                       reg_sample_n34 <= sample_n2;
2620                       reg_sample_n35 <= sample_n3;
2621                    end
2622
2623                10:   begin
2624                       reg_sample_36n <= sample_0n;
2625                       reg_sample_37n <= sample_1n;
2626                       reg_sample_38n <= sample_2n;
2627                       reg_sample_39n <= sample_3n;
2628                       reg_sample_n36 <= sample_n0;
2629                       reg_sample_n37 <= sample_n1;
2630                       reg_sample_n38 <= sample_n2;
2631                       reg_sample_n39 <= sample_n3;
2632                    end
2633
```

```
2634                11:   begin
2635                      reg_sample_40n <= sample_0n;
2636                      reg_sample_41n <= sample_1n;
2637                      reg_sample_42n <= sample_2n;
2638                      reg_sample_43n <= sample_3n;
2639                      reg_sample_n40 <= sample_n0;
2640                      reg_sample_n41 <= sample_n1;
2641                      reg_sample_n42 <= sample_n2;
2642                      reg_sample_n43 <= sample_n3;
2643                   end
2644
2645                12:   begin
2646                      reg_sample_44n <= sample_0n;
2647                      reg_sample_45n <= sample_1n;
2648                      reg_sample_46n <= sample_2n;
2649                      reg_sample_47n <= sample_3n;
2650                      reg_sample_n44 <= sample_n0;
2651                      reg_sample_n45 <= sample_n1;
2652                      reg_sample_n46 <= sample_n2;
2653                      reg_sample_n47 <= sample_n3;
2654                   end
2655
2656                13:   begin
2657                      reg_sample_48n <= sample_0n;
2658                      reg_sample_49n <= sample_1n;
2659                      reg_sample_50n <= sample_2n;
2660                      reg_sample_51n <= sample_3n;
2661                      reg_sample_n48 <= sample_n0;
2662                      reg_sample_n49 <= sample_n1;
2663                      reg_sample_n50 <= sample_n2;
2664                      reg_sample_n51 <= sample_n3;
2665                   end
2666
2667                14:   begin
2668                      reg_sample_52n <= sample_0n;
2669                      reg_sample_53n <= sample_1n;
2670                      reg_sample_54n <= sample_2n;
2671                      reg_sample_55n <= sample_3n;
2672                      reg_sample_n52 <= sample_n0;
2673                      reg_sample_n53 <= sample_n1;
2674                      reg_sample_n54 <= sample_n2;
2675                      reg_sample_n55 <= sample_n3;
2676                   end
2677
2678                15:   begin
2679                      reg_sample_56n <= sample_0n;
2680                      reg_sample_57n <= sample_1n;
2681                      reg_sample_58n <= sample_2n;
2682                      reg_sample_59n <= sample_3n;
2683                      reg_sample_n56 <= sample_n0;
2684                      reg_sample_n57 <= sample_n1;
2685                      reg_sample_n58 <= sample_n2;
2686                      reg_sample_n59 <= sample_n3;
2687                   end
2688
2689                16:   begin
2690                      reg_sample_60n <= sample_0n;
```

```
                    reg_sample_61n <= sample_1n;
                    reg_sample_62n <= sample_2n;
                    reg_sample_63n <= sample_3n;
                    reg_sample_n60 <= sample_n0;
                    reg_sample_n61 <= sample_n1;
                    reg_sample_n62 <= sample_n2;
                    reg_sample_n63 <= sample_n3;
                end

            default: begin
                    reg_sample_0n <= 8'b00000000;
                    reg_sample_1n <= 8'b00000000;
                    reg_sample_2n <= 8'b00000000;
                    reg_sample_3n <= 8'b00000000;
                    reg_sample_n0 <= 8'b00000000;
                    reg_sample_n1 <= 8'b00000000;
                    reg_sample_n2 <= 8'b00000000;
                    reg_sample_n3 <= 8'b00000000;
                end
        endcase
    end

    if (i_index_enable)
    begin
        reg_i <= i;
    end

    case (planar_samples_x)
        0: begin
                reg_planar_sample_0n <= reg_sample_0n;
                reg_planar_sample_1n <= reg_sample_1n;
                reg_planar_sample_2n <= reg_sample_2n;
                reg_planar_sample_3n <= reg_sample_3n;
            end

        1: begin
                reg_planar_sample_0n <= reg_sample_4n;
                reg_planar_sample_1n <= reg_sample_5n;
                reg_planar_sample_2n <= reg_sample_6n;
                reg_planar_sample_3n <= reg_sample_7n;
            end

        2: begin
                reg_planar_sample_0n <= reg_sample_8n;
                reg_planar_sample_1n <= reg_sample_9n;
                reg_planar_sample_2n <= reg_sample_10n;
                reg_planar_sample_3n <= reg_sample_11n;
            end

        3: begin
                reg_planar_sample_0n <= reg_sample_12n;
                reg_planar_sample_1n <= reg_sample_13n;
                reg_planar_sample_2n <= reg_sample_14n;
                reg_planar_sample_3n <= reg_sample_15n;
            end

        4: begin
```

```
                reg_planar_sample_0n <= reg_sample_16n;
                reg_planar_sample_1n <= reg_sample_17n;
                reg_planar_sample_2n <= reg_sample_18n;
                reg_planar_sample_3n <= reg_sample_19n;
            end

        5: begin
                reg_planar_sample_0n <= reg_sample_20n;
                reg_planar_sample_1n <= reg_sample_21n;
                reg_planar_sample_2n <= reg_sample_22n;
                reg_planar_sample_3n <= reg_sample_23n;
            end

        6: begin
                reg_planar_sample_0n <= reg_sample_24n;
                reg_planar_sample_1n <= reg_sample_25n;
                reg_planar_sample_2n <= reg_sample_26n;
                reg_planar_sample_3n <= reg_sample_27n;
            end

        7: begin
                reg_planar_sample_0n <= reg_sample_28n;
                reg_planar_sample_1n <= reg_sample_29n;
                reg_planar_sample_2n <= reg_sample_30n;
                reg_planar_sample_3n <= reg_sample_31n;
            end

        default: begin
                reg_planar_sample_0n <= 8'b00000000;
                reg_planar_sample_1n <= 8'b00000000;
                reg_planar_sample_2n <= 8'b00000000;
                reg_planar_sample_3n <= 8'b00000000;
            end
    endcase

    case (planar_samples_y)
        0: begin
                reg_planar_sample_n0 <= reg_sample_n0;
                reg_planar_sample_n1 <= reg_sample_n1;
                reg_planar_sample_n2 <= reg_sample_n2;
                reg_planar_sample_n3 <= reg_sample_n3;
            end

        1: begin
                reg_planar_sample_n0 <= reg_sample_n4;
                reg_planar_sample_n1 <= reg_sample_n5;
                reg_planar_sample_n2 <= reg_sample_n6;
                reg_planar_sample_n3 <= reg_sample_n7;
            end

        2: begin
                reg_planar_sample_n0 <= reg_sample_n8;
                reg_planar_sample_n1 <= reg_sample_n9;
                reg_planar_sample_n2 <= reg_sample_n10;
                reg_planar_sample_n3 <= reg_sample_n11;
            end

```

```
2805                3: begin
2806                      reg_planar_sample_n0 <= reg_sample_n12;
2807                      reg_planar_sample_n1 <= reg_sample_n13;
2808                      reg_planar_sample_n2 <= reg_sample_n14;
2809                      reg_planar_sample_n3 <= reg_sample_n15;
2810                   end
2811
2812                4: begin
2813                      reg_planar_sample_n0 <= reg_sample_n16;
2814                      reg_planar_sample_n1 <= reg_sample_n17;
2815                      reg_planar_sample_n2 <= reg_sample_n18;
2816                      reg_planar_sample_n3 <= reg_sample_n19;
2817                   end
2818
2819                5: begin
2820                      reg_planar_sample_n0 <= reg_sample_n20;
2821                      reg_planar_sample_n1 <= reg_sample_n21;
2822                      reg_planar_sample_n2 <= reg_sample_n22;
2823                      reg_planar_sample_n3 <= reg_sample_n23;
2824                   end
2825
2826                6: begin
2827                      reg_planar_sample_n0 <= reg_sample_n24;
2828                      reg_planar_sample_n1 <= reg_sample_n25;
2829                      reg_planar_sample_n2 <= reg_sample_n26;
2830                      reg_planar_sample_n3 <= reg_sample_n27;
2831                   end
2832
2833                7: begin
2834                      reg_planar_sample_n0 <= reg_sample_n28;
2835                      reg_planar_sample_n1 <= reg_sample_n29;
2836                      reg_planar_sample_n2 <= reg_sample_n30;
2837                      reg_planar_sample_n3 <= reg_sample_n31;
2838                   end
2839
2840                default: begin
2841                      reg_planar_sample_n0 <= 8'b00000000;
2842                      reg_planar_sample_n1 <= 8'b00000000;
2843                      reg_planar_sample_n2 <= 8'b00000000;
2844                      reg_planar_sample_n3 <= 8'b00000000;
2845                   end
2846             endcase
2847
2848             case (dc_samples_x)
2849                0: begin
2850                      reg_dc_sample_0n <= reg_sample_0n;
2851                      reg_dc_sample_1n <= reg_sample_1n;
2852                      reg_dc_sample_2n <= reg_sample_2n;
2853                      reg_dc_sample_3n <= reg_sample_3n;
2854                   end
2855
2856                1: begin
2857                      reg_dc_sample_0n <= reg_sample_4n;
2858                      reg_dc_sample_1n <= reg_sample_5n;
2859                      reg_dc_sample_2n <= reg_sample_6n;
2860                      reg_dc_sample_3n <= reg_sample_7n;
2861                   end
```

```
2862
2863                2: begin
2864                      reg_dc_sample_0n <= reg_sample_8n;
2865                      reg_dc_sample_1n <= reg_sample_9n;
2866                      reg_dc_sample_2n <= reg_sample_10n;
2867                      reg_dc_sample_3n <= reg_sample_11n;
2868                   end
2869
2870                3: begin
2871                      reg_dc_sample_0n <= reg_sample_12n;
2872                      reg_dc_sample_1n <= reg_sample_13n;
2873                      reg_dc_sample_2n <= reg_sample_14n;
2874                      reg_dc_sample_3n <= reg_sample_15n;
2875                   end
2876
2877                4: begin
2878                      reg_dc_sample_0n <= reg_sample_16n;
2879                      reg_dc_sample_1n <= reg_sample_17n;
2880                      reg_dc_sample_2n <= reg_sample_18n;
2881                      reg_dc_sample_3n <= reg_sample_19n;
2882                   end
2883
2884                5: begin
2885                      reg_dc_sample_0n <= reg_sample_20n;
2886                      reg_dc_sample_1n <= reg_sample_21n;
2887                      reg_dc_sample_2n <= reg_sample_22n;
2888                      reg_dc_sample_3n <= reg_sample_23n;
2889                   end
2890
2891                6: begin
2892                      reg_dc_sample_0n <= reg_sample_24n;
2893                      reg_dc_sample_1n <= reg_sample_25n;
2894                      reg_dc_sample_2n <= reg_sample_26n;
2895                      reg_dc_sample_3n <= reg_sample_27n;
2896                   end
2897
2898                7: begin
2899                      reg_dc_sample_0n <= reg_sample_28n;
2900                      reg_dc_sample_1n <= reg_sample_29n;
2901                      reg_dc_sample_2n <= reg_sample_30n;
2902                      reg_dc_sample_3n <= reg_sample_31n;
2903                   end
2904
2905                default: begin
2906                      reg_dc_sample_0n <= 8'b00000000;
2907                      reg_dc_sample_1n <= 8'b00000000;
2908                      reg_dc_sample_2n <= 8'b00000000;
2909                      reg_dc_sample_3n <= 8'b00000000;
2910                   end
2911             endcase
2912
2913             case (dc_samples_y)
2914                0: begin
2915                      reg_dc_sample_n0 <= reg_sample_n0;
2916                      reg_dc_sample_n1 <= reg_sample_n1;
2917                      reg_dc_sample_n2 <= reg_sample_n2;
2918                      reg_dc_sample_n3 <= reg_sample_n3;
```

```
2919                end
2920
2921            1: begin
2922                   reg_dc_sample_n0 <= reg_sample_n4;
2923                   reg_dc_sample_n1 <= reg_sample_n5;
2924                   reg_dc_sample_n2 <= reg_sample_n6;
2925                   reg_dc_sample_n3 <= reg_sample_n7;
2926               end
2927
2928            2: begin
2929                   reg_dc_sample_n0 <= reg_sample_n8;
2930                   reg_dc_sample_n1 <= reg_sample_n9;
2931                   reg_dc_sample_n2 <= reg_sample_n10;
2932                   reg_dc_sample_n3 <= reg_sample_n11;
2933               end
2934
2935            3: begin
2936                   reg_dc_sample_n0 <= reg_sample_n12;
2937                   reg_dc_sample_n1 <= reg_sample_n13;
2938                   reg_dc_sample_n2 <= reg_sample_n14;
2939                   reg_dc_sample_n3 <= reg_sample_n15;
2940               end
2941
2942            4: begin
2943                   reg_dc_sample_n0 <= reg_sample_n16;
2944                   reg_dc_sample_n1 <= reg_sample_n17;
2945                   reg_dc_sample_n2 <= reg_sample_n18;
2946                   reg_dc_sample_n3 <= reg_sample_n19;
2947               end
2948
2949            5: begin
2950                   reg_dc_sample_n0 <= reg_sample_n20;
2951                   reg_dc_sample_n1 <= reg_sample_n21;
2952                   reg_dc_sample_n2 <= reg_sample_n22;
2953                   reg_dc_sample_n3 <= reg_sample_n23;
2954               end
2955
2956            6: begin
2957                   reg_dc_sample_n0 <= reg_sample_n24;
2958                   reg_dc_sample_n1 <= reg_sample_n25;
2959                   reg_dc_sample_n2 <= reg_sample_n26;
2960                   reg_dc_sample_n3 <= reg_sample_n27;
2961               end
2962
2963            7: begin
2964                   reg_dc_sample_n0 <= reg_sample_n28;
2965                   reg_dc_sample_n1 <= reg_sample_n29;
2966                   reg_dc_sample_n2 <= reg_sample_n30;
2967                   reg_dc_sample_n3 <= reg_sample_n31;
2968               end
2969
2970            default: begin
2971                   reg_dc_sample_n0 <= 8'b00000000;
2972                   reg_dc_sample_n1 <= 8'b00000000;
2973                   reg_dc_sample_n2 <= 8'b00000000;
2974                   reg_dc_sample_n3 <= 8'b00000000;
2975               end
```

```
2976            endcase
2977
2978            reg_angular_mode2_ref1 <= angular_mode2_sample1;
2979            reg_angular_mode2_ref2 <= angular_mode2_sample2;
2980            reg_angular_mode2_ref3 <= angular_mode2_sample3;
2981            reg_angular_mode2_ref4 <= angular_mode2_sample4;
2982            reg_angular_mode2_ref5 <= angular_mode2_sample5;
2983
2984            reg_angular_mode3_ref1 <= angular_mode3_sample1;
2985            reg_angular_mode3_ref2 <= angular_mode3_sample2;
2986            reg_angular_mode3_ref3 <= angular_mode3_sample3;
2987            reg_angular_mode3_ref4 <= angular_mode3_sample4;
2988            reg_angular_mode3_ref5 <= angular_mode3_sample5;
2989
2990            reg_angular_mode4_ref1 <= angular_mode4_sample1;
2991            reg_angular_mode4_ref2 <= angular_mode4_sample2;
2992            reg_angular_mode4_ref3 <= angular_mode4_sample3;
2993            reg_angular_mode4_ref4 <= angular_mode4_sample4;
2994            reg_angular_mode4_ref5 <= angular_mode4_sample5;
2995
2996            reg_angular_mode5_ref1 <= angular_mode5_sample1;
2997            reg_angular_mode5_ref2 <= angular_mode5_sample2;
2998            reg_angular_mode5_ref3 <= angular_mode5_sample3;
2999            reg_angular_mode5_ref4 <= angular_mode5_sample4;
3000            reg_angular_mode5_ref5 <= angular_mode5_sample5;
3001
3002            reg_angular_mode6_ref1 <= angular_mode6_sample1;
3003            reg_angular_mode6_ref2 <= angular_mode6_sample2;
3004            reg_angular_mode6_ref3 <= angular_mode6_sample3;
3005            reg_angular_mode6_ref4 <= angular_mode6_sample4;
3006            reg_angular_mode6_ref5 <= angular_mode6_sample5;
3007
3008            reg_angular_mode7_ref1 <= angular_mode7_sample1;
3009            reg_angular_mode7_ref2 <= angular_mode7_sample2;
3010            reg_angular_mode7_ref3 <= angular_mode7_sample3;
3011            reg_angular_mode7_ref4 <= angular_mode7_sample4;
3012            reg_angular_mode7_ref5 <= angular_mode7_sample5;
3013
3014            reg_angular_mode8_ref1 <= angular_mode8_sample1;
3015            reg_angular_mode8_ref2 <= angular_mode8_sample2;
3016            reg_angular_mode8_ref3 <= angular_mode8_sample3;
3017            reg_angular_mode8_ref4 <= angular_mode8_sample4;
3018            reg_angular_mode8_ref5 <= angular_mode8_sample5;
3019
3020            reg_angular_mode9_ref1 <= angular_mode9_sample1;
3021            reg_angular_mode9_ref2 <= angular_mode9_sample2;
3022            reg_angular_mode9_ref3 <= angular_mode9_sample3;
3023            reg_angular_mode9_ref4 <= angular_mode9_sample4;
3024            reg_angular_mode9_ref5 <= angular_mode9_sample5;
3025
3026            reg_angular_mode10_ref0 <= angular_mode10_sample0;
3027            reg_angular_mode10_ref1 <= angular_mode10_sample1;
3028            reg_angular_mode10_ref2 <= angular_mode10_sample2;
3029            reg_angular_mode10_ref3 <= angular_mode10_sample3;
3030            reg_angular_mode10_ref4 <= angular_mode10_sample4;
3031            reg_angular_mode10_ref5 <= angular_mode10_sample5;
3032
```

```
3033            reg_angular_mode11_ref1 <= angular_mode11_sample1;
3034            reg_angular_mode11_ref2 <= angular_mode11_sample2;
3035            reg_angular_mode11_ref3 <= angular_mode11_sample3;
3036            reg_angular_mode11_ref4 <= angular_mode11_sample4;
3037            reg_angular_mode11_ref5 <= angular_mode11_sample5;
3038
3039            reg_angular_mode12_ref1 <= angular_mode12_sample1;
3040            reg_angular_mode12_ref2 <= angular_mode12_sample2;
3041            reg_angular_mode12_ref3 <= angular_mode12_sample3;
3042            reg_angular_mode12_ref4 <= angular_mode12_sample4;
3043            reg_angular_mode12_ref5 <= angular_mode12_sample5;
3044
3045            reg_angular_mode13_ref1 <= angular_mode13_sample1;
3046            reg_angular_mode13_ref2 <= angular_mode13_sample2;
3047            reg_angular_mode13_ref3 <= angular_mode13_sample3;
3048            reg_angular_mode13_ref4 <= angular_mode13_sample4;
3049            reg_angular_mode13_ref5 <= angular_mode13_sample5;
3050
3051            reg_angular_mode14_ref1 <= angular_mode14_sample1;
3052            reg_angular_mode14_ref2 <= angular_mode14_sample2;
3053            reg_angular_mode14_ref3 <= angular_mode14_sample3;
3054            reg_angular_mode14_ref4 <= angular_mode14_sample4;
3055            reg_angular_mode14_ref5 <= angular_mode14_sample5;
3056
3057            reg_angular_mode15_ref1 <= angular_mode15_sample1;
3058            reg_angular_mode15_ref2 <= angular_mode15_sample2;
3059            reg_angular_mode15_ref3 <= angular_mode15_sample3;
3060            reg_angular_mode15_ref4 <= angular_mode15_sample4;
3061            reg_angular_mode15_ref5 <= angular_mode15_sample5;
3062
3063            reg_angular_mode16_ref1 <= angular_mode16_sample1;
3064            reg_angular_mode16_ref2 <= angular_mode16_sample2;
3065            reg_angular_mode16_ref3 <= angular_mode16_sample3;
3066            reg_angular_mode16_ref4 <= angular_mode16_sample4;
3067            reg_angular_mode16_ref5 <= angular_mode16_sample5;
3068
3069            reg_angular_mode17_ref1 <= angular_mode17_sample1;
3070            reg_angular_mode17_ref2 <= angular_mode17_sample2;
3071            reg_angular_mode17_ref3 <= angular_mode17_sample3;
3072            reg_angular_mode17_ref4 <= angular_mode17_sample4;
3073            reg_angular_mode17_ref5 <= angular_mode17_sample5;
3074
3075            reg_angular_mode18_ref1 <= angular_mode18_sample1;
3076            reg_angular_mode18_ref2 <= angular_mode18_sample2;
3077            reg_angular_mode18_ref3 <= angular_mode18_sample3;
3078            reg_angular_mode18_ref4 <= angular_mode18_sample4;
3079            reg_angular_mode18_ref5 <= angular_mode18_sample5;
3080
3081            reg_angular_mode19_ref1 <= angular_mode19_sample1;
3082            reg_angular_mode19_ref2 <= angular_mode19_sample2;
3083            reg_angular_mode19_ref3 <= angular_mode19_sample3;
3084            reg_angular_mode19_ref4 <= angular_mode19_sample4;
3085            reg_angular_mode19_ref5 <= angular_mode19_sample5;
3086
3087            reg_angular_mode20_ref1 <= angular_mode20_sample1;
3088            reg_angular_mode20_ref2 <= angular_mode20_sample2;
3089            reg_angular_mode20_ref3 <= angular_mode20_sample3;
```

```
3090            reg_angular_mode20_ref4 <= angular_mode20_sample4;
3091            reg_angular_mode20_ref5 <= angular_mode20_sample5;
3092
3093            reg_angular_mode21_ref1 <= angular_mode21_sample1;
3094            reg_angular_mode21_ref2 <= angular_mode21_sample2;
3095            reg_angular_mode21_ref3 <= angular_mode21_sample3;
3096            reg_angular_mode21_ref4 <= angular_mode21_sample4;
3097            reg_angular_mode21_ref5 <= angular_mode21_sample5;
3098
3099            reg_angular_mode22_ref1 <= angular_mode22_sample1;
3100            reg_angular_mode22_ref2 <= angular_mode22_sample2;
3101            reg_angular_mode22_ref3 <= angular_mode22_sample3;
3102            reg_angular_mode22_ref4 <= angular_mode22_sample4;
3103            reg_angular_mode22_ref5 <= angular_mode22_sample5;
3104
3105            reg_angular_mode23_ref1 <= angular_mode23_sample1;
3106            reg_angular_mode23_ref2 <= angular_mode23_sample2;
3107            reg_angular_mode23_ref3 <= angular_mode23_sample3;
3108            reg_angular_mode23_ref4 <= angular_mode23_sample4;
3109            reg_angular_mode23_ref5 <= angular_mode23_sample5;
3110
3111            reg_angular_mode24_ref1 <= angular_mode24_sample1;
3112            reg_angular_mode24_ref2 <= angular_mode24_sample2;
3113            reg_angular_mode24_ref3 <= angular_mode24_sample3;
3114            reg_angular_mode24_ref4 <= angular_mode24_sample4;
3115            reg_angular_mode24_ref5 <= angular_mode24_sample5;
3116
3117            reg_angular_mode25_ref1 <= angular_mode25_sample1;
3118            reg_angular_mode25_ref2 <= angular_mode25_sample2;
3119            reg_angular_mode25_ref3 <= angular_mode25_sample3;
3120            reg_angular_mode25_ref4 <= angular_mode25_sample4;
3121            reg_angular_mode25_ref5 <= angular_mode25_sample5;
3122
3123            reg_angular_mode26_ref0 <= angular_mode26_sample0;
3124            reg_angular_mode26_ref1 <= angular_mode26_sample1;
3125            reg_angular_mode26_ref2 <= angular_mode26_sample2;
3126            reg_angular_mode26_ref3 <= angular_mode26_sample3;
3127            reg_angular_mode26_ref4 <= angular_mode26_sample4;
3128            reg_angular_mode26_ref5 <= angular_mode26_sample5;
3129
3130            reg_angular_mode27_ref1 <= angular_mode27_sample1;
3131            reg_angular_mode27_ref2 <= angular_mode27_sample2;
3132            reg_angular_mode27_ref3 <= angular_mode27_sample3;
3133            reg_angular_mode27_ref4 <= angular_mode27_sample4;
3134            reg_angular_mode27_ref5 <= angular_mode27_sample5;
3135
3136            reg_angular_mode28_ref1 <= angular_mode28_sample1;
3137            reg_angular_mode28_ref2 <= angular_mode28_sample2;
3138            reg_angular_mode28_ref3 <= angular_mode28_sample3;
3139            reg_angular_mode28_ref4 <= angular_mode28_sample4;
3140            reg_angular_mode28_ref5 <= angular_mode28_sample5;
3141
3142            reg_angular_mode29_ref1 <= angular_mode29_sample1;
3143            reg_angular_mode29_ref2 <= angular_mode29_sample2;
3144            reg_angular_mode29_ref3 <= angular_mode29_sample3;
3145            reg_angular_mode29_ref4 <= angular_mode29_sample4;
3146            reg_angular_mode29_ref5 <= angular_mode29_sample5;
```

```
3147
3148            reg_angular_mode30_ref1 <= angular_mode30_sample1;
3149            reg_angular_mode30_ref2 <= angular_mode30_sample2;
3150            reg_angular_mode30_ref3 <= angular_mode30_sample3;
3151            reg_angular_mode30_ref4 <= angular_mode30_sample4;
3152            reg_angular_mode30_ref5 <= angular_mode30_sample5;
3153
3154            reg_angular_mode31_ref1 <= angular_mode31_sample1;
3155            reg_angular_mode31_ref2 <= angular_mode31_sample2;
3156            reg_angular_mode31_ref3 <= angular_mode31_sample3;
3157            reg_angular_mode31_ref4 <= angular_mode31_sample4;
3158            reg_angular_mode31_ref5 <= angular_mode31_sample5;
3159
3160            reg_angular_mode32_ref1 <= angular_mode32_sample1;
3161            reg_angular_mode32_ref2 <= angular_mode32_sample2;
3162            reg_angular_mode32_ref3 <= angular_mode32_sample3;
3163            reg_angular_mode32_ref4 <= angular_mode32_sample4;
3164            reg_angular_mode32_ref5 <= angular_mode32_sample5;
3165
3166            reg_angular_mode33_ref1 <= angular_mode33_sample1;
3167            reg_angular_mode33_ref2 <= angular_mode33_sample2;
3168            reg_angular_mode33_ref3 <= angular_mode33_sample3;
3169            reg_angular_mode33_ref4 <= angular_mode33_sample4;
3170            reg_angular_mode33_ref5 <= angular_mode33_sample5;
3171
3172            reg_angular_mode34_ref1 <= angular_mode34_sample1;
3173            reg_angular_mode34_ref2 <= angular_mode34_sample2;
3174            reg_angular_mode34_ref3 <= angular_mode34_sample3;
3175            reg_angular_mode34_ref4 <= angular_mode34_sample4;
3176            reg_angular_mode34_ref5 <= angular_mode34_sample5;
3177        end
3178    end
3179
3180    assign i              = reg_i + 1;
3181    assign i_condition          = (reg_i < reg_iterations) ? 1 : 0;
3182
3183    assign planar_sample_Nn     = reg_sample_Nn;
3184    assign planar_sample_nN     = reg_sample_nN;
3185
3186    assign planar_sample_0n     = reg_planar_sample_0n;
3187    assign planar_sample_1n     = reg_planar_sample_1n;
3188    assign planar_sample_2n     = reg_planar_sample_2n;
3189    assign planar_sample_3n     = reg_planar_sample_3n;
3190    assign planar_sample_n0     = reg_planar_sample_n0;
3191    assign planar_sample_n1     = reg_planar_sample_n1;
3192    assign planar_sample_n2     = reg_planar_sample_n2;
3193    assign planar_sample_n3     = reg_planar_sample_n3;
3194
3195    assign dc_sample_0n         = reg_dc_sample_0n;
3196    assign dc_sample_1n         = reg_dc_sample_1n;
3197    assign dc_sample_2n         = reg_dc_sample_2n;
3198    assign dc_sample_3n         = reg_dc_sample_3n;
3199    assign dc_sample_n0         = reg_dc_sample_n0;
3200    assign dc_sample_n1         = reg_dc_sample_n1;
3201    assign dc_sample_n2         = reg_dc_sample_n2;
3202    assign dc_sample_n3         = reg_dc_sample_n3;
3203
```

```
3204     assign angular_mode2_ref1  = reg_angular_mode2_ref1;
3205     assign angular_mode2_ref2  = reg_angular_mode2_ref2;
3206     assign angular_mode2_ref3  = reg_angular_mode2_ref3;
3207     assign angular_mode2_ref4  = reg_angular_mode2_ref4;
3208     assign angular_mode2_ref5  = reg_angular_mode2_ref5;
3209
3210     assign angular_mode3_ref1  = reg_angular_mode3_ref1;
3211     assign angular_mode3_ref2  = reg_angular_mode3_ref2;
3212     assign angular_mode3_ref3  = reg_angular_mode3_ref3;
3213     assign angular_mode3_ref4  = reg_angular_mode3_ref4;
3214     assign angular_mode3_ref5  = reg_angular_mode3_ref5;
3215
3216     assign angular_mode4_ref1  = reg_angular_mode4_ref1;
3217     assign angular_mode4_ref2  = reg_angular_mode4_ref2;
3218     assign angular_mode4_ref3  = reg_angular_mode4_ref3;
3219     assign angular_mode4_ref4  = reg_angular_mode4_ref4;
3220     assign angular_mode4_ref5  = reg_angular_mode4_ref5;
3221
3222     assign angular_mode5_ref1  = reg_angular_mode5_ref1;
3223     assign angular_mode5_ref2  = reg_angular_mode5_ref2;
3224     assign angular_mode5_ref3  = reg_angular_mode5_ref3;
3225     assign angular_mode5_ref4  = reg_angular_mode5_ref4;
3226     assign angular_mode5_ref5  = reg_angular_mode5_ref5;
3227
3228     assign angular_mode6_ref1  = reg_angular_mode6_ref1;
3229     assign angular_mode6_ref2  = reg_angular_mode6_ref2;
3230     assign angular_mode6_ref3  = reg_angular_mode6_ref3;
3231     assign angular_mode6_ref4  = reg_angular_mode6_ref4;
3232     assign angular_mode6_ref5  = reg_angular_mode6_ref5;
3233
3234     assign angular_mode7_ref1  = reg_angular_mode7_ref1;
3235     assign angular_mode7_ref2  = reg_angular_mode7_ref2;
3236     assign angular_mode7_ref3  = reg_angular_mode7_ref3;
3237     assign angular_mode7_ref4  = reg_angular_mode7_ref4;
3238     assign angular_mode7_ref5  = reg_angular_mode7_ref5;
3239
3240     assign angular_mode8_ref1  = reg_angular_mode8_ref1;
3241     assign angular_mode8_ref2  = reg_angular_mode8_ref2;
3242     assign angular_mode8_ref3  = reg_angular_mode8_ref3;
3243     assign angular_mode8_ref4  = reg_angular_mode8_ref4;
3244     assign angular_mode8_ref5  = reg_angular_mode8_ref5;
3245
3246     assign angular_mode9_ref1  = reg_angular_mode9_ref1;
3247     assign angular_mode9_ref2  = reg_angular_mode9_ref2;
3248     assign angular_mode9_ref3  = reg_angular_mode9_ref3;
3249     assign angular_mode9_ref4  = reg_angular_mode9_ref4;
3250     assign angular_mode9_ref5  = reg_angular_mode9_ref5;
3251
3252     assign angular_mode10_ref0 = reg_angular_mode10_ref0;
3253     assign angular_mode10_ref1 = reg_angular_mode10_ref1;
3254     assign angular_mode10_ref2 = reg_angular_mode10_ref2;
3255     assign angular_mode10_ref3 = reg_angular_mode10_ref3;
3256     assign angular_mode10_ref4 = reg_angular_mode10_ref4;
3257     assign angular_mode10_ref5 = reg_angular_mode10_ref5;
3258
3259     assign angular_mode11_ref1 = reg_angular_mode11_ref1;
3260     assign angular_mode11_ref2 = reg_angular_mode11_ref2;
```

```
3261     assign angular_mode11_ref3 = reg_angular_mode11_ref3;
3262     assign angular_mode11_ref4 = reg_angular_mode11_ref4;
3263     assign angular_mode11_ref5 = reg_angular_mode11_ref5;
3264
3265     assign angular_mode12_ref1 = reg_angular_mode12_ref1;
3266     assign angular_mode12_ref2 = reg_angular_mode12_ref2;
3267     assign angular_mode12_ref3 = reg_angular_mode12_ref3;
3268     assign angular_mode12_ref4 = reg_angular_mode12_ref4;
3269     assign angular_mode12_ref5 = reg_angular_mode12_ref5;
3270
3271     assign angular_mode13_ref1 = reg_angular_mode13_ref1;
3272     assign angular_mode13_ref2 = reg_angular_mode13_ref2;
3273     assign angular_mode13_ref3 = reg_angular_mode13_ref3;
3274     assign angular_mode13_ref4 = reg_angular_mode13_ref4;
3275     assign angular_mode13_ref5 = reg_angular_mode13_ref5;
3276
3277     assign angular_mode14_ref1 = reg_angular_mode14_ref1;
3278     assign angular_mode14_ref2 = reg_angular_mode14_ref2;
3279     assign angular_mode14_ref3 = reg_angular_mode14_ref3;
3280     assign angular_mode14_ref4 = reg_angular_mode14_ref4;
3281     assign angular_mode14_ref5 = reg_angular_mode14_ref5;
3282
3283     assign angular_mode15_ref1 = reg_angular_mode15_ref1;
3284     assign angular_mode15_ref2 = reg_angular_mode15_ref2;
3285     assign angular_mode15_ref3 = reg_angular_mode15_ref3;
3286     assign angular_mode15_ref4 = reg_angular_mode15_ref4;
3287     assign angular_mode15_ref5 = reg_angular_mode15_ref5;
3288
3289     assign angular_mode16_ref1 = reg_angular_mode16_ref1;
3290     assign angular_mode16_ref2 = reg_angular_mode16_ref2;
3291     assign angular_mode16_ref3 = reg_angular_mode16_ref3;
3292     assign angular_mode16_ref4 = reg_angular_mode16_ref4;
3293     assign angular_mode16_ref5 = reg_angular_mode16_ref5;
3294
3295     assign angular_mode17_ref1 = reg_angular_mode17_ref1;
3296     assign angular_mode17_ref2 = reg_angular_mode17_ref2;
3297     assign angular_mode17_ref3 = reg_angular_mode17_ref3;
3298     assign angular_mode17_ref4 = reg_angular_mode17_ref4;
3299     assign angular_mode17_ref5 = reg_angular_mode17_ref5;
3300
3301     assign angular_mode18_ref1 = reg_angular_mode18_ref1;
3302     assign angular_mode18_ref2 = reg_angular_mode18_ref2;
3303     assign angular_mode18_ref3 = reg_angular_mode18_ref3;
3304     assign angular_mode18_ref4 = reg_angular_mode18_ref4;
3305     assign angular_mode18_ref5 = reg_angular_mode18_ref5;
3306
3307     assign angular_mode19_ref1 = reg_angular_mode19_ref1;
3308     assign angular_mode19_ref2 = reg_angular_mode19_ref2;
3309     assign angular_mode19_ref3 = reg_angular_mode19_ref3;
3310     assign angular_mode19_ref4 = reg_angular_mode19_ref4;
3311     assign angular_mode19_ref5 = reg_angular_mode19_ref5;
3312
3313     assign angular_mode20_ref1 = reg_angular_mode20_ref1;
3314     assign angular_mode20_ref2 = reg_angular_mode20_ref2;
3315     assign angular_mode20_ref3 = reg_angular_mode20_ref3;
3316     assign angular_mode20_ref4 = reg_angular_mode20_ref4;
3317     assign angular_mode20_ref5 = reg_angular_mode20_ref5;
```

```
3318
3319     assign angular_mode21_ref1 = reg_angular_mode21_ref1;
3320     assign angular_mode21_ref2 = reg_angular_mode21_ref2;
3321     assign angular_mode21_ref3 = reg_angular_mode21_ref3;
3322     assign angular_mode21_ref4 = reg_angular_mode21_ref4;
3323     assign angular_mode21_ref5 = reg_angular_mode21_ref5;
3324
3325     assign angular_mode22_ref1 = reg_angular_mode22_ref1;
3326     assign angular_mode22_ref2 = reg_angular_mode22_ref2;
3327     assign angular_mode22_ref3 = reg_angular_mode22_ref3;
3328     assign angular_mode22_ref4 = reg_angular_mode22_ref4;
3329     assign angular_mode22_ref5 = reg_angular_mode22_ref5;
3330
3331     assign angular_mode23_ref1 = reg_angular_mode23_ref1;
3332     assign angular_mode23_ref2 = reg_angular_mode23_ref2;
3333     assign angular_mode23_ref3 = reg_angular_mode23_ref3;
3334     assign angular_mode23_ref4 = reg_angular_mode23_ref4;
3335     assign angular_mode23_ref5 = reg_angular_mode23_ref5;
3336
3337     assign angular_mode24_ref1 = reg_angular_mode24_ref1;
3338     assign angular_mode24_ref2 = reg_angular_mode24_ref2;
3339     assign angular_mode24_ref3 = reg_angular_mode24_ref3;
3340     assign angular_mode24_ref4 = reg_angular_mode24_ref4;
3341     assign angular_mode24_ref5 = reg_angular_mode24_ref5;
3342
3343     assign angular_mode25_ref1 = reg_angular_mode25_ref1;
3344     assign angular_mode25_ref2 = reg_angular_mode25_ref2;
3345     assign angular_mode25_ref3 = reg_angular_mode25_ref3;
3346     assign angular_mode25_ref4 = reg_angular_mode25_ref4;
3347     assign angular_mode25_ref5 = reg_angular_mode25_ref5;
3348
3349     assign angular_mode26_ref0 = reg_angular_mode26_ref0;
3350     assign angular_mode26_ref1 = reg_angular_mode26_ref1;
3351     assign angular_mode26_ref2 = reg_angular_mode26_ref2;
3352     assign angular_mode26_ref3 = reg_angular_mode26_ref3;
3353     assign angular_mode26_ref4 = reg_angular_mode26_ref4;
3354     assign angular_mode26_ref5 = reg_angular_mode26_ref5;
3355
3356     assign angular_mode27_ref1 = reg_angular_mode27_ref1;
3357     assign angular_mode27_ref2 = reg_angular_mode27_ref2;
3358     assign angular_mode27_ref3 = reg_angular_mode27_ref3;
3359     assign angular_mode27_ref4 = reg_angular_mode27_ref4;
3360     assign angular_mode27_ref5 = reg_angular_mode27_ref5;
3361
3362     assign angular_mode28_ref1 = reg_angular_mode28_ref1;
3363     assign angular_mode28_ref2 = reg_angular_mode28_ref2;
3364     assign angular_mode28_ref3 = reg_angular_mode28_ref3;
3365     assign angular_mode28_ref4 = reg_angular_mode28_ref4;
3366     assign angular_mode28_ref5 = reg_angular_mode28_ref5;
3367
3368     assign angular_mode29_ref1 = reg_angular_mode29_ref1;
3369     assign angular_mode29_ref2 = reg_angular_mode29_ref2;
3370     assign angular_mode29_ref3 = reg_angular_mode29_ref3;
3371     assign angular_mode29_ref4 = reg_angular_mode29_ref4;
3372     assign angular_mode29_ref5 = reg_angular_mode29_ref5;
3373
3374     assign angular_mode30_ref1 = reg_angular_mode30_ref1;
```

```
3375     assign angular_mode30_ref2 = reg_angular_mode30_ref2;
3376     assign angular_mode30_ref3 = reg_angular_mode30_ref3;
3377     assign angular_mode30_ref4 = reg_angular_mode30_ref4;
3378     assign angular_mode30_ref5 = reg_angular_mode30_ref5;
3379 
3380     assign angular_mode31_ref1 = reg_angular_mode31_ref1;
3381     assign angular_mode31_ref2 = reg_angular_mode31_ref2;
3382     assign angular_mode31_ref3 = reg_angular_mode31_ref3;
3383     assign angular_mode31_ref4 = reg_angular_mode31_ref4;
3384     assign angular_mode31_ref5 = reg_angular_mode31_ref5;
3385 
3386     assign angular_mode32_ref1 = reg_angular_mode32_ref1;
3387     assign angular_mode32_ref2 = reg_angular_mode32_ref2;
3388     assign angular_mode32_ref3 = reg_angular_mode32_ref3;
3389     assign angular_mode32_ref4 = reg_angular_mode32_ref4;
3390     assign angular_mode32_ref5 = reg_angular_mode32_ref5;
3391 
3392     assign angular_mode33_ref1 = reg_angular_mode33_ref1;
3393     assign angular_mode33_ref2 = reg_angular_mode33_ref2;
3394     assign angular_mode33_ref3 = reg_angular_mode33_ref3;
3395     assign angular_mode33_ref4 = reg_angular_mode33_ref4;
3396     assign angular_mode33_ref5 = reg_angular_mode33_ref5;
3397 
3398     assign angular_mode34_ref1 = reg_angular_mode34_ref1;
3399     assign angular_mode34_ref2 = reg_angular_mode34_ref2;
3400     assign angular_mode34_ref3 = reg_angular_mode34_ref3;
3401     assign angular_mode34_ref4 = reg_angular_mode34_ref4;
3402     assign angular_mode34_ref5 = reg_angular_mode34_ref5;
3403 
3404 endmodule
```

Listing B.23 – Código fonte da interface do gerenciador de amostras

```
1
2  module main_top_level
3
4     #(parameter WIDTH = 8)
5
6     (
7        input ack,
8              clock,
9              reset,
10             start,
11
12       input[WIDTH-1:0]  sample_0n,
13                         sample_1n,
14                         sample_2n,
15                         sample_3n,
16                         sample_n0,
17                         sample_n1,
18                         sample_n2,
19                         sample_n3,
20
21       input[WIDTH-1:0]  sample_Nn,
22                         sample_nN,
23                         sample_NN,
24
```

```
25          input[WIDTH-3:0]  N,
26
27          output   read_samples,
28                 read_def,
29                 done,
30                 check_planar_step,
31                 check_dc_step,
32                 check_angular_mode2_step,
33                 check_angular_mode3_step,
34                 check_angular_mode4_step,
35                 check_angular_mode5_step,
36                 check_angular_mode6_step,
37                 check_angular_mode7_step,
38                 check_angular_mode8_step,
39                 check_angular_mode9_step,
40                 check_angular_mode10_step,
41                 check_angular_mode11_step,
42                 check_angular_mode12_step,
43                 check_angular_mode13_step,
44                 check_angular_mode14_step,
45                 check_angular_mode15_step,
46                 check_angular_mode16_step,
47                 check_angular_mode17_step,
48                 check_angular_mode18_step,
49                 check_angular_mode19_step,
50                 check_angular_mode20_step,
51                 check_angular_mode21_step,
52                 check_angular_mode22_step,
53                 check_angular_mode23_step,
54                 check_angular_mode24_step,
55                 check_angular_mode25_step,
56                 check_angular_mode26_step,
57                 check_angular_mode27_step,
58                 check_angular_mode28_step,
59                 check_angular_mode29_step,
60                 check_angular_mode30_step,
61                 check_angular_mode31_step,
62                 check_angular_mode32_step,
63                 check_angular_mode33_step,
64                 check_angular_mode34_step,
65
66          output[WIDTH-1:0] planar_p00,
67                            planar_p01,
68                            planar_p02,
69                            planar_p03,
70                            planar_p10,
71                            planar_p11,
72                            planar_p12,
73                            planar_p13,
74                            planar_p20,
75                            planar_p21,
76                            planar_p22,
77                            planar_p23,
78                            planar_p30,
79                            planar_p31,
80                            planar_p32,
81                            planar_p33,
```

```
82
83                      dc_p00,
84                      dc_p01,
85                      dc_p02,
86                      dc_p03,
87                      dc_p10,
88                      dc_p11,
89                      dc_p12,
90                      dc_p13,
91                      dc_p20,
92                      dc_p21,
93                      dc_p22,
94                      dc_p23,
95                      dc_p30,
96                      dc_p31,
97                      dc_p32,
98                      dc_p33,
99
100                     angular_mode2_p00,
101                     angular_mode2_p01,
102                     angular_mode2_p02,
103                     angular_mode2_p03,
104                     angular_mode2_p10,
105                     angular_mode2_p11,
106                     angular_mode2_p12,
107                     angular_mode2_p13,
108                     angular_mode2_p20,
109                     angular_mode2_p21,
110                     angular_mode2_p22,
111                     angular_mode2_p23,
112                     angular_mode2_p30,
113                     angular_mode2_p31,
114                     angular_mode2_p32,
115                     angular_mode2_p33,
116
117                     angular_mode3_p00,
118                     angular_mode3_p01,
119                     angular_mode3_p02,
120                     angular_mode3_p03,
121                     angular_mode3_p10,
122                     angular_mode3_p11,
123                     angular_mode3_p12,
124                     angular_mode3_p13,
125                     angular_mode3_p20,
126                     angular_mode3_p21,
127                     angular_mode3_p22,
128                     angular_mode3_p23,
129                     angular_mode3_p30,
130                     angular_mode3_p31,
131                     angular_mode3_p32,
132                     angular_mode3_p33,
133
134                     angular_mode4_p00,
135                     angular_mode4_p01,
136                     angular_mode4_p02,
137                     angular_mode4_p03,
138                     angular_mode4_p10,
```

```
                    angular_mode4_p11,
                    angular_mode4_p12,
                    angular_mode4_p13,
                    angular_mode4_p20,
                    angular_mode4_p21,
                    angular_mode4_p22,
                    angular_mode4_p23,
                    angular_mode4_p30,
                    angular_mode4_p31,
                    angular_mode4_p32,
                    angular_mode4_p33,

                    angular_mode5_p00,
                    angular_mode5_p01,
                    angular_mode5_p02,
                    angular_mode5_p03,
                    angular_mode5_p10,
                    angular_mode5_p11,
                    angular_mode5_p12,
                    angular_mode5_p13,
                    angular_mode5_p20,
                    angular_mode5_p21,
                    angular_mode5_p22,
                    angular_mode5_p23,
                    angular_mode5_p30,
                    angular_mode5_p31,
                    angular_mode5_p32,
                    angular_mode5_p33,

                    angular_mode6_p00,
                    angular_mode6_p01,
                    angular_mode6_p02,
                    angular_mode6_p03,
                    angular_mode6_p10,
                    angular_mode6_p11,
                    angular_mode6_p12,
                    angular_mode6_p13,
                    angular_mode6_p20,
                    angular_mode6_p21,
                    angular_mode6_p22,
                    angular_mode6_p23,
                    angular_mode6_p30,
                    angular_mode6_p31,
                    angular_mode6_p32,
                    angular_mode6_p33,

                    angular_mode7_p00,
                    angular_mode7_p01,
                    angular_mode7_p02,
                    angular_mode7_p03,
                    angular_mode7_p10,
                    angular_mode7_p11,
                    angular_mode7_p12,
                    angular_mode7_p13,
                    angular_mode7_p20,
                    angular_mode7_p21,
                    angular_mode7_p22,
```

```
196                     angular_mode7_p23,
197                     angular_mode7_p30,
198                     angular_mode7_p31,
199                     angular_mode7_p32,
200                     angular_mode7_p33,
201
202                     angular_mode8_p00,
203                     angular_mode8_p01,
204                     angular_mode8_p02,
205                     angular_mode8_p03,
206                     angular_mode8_p10,
207                     angular_mode8_p11,
208                     angular_mode8_p12,
209                     angular_mode8_p13,
210                     angular_mode8_p20,
211                     angular_mode8_p21,
212                     angular_mode8_p22,
213                     angular_mode8_p23,
214                     angular_mode8_p30,
215                     angular_mode8_p31,
216                     angular_mode8_p32,
217                     angular_mode8_p33,
218
219                     angular_mode9_p00,
220                     angular_mode9_p01,
221                     angular_mode9_p02,
222                     angular_mode9_p03,
223                     angular_mode9_p10,
224                     angular_mode9_p11,
225                     angular_mode9_p12,
226                     angular_mode9_p13,
227                     angular_mode9_p20,
228                     angular_mode9_p21,
229                     angular_mode9_p22,
230                     angular_mode9_p23,
231                     angular_mode9_p30,
232                     angular_mode9_p31,
233                     angular_mode9_p32,
234                     angular_mode9_p33,
235
236                     angular_mode10_p00,
237                     angular_mode10_p01,
238                     angular_mode10_p02,
239                     angular_mode10_p03,
240                     angular_mode10_p10,
241                     angular_mode10_p11,
242                     angular_mode10_p12,
243                     angular_mode10_p13,
244                     angular_mode10_p20,
245                     angular_mode10_p21,
246                     angular_mode10_p22,
247                     angular_mode10_p23,
248                     angular_mode10_p30,
249                     angular_mode10_p31,
250                     angular_mode10_p32,
251                     angular_mode10_p33,
252
```

```
253                     angular_mode11_p00,
254                     angular_mode11_p01,
255                     angular_mode11_p02,
256                     angular_mode11_p03,
257                     angular_mode11_p10,
258                     angular_mode11_p11,
259                     angular_mode11_p12,
260                     angular_mode11_p13,
261                     angular_mode11_p20,
262                     angular_mode11_p21,
263                     angular_mode11_p22,
264                     angular_mode11_p23,
265                     angular_mode11_p30,
266                     angular_mode11_p31,
267                     angular_mode11_p32,
268                     angular_mode11_p33,
269
270                     angular_mode12_p00,
271                     angular_mode12_p01,
272                     angular_mode12_p02,
273                     angular_mode12_p03,
274                     angular_mode12_p10,
275                     angular_mode12_p11,
276                     angular_mode12_p12,
277                     angular_mode12_p13,
278                     angular_mode12_p20,
279                     angular_mode12_p21,
280                     angular_mode12_p22,
281                     angular_mode12_p23,
282                     angular_mode12_p30,
283                     angular_mode12_p31,
284                     angular_mode12_p32,
285                     angular_mode12_p33,
286
287                     angular_mode13_p00,
288                     angular_mode13_p01,
289                     angular_mode13_p02,
290                     angular_mode13_p03,
291                     angular_mode13_p10,
292                     angular_mode13_p11,
293                     angular_mode13_p12,
294                     angular_mode13_p13,
295                     angular_mode13_p20,
296                     angular_mode13_p21,
297                     angular_mode13_p22,
298                     angular_mode13_p23,
299                     angular_mode13_p30,
300                     angular_mode13_p31,
301                     angular_mode13_p32,
302                     angular_mode13_p33,
303
304                     angular_mode14_p00,
305                     angular_mode14_p01,
306                     angular_mode14_p02,
307                     angular_mode14_p03,
308                     angular_mode14_p10,
309                     angular_mode14_p11,
```

```
                    angular_mode14_p12,
                    angular_mode14_p13,
                    angular_mode14_p20,
                    angular_mode14_p21,
                    angular_mode14_p22,
                    angular_mode14_p23,
                    angular_mode14_p30,
                    angular_mode14_p31,
                    angular_mode14_p32,
                    angular_mode14_p33,

                    angular_mode15_p00,
                    angular_mode15_p01,
                    angular_mode15_p02,
                    angular_mode15_p03,
                    angular_mode15_p10,
                    angular_mode15_p11,
                    angular_mode15_p12,
                    angular_mode15_p13,
                    angular_mode15_p20,
                    angular_mode15_p21,
                    angular_mode15_p22,
                    angular_mode15_p23,
                    angular_mode15_p30,
                    angular_mode15_p31,
                    angular_mode15_p32,
                    angular_mode15_p33,

                    angular_mode16_p00,
                    angular_mode16_p01,
                    angular_mode16_p02,
                    angular_mode16_p03,
                    angular_mode16_p10,
                    angular_mode16_p11,
                    angular_mode16_p12,
                    angular_mode16_p13,
                    angular_mode16_p20,
                    angular_mode16_p21,
                    angular_mode16_p22,
                    angular_mode16_p23,
                    angular_mode16_p30,
                    angular_mode16_p31,
                    angular_mode16_p32,
                    angular_mode16_p33,

                    angular_mode17_p00,
                    angular_mode17_p01,
                    angular_mode17_p02,
                    angular_mode17_p03,
                    angular_mode17_p10,
                    angular_mode17_p11,
                    angular_mode17_p12,
                    angular_mode17_p13,
                    angular_mode17_p20,
                    angular_mode17_p21,
                    angular_mode17_p22,
                    angular_mode17_p23,
```

```
                    angular_mode17_p30,
                    angular_mode17_p31,
                    angular_mode17_p32,
                    angular_mode17_p33,

                    angular_mode18_p00,
                    angular_mode18_p01,
                    angular_mode18_p02,
                    angular_mode18_p03,
                    angular_mode18_p10,
                    angular_mode18_p11,
                    angular_mode18_p12,
                    angular_mode18_p13,
                    angular_mode18_p20,
                    angular_mode18_p21,
                    angular_mode18_p22,
                    angular_mode18_p23,
                    angular_mode18_p30,
                    angular_mode18_p31,
                    angular_mode18_p32,
                    angular_mode18_p33,

                    angular_mode19_p00,
                    angular_mode19_p01,
                    angular_mode19_p02,
                    angular_mode19_p03,
                    angular_mode19_p10,
                    angular_mode19_p11,
                    angular_mode19_p12,
                    angular_mode19_p13,
                    angular_mode19_p20,
                    angular_mode19_p21,
                    angular_mode19_p22,
                    angular_mode19_p23,
                    angular_mode19_p30,
                    angular_mode19_p31,
                    angular_mode19_p32,
                    angular_mode19_p33,

                    angular_mode20_p00,
                    angular_mode20_p01,
                    angular_mode20_p02,
                    angular_mode20_p03,
                    angular_mode20_p10,
                    angular_mode20_p11,
                    angular_mode20_p12,
                    angular_mode20_p13,
                    angular_mode20_p20,
                    angular_mode20_p21,
                    angular_mode20_p22,
                    angular_mode20_p23,
                    angular_mode20_p30,
                    angular_mode20_p31,
                    angular_mode20_p32,
                    angular_mode20_p33,

                    angular_mode21_p00,
```

```
                    angular_mode21_p01,
                    angular_mode21_p02,
                    angular_mode21_p03,
                    angular_mode21_p10,
                    angular_mode21_p11,
                    angular_mode21_p12,
                    angular_mode21_p13,
                    angular_mode21_p20,
                    angular_mode21_p21,
                    angular_mode21_p22,
                    angular_mode21_p23,
                    angular_mode21_p30,
                    angular_mode21_p31,
                    angular_mode21_p32,
                    angular_mode21_p33,

                    angular_mode22_p00,
                    angular_mode22_p01,
                    angular_mode22_p02,
                    angular_mode22_p03,
                    angular_mode22_p10,
                    angular_mode22_p11,
                    angular_mode22_p12,
                    angular_mode22_p13,
                    angular_mode22_p20,
                    angular_mode22_p21,
                    angular_mode22_p22,
                    angular_mode22_p23,
                    angular_mode22_p30,
                    angular_mode22_p31,
                    angular_mode22_p32,
                    angular_mode22_p33,

                    angular_mode23_p00,
                    angular_mode23_p01,
                    angular_mode23_p02,
                    angular_mode23_p03,
                    angular_mode23_p10,
                    angular_mode23_p11,
                    angular_mode23_p12,
                    angular_mode23_p13,
                    angular_mode23_p20,
                    angular_mode23_p21,
                    angular_mode23_p22,
                    angular_mode23_p23,
                    angular_mode23_p30,
                    angular_mode23_p31,
                    angular_mode23_p32,
                    angular_mode23_p33,

                    angular_mode24_p00,
                    angular_mode24_p01,
                    angular_mode24_p02,
                    angular_mode24_p03,
                    angular_mode24_p10,
                    angular_mode24_p11,
                    angular_mode24_p12,
```

```
                    angular_mode24_p13,
                    angular_mode24_p20,
                    angular_mode24_p21,
                    angular_mode24_p22,
                    angular_mode24_p23,
                    angular_mode24_p30,
                    angular_mode24_p31,
                    angular_mode24_p32,
                    angular_mode24_p33,

                    angular_mode25_p00,
                    angular_mode25_p01,
                    angular_mode25_p02,
                    angular_mode25_p03,
                    angular_mode25_p10,
                    angular_mode25_p11,
                    angular_mode25_p12,
                    angular_mode25_p13,
                    angular_mode25_p20,
                    angular_mode25_p21,
                    angular_mode25_p22,
                    angular_mode25_p23,
                    angular_mode25_p30,
                    angular_mode25_p31,
                    angular_mode25_p32,
                    angular_mode25_p33,

                    angular_mode26_p00,
                    angular_mode26_p01,
                    angular_mode26_p02,
                    angular_mode26_p03,
                    angular_mode26_p10,
                    angular_mode26_p11,
                    angular_mode26_p12,
                    angular_mode26_p13,
                    angular_mode26_p20,
                    angular_mode26_p21,
                    angular_mode26_p22,
                    angular_mode26_p23,
                    angular_mode26_p30,
                    angular_mode26_p31,
                    angular_mode26_p32,
                    angular_mode26_p33,

                    angular_mode27_p00,
                    angular_mode27_p01,
                    angular_mode27_p02,
                    angular_mode27_p03,
                    angular_mode27_p10,
                    angular_mode27_p11,
                    angular_mode27_p12,
                    angular_mode27_p13,
                    angular_mode27_p20,
                    angular_mode27_p21,
                    angular_mode27_p22,
                    angular_mode27_p23,
                    angular_mode27_p30,
```

```
                    angular_mode27_p31,
                    angular_mode27_p32,
                    angular_mode27_p33,

                    angular_mode28_p00,
                    angular_mode28_p01,
                    angular_mode28_p02,
                    angular_mode28_p03,
                    angular_mode28_p10,
                    angular_mode28_p11,
                    angular_mode28_p12,
                    angular_mode28_p13,
                    angular_mode28_p20,
                    angular_mode28_p21,
                    angular_mode28_p22,
                    angular_mode28_p23,
                    angular_mode28_p30,
                    angular_mode28_p31,
                    angular_mode28_p32,
                    angular_mode28_p33,

                    angular_mode29_p00,
                    angular_mode29_p01,
                    angular_mode29_p02,
                    angular_mode29_p03,
                    angular_mode29_p10,
                    angular_mode29_p11,
                    angular_mode29_p12,
                    angular_mode29_p13,
                    angular_mode29_p20,
                    angular_mode29_p21,
                    angular_mode29_p22,
                    angular_mode29_p23,
                    angular_mode29_p30,
                    angular_mode29_p31,
                    angular_mode29_p32,
                    angular_mode29_p33,

                    angular_mode30_p00,
                    angular_mode30_p01,
                    angular_mode30_p02,
                    angular_mode30_p03,
                    angular_mode30_p10,
                    angular_mode30_p11,
                    angular_mode30_p12,
                    angular_mode30_p13,
                    angular_mode30_p20,
                    angular_mode30_p21,
                    angular_mode30_p22,
                    angular_mode30_p23,
                    angular_mode30_p30,
                    angular_mode30_p31,
                    angular_mode30_p32,
                    angular_mode30_p33,

                    angular_mode31_p00,
                    angular_mode31_p01,
```

```
595                     angular_mode31_p02,
596                     angular_mode31_p03,
597                     angular_mode31_p10,
598                     angular_mode31_p11,
599                     angular_mode31_p12,
600                     angular_mode31_p13,
601                     angular_mode31_p20,
602                     angular_mode31_p21,
603                     angular_mode31_p22,
604                     angular_mode31_p23,
605                     angular_mode31_p30,
606                     angular_mode31_p31,
607                     angular_mode31_p32,
608                     angular_mode31_p33,
609
610                     angular_mode32_p00,
611                     angular_mode32_p01,
612                     angular_mode32_p02,
613                     angular_mode32_p03,
614                     angular_mode32_p10,
615                     angular_mode32_p11,
616                     angular_mode32_p12,
617                     angular_mode32_p13,
618                     angular_mode32_p20,
619                     angular_mode32_p21,
620                     angular_mode32_p22,
621                     angular_mode32_p23,
622                     angular_mode32_p30,
623                     angular_mode32_p31,
624                     angular_mode32_p32,
625                     angular_mode32_p33,
626
627                     angular_mode33_p00,
628                     angular_mode33_p01,
629                     angular_mode33_p02,
630                     angular_mode33_p03,
631                     angular_mode33_p10,
632                     angular_mode33_p11,
633                     angular_mode33_p12,
634                     angular_mode33_p13,
635                     angular_mode33_p20,
636                     angular_mode33_p21,
637                     angular_mode33_p22,
638                     angular_mode33_p23,
639                     angular_mode33_p30,
640                     angular_mode33_p31,
641                     angular_mode33_p32,
642                     angular_mode33_p33,
643
644                     angular_mode34_p00,
645                     angular_mode34_p01,
646                     angular_mode34_p02,
647                     angular_mode34_p03,
648                     angular_mode34_p10,
649                     angular_mode34_p11,
650                     angular_mode34_p12,
651                     angular_mode34_p13,
```

```
652                     angular_mode34_p20,
653                     angular_mode34_p21,
654                     angular_mode34_p22,
655                     angular_mode34_p23,
656                     angular_mode34_p30,
657                     angular_mode34_p31,
658                     angular_mode34_p32,
659                     angular_mode34_p33
660     );
661
662     wire  i_condition,
663           def_enable,
664           i_index_enable,
665           samples_enable,
666           reset_registers,
667           start_predictors,
668           planar_done,
669           dc_done,
670           angular_mode2_done,
671           angular_mode3_done,
672           angular_mode4_done,
673           angular_mode5_done,
674           angular_mode6_done,
675           angular_mode7_done,
676           angular_mode8_done,
677           angular_mode9_done,
678           angular_mode10_done,
679           angular_mode11_done,
680           angular_mode12_done,
681           angular_mode13_done,
682           angular_mode14_done,
683           angular_mode15_done,
684           angular_mode16_done,
685           angular_mode17_done,
686           angular_mode18_done,
687           angular_mode19_done,
688           angular_mode20_done,
689           angular_mode21_done,
690           angular_mode22_done,
691           angular_mode23_done,
692           angular_mode24_done,
693           angular_mode25_done,
694           angular_mode26_done,
695           angular_mode27_done,
696           angular_mode28_done,
697           angular_mode29_done,
698           angular_mode30_done,
699           angular_mode31_done,
700           angular_mode32_done,
701           angular_mode33_done,
702           angular_mode34_done;
703
704     wire[WIDTH-6:0]   planar_samples_x,
705                 planar_samples_y,
706                 dc_samples_x,
707                 dc_samples_y;
708
```

```
709     wire[WIDTH-2:0] filter_samples_mode10,
710                     filter_samples_mode26;
711
712     wire signed[WIDTH-1:0]  angular_mode2_samples,
713                             angular_mode3_samples,
714                             angular_mode4_samples,
715                             angular_mode5_samples,
716                             angular_mode6_samples,
717                             angular_mode7_samples,
718                             angular_mode8_samples,
719                             angular_mode9_samples,
720                             angular_mode10_samples,
721                             angular_mode11_samples,
722                             angular_mode12_samples,
723                             angular_mode13_samples,
724                             angular_mode14_samples,
725                             angular_mode15_samples,
726                             angular_mode16_samples,
727                             angular_mode17_samples,
728                             angular_mode18_samples,
729                             angular_mode19_samples,
730                             angular_mode20_samples,
731                             angular_mode21_samples,
732                             angular_mode22_samples,
733                             angular_mode23_samples,
734                             angular_mode24_samples,
735                             angular_mode25_samples,
736                             angular_mode26_samples,
737                             angular_mode27_samples,
738                             angular_mode28_samples,
739                             angular_mode29_samples,
740                             angular_mode30_samples,
741                             angular_mode31_samples,
742                             angular_mode32_samples,
743                             angular_mode33_samples,
744                             angular_mode34_samples;
745
746     wire[WIDTH-1:0]   planar_sample_0n,
747                       planar_sample_1n,
748                       planar_sample_2n,
749                       planar_sample_3n,
750                       planar_sample_Nn,
751                       planar_sample_n0,
752                       planar_sample_n1,
753                       planar_sample_n2,
754                       planar_sample_n3,
755                       planar_sample_nN,
756
757                       dc_sample_0n,
758                       dc_sample_1n,
759                       dc_sample_2n,
760                       dc_sample_3n,
761                       dc_sample_n0,
762                       dc_sample_n1,
763                       dc_sample_n2,
764                       dc_sample_n3,
765
```

```
                angular_mode2_ref1,
                angular_mode2_ref2,
                angular_mode2_ref3,
                angular_mode2_ref4,
                angular_mode2_ref5,

                angular_mode3_ref1,
                angular_mode3_ref2,
                angular_mode3_ref3,
                angular_mode3_ref4,
                angular_mode3_ref5,

                angular_mode4_ref1,
                angular_mode4_ref2,
                angular_mode4_ref3,
                angular_mode4_ref4,
                angular_mode4_ref5,

                angular_mode5_ref1,
                angular_mode5_ref2,
                angular_mode5_ref3,
                angular_mode5_ref4,
                angular_mode5_ref5,

                angular_mode6_ref1,
                angular_mode6_ref2,
                angular_mode6_ref3,
                angular_mode6_ref4,
                angular_mode6_ref5,

                angular_mode7_ref1,
                angular_mode7_ref2,
                angular_mode7_ref3,
                angular_mode7_ref4,
                angular_mode7_ref5,

                angular_mode8_ref1,
                angular_mode8_ref2,
                angular_mode8_ref3,
                angular_mode8_ref4,
                angular_mode8_ref5,

                angular_mode9_ref1,
                angular_mode9_ref2,
                angular_mode9_ref3,
                angular_mode9_ref4,
                angular_mode9_ref5,

                angular_mode10_ref0,
                angular_mode10_ref1,
                angular_mode10_ref2,
                angular_mode10_ref3,
                angular_mode10_ref4,
                angular_mode10_ref5,

                angular_mode11_ref1,
                angular_mode11_ref2,
```

```
823                 angular_mode11_ref3,
824                 angular_mode11_ref4,
825                 angular_mode11_ref5,
826
827                 angular_mode12_ref1,
828                 angular_mode12_ref2,
829                 angular_mode12_ref3,
830                 angular_mode12_ref4,
831                 angular_mode12_ref5,
832
833                 angular_mode13_ref1,
834                 angular_mode13_ref2,
835                 angular_mode13_ref3,
836                 angular_mode13_ref4,
837                 angular_mode13_ref5,
838
839                 angular_mode14_ref1,
840                 angular_mode14_ref2,
841                 angular_mode14_ref3,
842                 angular_mode14_ref4,
843                 angular_mode14_ref5,
844
845                 angular_mode15_ref1,
846                 angular_mode15_ref2,
847                 angular_mode15_ref3,
848                 angular_mode15_ref4,
849                 angular_mode15_ref5,
850
851                 angular_mode16_ref1,
852                 angular_mode16_ref2,
853                 angular_mode16_ref3,
854                 angular_mode16_ref4,
855                 angular_mode16_ref5,
856
857                 angular_mode17_ref1,
858                 angular_mode17_ref2,
859                 angular_mode17_ref3,
860                 angular_mode17_ref4,
861                 angular_mode17_ref5,
862
863                 angular_mode18_ref1,
864                 angular_mode18_ref2,
865                 angular_mode18_ref3,
866                 angular_mode18_ref4,
867                 angular_mode18_ref5,
868
869                 angular_mode19_ref1,
870                 angular_mode19_ref2,
871                 angular_mode19_ref3,
872                 angular_mode19_ref4,
873                 angular_mode19_ref5,
874
875                 angular_mode20_ref1,
876                 angular_mode20_ref2,
877                 angular_mode20_ref3,
878                 angular_mode20_ref4,
879                 angular_mode20_ref5,
```

```
880
881                 angular_mode21_ref1,
882                 angular_mode21_ref2,
883                 angular_mode21_ref3,
884                 angular_mode21_ref4,
885                 angular_mode21_ref5,
886
887                 angular_mode22_ref1,
888                 angular_mode22_ref2,
889                 angular_mode22_ref3,
890                 angular_mode22_ref4,
891                 angular_mode22_ref5,
892
893                 angular_mode23_ref1,
894                 angular_mode23_ref2,
895                 angular_mode23_ref3,
896                 angular_mode23_ref4,
897                 angular_mode23_ref5,
898
899                 angular_mode24_ref1,
900                 angular_mode24_ref2,
901                 angular_mode24_ref3,
902                 angular_mode24_ref4,
903                 angular_mode24_ref5,
904
905                 angular_mode25_ref1,
906                 angular_mode25_ref2,
907                 angular_mode25_ref3,
908                 angular_mode25_ref4,
909                 angular_mode25_ref5,
910
911                 angular_mode26_ref0,
912                 angular_mode26_ref1,
913                 angular_mode26_ref2,
914                 angular_mode26_ref3,
915                 angular_mode26_ref4,
916                 angular_mode26_ref5,
917
918                 angular_mode27_ref1,
919                 angular_mode27_ref2,
920                 angular_mode27_ref3,
921                 angular_mode27_ref4,
922                 angular_mode27_ref5,
923
924                 angular_mode28_ref1,
925                 angular_mode28_ref2,
926                 angular_mode28_ref3,
927                 angular_mode28_ref4,
928                 angular_mode28_ref5,
929
930                 angular_mode29_ref1,
931                 angular_mode29_ref2,
932                 angular_mode29_ref3,
933                 angular_mode29_ref4,
934                 angular_mode29_ref5,
935
936                 angular_mode30_ref1,
```

```
937                 angular_mode30_ref2,
938                 angular_mode30_ref3,
939                 angular_mode30_ref4,
940                 angular_mode30_ref5,
941
942                 angular_mode31_ref1,
943                 angular_mode31_ref2,
944                 angular_mode31_ref3,
945                 angular_mode31_ref4,
946                 angular_mode31_ref5,
947
948                 angular_mode32_ref1,
949                 angular_mode32_ref2,
950                 angular_mode32_ref3,
951                 angular_mode32_ref4,
952                 angular_mode32_ref5,
953
954                 angular_mode33_ref1,
955                 angular_mode33_ref2,
956                 angular_mode33_ref3,
957                 angular_mode33_ref4,
958                 angular_mode33_ref5,
959
960                 angular_mode34_ref1,
961                 angular_mode34_ref2,
962                 angular_mode34_ref3,
963                 angular_mode34_ref4,
964                 angular_mode34_ref5;
965
966     main_control main_control_block(
967        .ack(ack),
968        .clock(clock),
969        .reset(reset),
970        .start(start),
971        .i_condition(i_condition),
972        .planar_done(planar_done),
973        .dc_done(dc_done),
974        .angular_mode2_done(angular_mode2_done),
975        .angular_mode3_done(angular_mode3_done),
976        .angular_mode4_done(angular_mode4_done),
977        .angular_mode5_done(angular_mode5_done),
978        .angular_mode6_done(angular_mode6_done),
979        .angular_mode7_done(angular_mode7_done),
980        .angular_mode8_done(angular_mode8_done),
981        .angular_mode9_done(angular_mode9_done),
982        .angular_mode10_done(angular_mode10_done),
983        .angular_mode11_done(angular_mode11_done),
984        .angular_mode12_done(angular_mode12_done),
985        .angular_mode13_done(angular_mode13_done),
986        .angular_mode14_done(angular_mode14_done),
987        .angular_mode15_done(angular_mode15_done),
988        .angular_mode16_done(angular_mode16_done),
989        .angular_mode17_done(angular_mode17_done),
990        .angular_mode18_done(angular_mode18_done),
991        .angular_mode19_done(angular_mode19_done),
992        .angular_mode20_done(angular_mode20_done),
993        .angular_mode21_done(angular_mode21_done),
```

```
994         .angular_mode22_done(angular_mode22_done),
995         .angular_mode23_done(angular_mode23_done),
996         .angular_mode24_done(angular_mode24_done),
997         .angular_mode25_done(angular_mode25_done),
998         .angular_mode26_done(angular_mode26_done),
999         .angular_mode27_done(angular_mode27_done),
1000        .angular_mode28_done(angular_mode28_done),
1001        .angular_mode29_done(angular_mode29_done),
1002        .angular_mode30_done(angular_mode30_done),
1003        .angular_mode31_done(angular_mode31_done),
1004        .angular_mode32_done(angular_mode32_done),
1005        .angular_mode33_done(angular_mode33_done),
1006        .angular_mode34_done(angular_mode34_done),
1007
1008        .samples_request(read_samples),
1009        .def_enable(def_enable),
1010        .i_index_enable(i_index_enable),
1011        .samples_enable(samples_enable),
1012        .reset_registers(reset_registers),
1013        .start_predictors(start_predictors),
1014        .done(done)
1015    );
1016
1017    main_operative #(WIDTH) main_operative_block(
1018        .clock(clock),
1019        .reset(reset_registers),
1020        .def_enable(def_enable),
1021        .samples_enable(samples_enable),
1022        .i_index_enable(i_index_enable),
1023
1024        .planar_samples_x(planar_samples_x),
1025        .planar_samples_y(planar_samples_y),
1026
1027        .dc_samples_x(dc_samples_x),
1028        .dc_samples_y(dc_samples_y),
1029
1030        .filter_samples_mode10(filter_samples_mode10),
1031        .filter_samples_mode26(filter_samples_mode26),
1032
1033        .angular_mode2_samples(angular_mode2_samples),
1034        .angular_mode3_samples(angular_mode3_samples),
1035        .angular_mode4_samples(angular_mode4_samples),
1036        .angular_mode5_samples(angular_mode5_samples),
1037        .angular_mode6_samples(angular_mode6_samples),
1038        .angular_mode7_samples(angular_mode7_samples),
1039        .angular_mode8_samples(angular_mode8_samples),
1040        .angular_mode9_samples(angular_mode9_samples),
1041        .angular_mode10_samples(angular_mode10_samples),
1042        .angular_mode11_samples(angular_mode11_samples),
1043        .angular_mode12_samples(angular_mode12_samples),
1044        .angular_mode13_samples(angular_mode13_samples),
1045        .angular_mode14_samples(angular_mode14_samples),
1046        .angular_mode15_samples(angular_mode15_samples),
1047        .angular_mode16_samples(angular_mode16_samples),
1048        .angular_mode17_samples(angular_mode17_samples),
1049        .angular_mode18_samples(angular_mode18_samples),
1050        .angular_mode19_samples(angular_mode19_samples),
```

```
        .angular_mode20_samples(angular_mode20_samples),
        .angular_mode21_samples(angular_mode21_samples),
        .angular_mode22_samples(angular_mode22_samples),
        .angular_mode23_samples(angular_mode23_samples),
        .angular_mode24_samples(angular_mode24_samples),
        .angular_mode25_samples(angular_mode25_samples),
        .angular_mode26_samples(angular_mode26_samples),
        .angular_mode27_samples(angular_mode27_samples),
        .angular_mode28_samples(angular_mode28_samples),
        .angular_mode29_samples(angular_mode29_samples),
        .angular_mode30_samples(angular_mode30_samples),
        .angular_mode31_samples(angular_mode31_samples),
        .angular_mode32_samples(angular_mode32_samples),
        .angular_mode33_samples(angular_mode33_samples),
        .angular_mode34_samples(angular_mode34_samples),

        .sample_0n(sample_0n),
        .sample_1n(sample_1n),
        .sample_2n(sample_2n),
        .sample_3n(sample_3n),
        .sample_n0(sample_n0),
        .sample_n1(sample_n1),
        .sample_n2(sample_n2),
        .sample_n3(sample_n3),
        .sample_Nn(sample_Nn),
        .sample_nN(sample_nN),
        .sample_NN(sample_NN),
        .N(N),

        .i_condition(i_condition),

        .planar_sample_0n(planar_sample_0n),
        .planar_sample_1n(planar_sample_1n),
        .planar_sample_2n(planar_sample_2n),
        .planar_sample_3n(planar_sample_3n),
        .planar_sample_Nn(planar_sample_Nn),
        .planar_sample_n0(planar_sample_n0),
        .planar_sample_n1(planar_sample_n1),
        .planar_sample_n2(planar_sample_n2),
        .planar_sample_n3(planar_sample_n3),
        .planar_sample_nN(planar_sample_nN),

        .dc_sample_0n(dc_sample_0n),
        .dc_sample_1n(dc_sample_1n),
        .dc_sample_2n(dc_sample_2n),
        .dc_sample_3n(dc_sample_3n),
        .dc_sample_n0(dc_sample_n0),
        .dc_sample_n1(dc_sample_n1),
        .dc_sample_n2(dc_sample_n2),
        .dc_sample_n3(dc_sample_n3),

        .angular_mode2_ref1(angular_mode2_ref1),
        .angular_mode2_ref2(angular_mode2_ref2),
        .angular_mode2_ref3(angular_mode2_ref3),
        .angular_mode2_ref4(angular_mode2_ref4),
        .angular_mode2_ref5(angular_mode2_ref5),

```

```
        .angular_mode3_ref1(angular_mode3_ref1),
        .angular_mode3_ref2(angular_mode3_ref2),
        .angular_mode3_ref3(angular_mode3_ref3),
        .angular_mode3_ref4(angular_mode3_ref4),
        .angular_mode3_ref5(angular_mode3_ref5),

        .angular_mode4_ref1(angular_mode4_ref1),
        .angular_mode4_ref2(angular_mode4_ref2),
        .angular_mode4_ref3(angular_mode4_ref3),
        .angular_mode4_ref4(angular_mode4_ref4),
        .angular_mode4_ref5(angular_mode4_ref5),

        .angular_mode5_ref1(angular_mode5_ref1),
        .angular_mode5_ref2(angular_mode5_ref2),
        .angular_mode5_ref3(angular_mode5_ref3),
        .angular_mode5_ref4(angular_mode5_ref4),
        .angular_mode5_ref5(angular_mode5_ref5),

        .angular_mode6_ref1(angular_mode6_ref1),
        .angular_mode6_ref2(angular_mode6_ref2),
        .angular_mode6_ref3(angular_mode6_ref3),
        .angular_mode6_ref4(angular_mode6_ref4),
        .angular_mode6_ref5(angular_mode6_ref5),

        .angular_mode7_ref1(angular_mode7_ref1),
        .angular_mode7_ref2(angular_mode7_ref2),
        .angular_mode7_ref3(angular_mode7_ref3),
        .angular_mode7_ref4(angular_mode7_ref4),
        .angular_mode7_ref5(angular_mode7_ref5),

        .angular_mode8_ref1(angular_mode8_ref1),
        .angular_mode8_ref2(angular_mode8_ref2),
        .angular_mode8_ref3(angular_mode8_ref3),
        .angular_mode8_ref4(angular_mode8_ref4),
        .angular_mode8_ref5(angular_mode8_ref5),

        .angular_mode9_ref1(angular_mode9_ref1),
        .angular_mode9_ref2(angular_mode9_ref2),
        .angular_mode9_ref3(angular_mode9_ref3),
        .angular_mode9_ref4(angular_mode9_ref4),
        .angular_mode9_ref5(angular_mode9_ref5),

        .angular_mode10_ref0(angular_mode10_ref0),
        .angular_mode10_ref1(angular_mode10_ref1),
        .angular_mode10_ref2(angular_mode10_ref2),
        .angular_mode10_ref3(angular_mode10_ref3),
        .angular_mode10_ref4(angular_mode10_ref4),
        .angular_mode10_ref5(angular_mode10_ref5),

        .angular_mode11_ref1(angular_mode11_ref1),
        .angular_mode11_ref2(angular_mode11_ref2),
        .angular_mode11_ref3(angular_mode11_ref3),
        .angular_mode11_ref4(angular_mode11_ref4),
        .angular_mode11_ref5(angular_mode11_ref5),

        .angular_mode12_ref1(angular_mode12_ref1),
        .angular_mode12_ref2(angular_mode12_ref2),
```

```
        .angular_mode12_ref3(angular_mode12_ref3),
        .angular_mode12_ref4(angular_mode12_ref4),
        .angular_mode12_ref5(angular_mode12_ref5),

        .angular_mode13_ref1(angular_mode13_ref1),
        .angular_mode13_ref2(angular_mode13_ref2),
        .angular_mode13_ref3(angular_mode13_ref3),
        .angular_mode13_ref4(angular_mode13_ref4),
        .angular_mode13_ref5(angular_mode13_ref5),

        .angular_mode14_ref1(angular_mode14_ref1),
        .angular_mode14_ref2(angular_mode14_ref2),
        .angular_mode14_ref3(angular_mode14_ref3),
        .angular_mode14_ref4(angular_mode14_ref4),
        .angular_mode14_ref5(angular_mode14_ref5),

        .angular_mode15_ref1(angular_mode15_ref1),
        .angular_mode15_ref2(angular_mode15_ref2),
        .angular_mode15_ref3(angular_mode15_ref3),
        .angular_mode15_ref4(angular_mode15_ref4),
        .angular_mode15_ref5(angular_mode15_ref5),

        .angular_mode16_ref1(angular_mode16_ref1),
        .angular_mode16_ref2(angular_mode16_ref2),
        .angular_mode16_ref3(angular_mode16_ref3),
        .angular_mode16_ref4(angular_mode16_ref4),
        .angular_mode16_ref5(angular_mode16_ref5),

        .angular_mode17_ref1(angular_mode17_ref1),
        .angular_mode17_ref2(angular_mode17_ref2),
        .angular_mode17_ref3(angular_mode17_ref3),
        .angular_mode17_ref4(angular_mode17_ref4),
        .angular_mode17_ref5(angular_mode17_ref5),

        .angular_mode18_ref1(angular_mode18_ref1),
        .angular_mode18_ref2(angular_mode18_ref2),
        .angular_mode18_ref3(angular_mode18_ref3),
        .angular_mode18_ref4(angular_mode18_ref4),
        .angular_mode18_ref5(angular_mode18_ref5),

        .angular_mode19_ref1(angular_mode19_ref1),
        .angular_mode19_ref2(angular_mode19_ref2),
        .angular_mode19_ref3(angular_mode19_ref3),
        .angular_mode19_ref4(angular_mode19_ref4),
        .angular_mode19_ref5(angular_mode19_ref5),

        .angular_mode20_ref1(angular_mode20_ref1),
        .angular_mode20_ref2(angular_mode20_ref2),
        .angular_mode20_ref3(angular_mode20_ref3),
        .angular_mode20_ref4(angular_mode20_ref4),
        .angular_mode20_ref5(angular_mode20_ref5),

        .angular_mode21_ref1(angular_mode21_ref1),
        .angular_mode21_ref2(angular_mode21_ref2),
        .angular_mode21_ref3(angular_mode21_ref3),
        .angular_mode21_ref4(angular_mode21_ref4),
        .angular_mode21_ref5(angular_mode21_ref5),
```

```
        .angular_mode22_ref1(angular_mode22_ref1),
        .angular_mode22_ref2(angular_mode22_ref2),
        .angular_mode22_ref3(angular_mode22_ref3),
        .angular_mode22_ref4(angular_mode22_ref4),
        .angular_mode22_ref5(angular_mode22_ref5),

        .angular_mode23_ref1(angular_mode23_ref1),
        .angular_mode23_ref2(angular_mode23_ref2),
        .angular_mode23_ref3(angular_mode23_ref3),
        .angular_mode23_ref4(angular_mode23_ref4),
        .angular_mode23_ref5(angular_mode23_ref5),

        .angular_mode24_ref1(angular_mode24_ref1),
        .angular_mode24_ref2(angular_mode24_ref2),
        .angular_mode24_ref3(angular_mode24_ref3),
        .angular_mode24_ref4(angular_mode24_ref4),
        .angular_mode24_ref5(angular_mode24_ref5),

        .angular_mode25_ref1(angular_mode25_ref1),
        .angular_mode25_ref2(angular_mode25_ref2),
        .angular_mode25_ref3(angular_mode25_ref3),
        .angular_mode25_ref4(angular_mode25_ref4),
        .angular_mode25_ref5(angular_mode25_ref5),

        .angular_mode26_ref0(angular_mode26_ref0),
        .angular_mode26_ref1(angular_mode26_ref1),
        .angular_mode26_ref2(angular_mode26_ref2),
        .angular_mode26_ref3(angular_mode26_ref3),
        .angular_mode26_ref4(angular_mode26_ref4),
        .angular_mode26_ref5(angular_mode26_ref5),

        .angular_mode27_ref1(angular_mode27_ref1),
        .angular_mode27_ref2(angular_mode27_ref2),
        .angular_mode27_ref3(angular_mode27_ref3),
        .angular_mode27_ref4(angular_mode27_ref4),
        .angular_mode27_ref5(angular_mode27_ref5),

        .angular_mode28_ref1(angular_mode28_ref1),
        .angular_mode28_ref2(angular_mode28_ref2),
        .angular_mode28_ref3(angular_mode28_ref3),
        .angular_mode28_ref4(angular_mode28_ref4),
        .angular_mode28_ref5(angular_mode28_ref5),

        .angular_mode29_ref1(angular_mode29_ref1),
        .angular_mode29_ref2(angular_mode29_ref2),
        .angular_mode29_ref3(angular_mode29_ref3),
        .angular_mode29_ref4(angular_mode29_ref4),
        .angular_mode29_ref5(angular_mode29_ref5),

        .angular_mode30_ref1(angular_mode30_ref1),
        .angular_mode30_ref2(angular_mode30_ref2),
        .angular_mode30_ref3(angular_mode30_ref3),
        .angular_mode30_ref4(angular_mode30_ref4),
        .angular_mode30_ref5(angular_mode30_ref5),

        .angular_mode31_ref1(angular_mode31_ref1),
```

```
1279        .angular_mode31_ref2(angular_mode31_ref2),
1280        .angular_mode31_ref3(angular_mode31_ref3),
1281        .angular_mode31_ref4(angular_mode31_ref4),
1282        .angular_mode31_ref5(angular_mode31_ref5),
1283
1284        .angular_mode32_ref1(angular_mode32_ref1),
1285        .angular_mode32_ref2(angular_mode32_ref2),
1286        .angular_mode32_ref3(angular_mode32_ref3),
1287        .angular_mode32_ref4(angular_mode32_ref4),
1288        .angular_mode32_ref5(angular_mode32_ref5),
1289
1290        .angular_mode33_ref1(angular_mode33_ref1),
1291        .angular_mode33_ref2(angular_mode33_ref2),
1292        .angular_mode33_ref3(angular_mode33_ref3),
1293        .angular_mode33_ref4(angular_mode33_ref4),
1294        .angular_mode33_ref5(angular_mode33_ref5),
1295
1296        .angular_mode34_ref1(angular_mode34_ref1),
1297        .angular_mode34_ref2(angular_mode34_ref2),
1298        .angular_mode34_ref3(angular_mode34_ref3),
1299        .angular_mode34_ref4(angular_mode34_ref4),
1300        .angular_mode34_ref5(angular_mode34_ref5)
1301    );
1302
1303    planar_top_level #(WIDTH) planar_block(
1304        .clock(clock),
1305        .reset(reset_registers),
1306        .start(start_predictors),
1307        .sample_0n(planar_sample_0n),
1308        .sample_1n(planar_sample_1n),
1309        .sample_2n(planar_sample_2n),
1310        .sample_3n(planar_sample_3n),
1311        .sample_Nn(planar_sample_Nn),
1312        .sample_n0(planar_sample_n0),
1313        .sample_n1(planar_sample_n1),
1314        .sample_n2(planar_sample_n2),
1315        .sample_n3(planar_sample_n3),
1316        .sample_nN(planar_sample_nN),
1317        .N(N),
1318
1319        .samples_x(planar_samples_x),
1320        .samples_y(planar_samples_y),
1321        .part_done(check_planar_step),
1322        .done(planar_done),
1323        .p00(planar_p00),
1324        .p01(planar_p01),
1325        .p02(planar_p02),
1326        .p03(planar_p03),
1327        .p10(planar_p10),
1328        .p11(planar_p11),
1329        .p12(planar_p12),
1330        .p13(planar_p13),
1331        .p20(planar_p20),
1332        .p21(planar_p21),
1333        .p22(planar_p22),
1334        .p23(planar_p23),
1335        .p30(planar_p30),
```

```
        .p31(planar_p31),
        .p32(planar_p32),
        .p33(planar_p33)
    );

    dc_top_level #(WIDTH) dc_block(
        .clock(clock),
        .reset(reset_registers),
        .start(start_predictors),
        .sample_0n(dc_sample_0n),
        .sample_1n(dc_sample_1n),
        .sample_2n(dc_sample_2n),
        .sample_3n(dc_sample_3n),
        .sample_n0(dc_sample_n0),
        .sample_n1(dc_sample_n1),
        .sample_n2(dc_sample_n2),
        .sample_n3(dc_sample_n3),
        .N(N),

        .samples_x(dc_samples_x),
        .samples_y(dc_samples_y),
        .part_done(check_dc_step),
        .done(dc_done),
        .p00(dc_p00),
        .p01(dc_p01),
        .p02(dc_p02),
        .p03(dc_p03),
        .p10(dc_p10),
        .p11(dc_p11),
        .p12(dc_p12),
        .p13(dc_p13),
        .p20(dc_p20),
        .p21(dc_p21),
        .p22(dc_p22),
        .p23(dc_p23),
        .p30(dc_p30),
        .p31(dc_p31),
        .p32(dc_p32),
        .p33(dc_p33)
    );

    angular_top_level #(WIDTH, 2) angular_mode2_block(
        .clock(clock),
        .reset(reset_registers),
        .start(start_predictors),
        .ref1(angular_mode2_ref1),
        .ref2(angular_mode2_ref2),
        .ref3(angular_mode2_ref3),
        .ref4(angular_mode2_ref4),
        .ref5(angular_mode2_ref5),
        .N(N),

        .samples(angular_mode2_samples),
        .part_done(check_angular_mode2_step),
        .done(angular_mode2_done),
        .p00(angular_mode2_p00),
        .p01(angular_mode2_p01),
```

```
        .p02(angular_mode2_p02),
        .p03(angular_mode2_p03),
        .p10(angular_mode2_p10),
        .p11(angular_mode2_p11),
        .p12(angular_mode2_p12),
        .p13(angular_mode2_p13),
        .p20(angular_mode2_p20),
        .p21(angular_mode2_p21),
        .p22(angular_mode2_p22),
        .p23(angular_mode2_p23),
        .p30(angular_mode2_p30),
        .p31(angular_mode2_p31),
        .p32(angular_mode2_p32),
        .p33(angular_mode2_p33)
    );

    angular_top_level #(WIDTH, 3) angular_mode3_block(
        .clock(clock),
        .reset(reset_registers),
        .start(start_predictors),
        .ref1(angular_mode3_ref1),
        .ref2(angular_mode3_ref2),
        .ref3(angular_mode3_ref3),
        .ref4(angular_mode3_ref4),
        .ref5(angular_mode3_ref5),
        .N(N),

        .samples(angular_mode3_samples),
        .part_done(check_angular_mode3_step),
        .done(angular_mode3_done),
        .p00(angular_mode3_p00),
        .p01(angular_mode3_p01),
        .p02(angular_mode3_p02),
        .p03(angular_mode3_p03),
        .p10(angular_mode3_p10),
        .p11(angular_mode3_p11),
        .p12(angular_mode3_p12),
        .p13(angular_mode3_p13),
        .p20(angular_mode3_p20),
        .p21(angular_mode3_p21),
        .p22(angular_mode3_p22),
        .p23(angular_mode3_p23),
        .p30(angular_mode3_p30),
        .p31(angular_mode3_p31),
        .p32(angular_mode3_p32),
        .p33(angular_mode3_p33)
    );

    angular_top_level #(WIDTH, 4) angular_mode4_block(
        .clock(clock),
        .reset(reset_registers),
        .start(start_predictors),
        .ref1(angular_mode4_ref1),
        .ref2(angular_mode4_ref2),
        .ref3(angular_mode4_ref3),
        .ref4(angular_mode4_ref4),
        .ref5(angular_mode4_ref5),
```

```
        .N(N),

        .samples(angular_mode4_samples),
        .part_done(check_angular_mode4_step),
        .done(angular_mode4_done),
        .p00(angular_mode4_p00),
        .p01(angular_mode4_p01),
        .p02(angular_mode4_p02),
        .p03(angular_mode4_p03),
        .p10(angular_mode4_p10),
        .p11(angular_mode4_p11),
        .p12(angular_mode4_p12),
        .p13(angular_mode4_p13),
        .p20(angular_mode4_p20),
        .p21(angular_mode4_p21),
        .p22(angular_mode4_p22),
        .p23(angular_mode4_p23),
        .p30(angular_mode4_p30),
        .p31(angular_mode4_p31),
        .p32(angular_mode4_p32),
        .p33(angular_mode4_p33)
    );

    angular_top_level #(WIDTH, 5) angular_mode5_block(
        .clock(clock),
        .reset(reset_registers),
        .start(start_predictors),
        .ref1(angular_mode5_ref1),
        .ref2(angular_mode5_ref2),
        .ref3(angular_mode5_ref3),
        .ref4(angular_mode5_ref4),
        .ref5(angular_mode5_ref5),
        .N(N),

        .samples(angular_mode5_samples),
        .part_done(check_angular_mode5_step),
        .done(angular_mode5_done),
        .p00(angular_mode5_p00),
        .p01(angular_mode5_p01),
        .p02(angular_mode5_p02),
        .p03(angular_mode5_p03),
        .p10(angular_mode5_p10),
        .p11(angular_mode5_p11),
        .p12(angular_mode5_p12),
        .p13(angular_mode5_p13),
        .p20(angular_mode5_p20),
        .p21(angular_mode5_p21),
        .p22(angular_mode5_p22),
        .p23(angular_mode5_p23),
        .p30(angular_mode5_p30),
        .p31(angular_mode5_p31),
        .p32(angular_mode5_p32),
        .p33(angular_mode5_p33)
    );

    angular_top_level #(WIDTH, 6) angular_mode6_block(
        .clock(clock),
```

```
        .reset(reset_registers),
        .start(start_predictors),
        .ref1(angular_mode6_ref1),
        .ref2(angular_mode6_ref2),
        .ref3(angular_mode6_ref3),
        .ref4(angular_mode6_ref4),
        .ref5(angular_mode6_ref5),
        .N(N),

        .samples(angular_mode6_samples),
        .part_done(check_angular_mode6_step),
        .done(angular_mode6_done),
        .p00(angular_mode6_p00),
        .p01(angular_mode6_p01),
        .p02(angular_mode6_p02),
        .p03(angular_mode6_p03),
        .p10(angular_mode6_p10),
        .p11(angular_mode6_p11),
        .p12(angular_mode6_p12),
        .p13(angular_mode6_p13),
        .p20(angular_mode6_p20),
        .p21(angular_mode6_p21),
        .p22(angular_mode6_p22),
        .p23(angular_mode6_p23),
        .p30(angular_mode6_p30),
        .p31(angular_mode6_p31),
        .p32(angular_mode6_p32),
        .p33(angular_mode6_p33)
    );

    angular_top_level #(WIDTH, 7) angular_mode7_block(
        .clock(clock),
        .reset(reset_registers),
        .start(start_predictors),
        .ref1(angular_mode7_ref1),
        .ref2(angular_mode7_ref2),
        .ref3(angular_mode7_ref3),
        .ref4(angular_mode7_ref4),
        .ref5(angular_mode7_ref5),
        .N(N),

        .samples(angular_mode7_samples),
        .part_done(check_angular_mode7_step),
        .done(angular_mode7_done),
        .p00(angular_mode7_p00),
        .p01(angular_mode7_p01),
        .p02(angular_mode7_p02),
        .p03(angular_mode7_p03),
        .p10(angular_mode7_p10),
        .p11(angular_mode7_p11),
        .p12(angular_mode7_p12),
        .p13(angular_mode7_p13),
        .p20(angular_mode7_p20),
        .p21(angular_mode7_p21),
        .p22(angular_mode7_p22),
        .p23(angular_mode7_p23),
        .p30(angular_mode7_p30),
```

```
1564        .p31(angular_mode7_p31),
1565        .p32(angular_mode7_p32),
1566        .p33(angular_mode7_p33)
1567    );
1568
1569    angular_top_level #(WIDTH, 8) angular_mode8_block(
1570        .clock(clock),
1571        .reset(reset_registers),
1572        .start(start_predictors),
1573        .ref1(angular_mode8_ref1),
1574        .ref2(angular_mode8_ref2),
1575        .ref3(angular_mode8_ref3),
1576        .ref4(angular_mode8_ref4),
1577        .ref5(angular_mode8_ref5),
1578        .N(N),
1579
1580        .samples(angular_mode8_samples),
1581        .part_done(check_angular_mode8_step),
1582        .done(angular_mode8_done),
1583        .p00(angular_mode8_p00),
1584        .p01(angular_mode8_p01),
1585        .p02(angular_mode8_p02),
1586        .p03(angular_mode8_p03),
1587        .p10(angular_mode8_p10),
1588        .p11(angular_mode8_p11),
1589        .p12(angular_mode8_p12),
1590        .p13(angular_mode8_p13),
1591        .p20(angular_mode8_p20),
1592        .p21(angular_mode8_p21),
1593        .p22(angular_mode8_p22),
1594        .p23(angular_mode8_p23),
1595        .p30(angular_mode8_p30),
1596        .p31(angular_mode8_p31),
1597        .p32(angular_mode8_p32),
1598        .p33(angular_mode8_p33)
1599    );
1600
1601    angular_top_level #(WIDTH, 9) angular_mode9_block(
1602        .clock(clock),
1603        .reset(reset_registers),
1604        .start(start_predictors),
1605        .ref1(angular_mode9_ref1),
1606        .ref2(angular_mode9_ref2),
1607        .ref3(angular_mode9_ref3),
1608        .ref4(angular_mode9_ref4),
1609        .ref5(angular_mode9_ref5),
1610        .N(N),
1611
1612        .samples(angular_mode9_samples),
1613        .part_done(check_angular_mode9_step),
1614        .done(angular_mode9_done),
1615        .p00(angular_mode9_p00),
1616        .p01(angular_mode9_p01),
1617        .p02(angular_mode9_p02),
1618        .p03(angular_mode9_p03),
1619        .p10(angular_mode9_p10),
1620        .p11(angular_mode9_p11),
```

```
1621        .p12(angular_mode9_p12),
1622        .p13(angular_mode9_p13),
1623        .p20(angular_mode9_p20),
1624        .p21(angular_mode9_p21),
1625        .p22(angular_mode9_p22),
1626        .p23(angular_mode9_p23),
1627        .p30(angular_mode9_p30),
1628        .p31(angular_mode9_p31),
1629        .p32(angular_mode9_p32),
1630        .p33(angular_mode9_p33)
1631    );
1632
1633    angular_directly_top_level #(WIDTH, 10) angular_mode10_block(
1634        .clock(clock),
1635        .reset(reset_registers),
1636        .start(start_predictors),
1637        .ref0(angular_mode10_ref0),
1638        .ref1(angular_mode10_ref1),
1639        .ref2(angular_mode10_ref2),
1640        .ref3(angular_mode10_ref3),
1641        .ref4(angular_mode10_ref4),
1642        .ref5(angular_mode10_ref5),
1643        .N(N),
1644
1645        .filter_samples(filter_samples_mode10),
1646        .samples(angular_mode10_samples),
1647        .part_done(check_angular_mode10_step),
1648        .done(angular_mode10_done),
1649        .p00(angular_mode10_p00),
1650        .p01(angular_mode10_p01),
1651        .p02(angular_mode10_p02),
1652        .p03(angular_mode10_p03),
1653        .p10(angular_mode10_p10),
1654        .p11(angular_mode10_p11),
1655        .p12(angular_mode10_p12),
1656        .p13(angular_mode10_p13),
1657        .p20(angular_mode10_p20),
1658        .p21(angular_mode10_p21),
1659        .p22(angular_mode10_p22),
1660        .p23(angular_mode10_p23),
1661        .p30(angular_mode10_p30),
1662        .p31(angular_mode10_p31),
1663        .p32(angular_mode10_p32),
1664        .p33(angular_mode10_p33)
1665    );
1666
1667    angular_top_level #(WIDTH, 11) angular_mode11_block(
1668        .clock(clock),
1669        .reset(reset_registers),
1670        .start(start_predictors),
1671        .ref1(angular_mode11_ref1),
1672        .ref2(angular_mode11_ref2),
1673        .ref3(angular_mode11_ref3),
1674        .ref4(angular_mode11_ref4),
1675        .ref5(angular_mode11_ref5),
1676        .N(N),
1677
```

```
        .samples(angular_mode11_samples),
        .part_done(check_angular_mode11_step),
        .done(angular_mode11_done),
        .p00(angular_mode11_p00),
        .p01(angular_mode11_p01),
        .p02(angular_mode11_p02),
        .p03(angular_mode11_p03),
        .p10(angular_mode11_p10),
        .p11(angular_mode11_p11),
        .p12(angular_mode11_p12),
        .p13(angular_mode11_p13),
        .p20(angular_mode11_p20),
        .p21(angular_mode11_p21),
        .p22(angular_mode11_p22),
        .p23(angular_mode11_p23),
        .p30(angular_mode11_p30),
        .p31(angular_mode11_p31),
        .p32(angular_mode11_p32),
        .p33(angular_mode11_p33)
    );

    angular_top_level #(WIDTH, 12) angular_mode12_block(
        .clock(clock),
        .reset(reset_registers),
        .start(start_predictors),
        .ref1(angular_mode12_ref1),
        .ref2(angular_mode12_ref2),
        .ref3(angular_mode12_ref3),
        .ref4(angular_mode12_ref4),
        .ref5(angular_mode12_ref5),
        .N(N),

        .samples(angular_mode12_samples),
        .part_done(check_angular_mode12_step),
        .done(angular_mode12_done),
        .p00(angular_mode12_p00),
        .p01(angular_mode12_p01),
        .p02(angular_mode12_p02),
        .p03(angular_mode12_p03),
        .p10(angular_mode12_p10),
        .p11(angular_mode12_p11),
        .p12(angular_mode12_p12),
        .p13(angular_mode12_p13),
        .p20(angular_mode12_p20),
        .p21(angular_mode12_p21),
        .p22(angular_mode12_p22),
        .p23(angular_mode12_p23),
        .p30(angular_mode12_p30),
        .p31(angular_mode12_p31),
        .p32(angular_mode12_p32),
        .p33(angular_mode12_p33)
    );

    angular_top_level #(WIDTH, 13) angular_mode13_block(
        .clock(clock),
        .reset(reset_registers),
        .start(start_predictors),
```

```
        .ref1(angular_mode13_ref1),
        .ref2(angular_mode13_ref2),
        .ref3(angular_mode13_ref3),
        .ref4(angular_mode13_ref4),
        .ref5(angular_mode13_ref5),
        .N(N),

        .samples(angular_mode13_samples),
        .part_done(check_angular_mode13_step),
        .done(angular_mode13_done),
        .p00(angular_mode13_p00),
        .p01(angular_mode13_p01),
        .p02(angular_mode13_p02),
        .p03(angular_mode13_p03),
        .p10(angular_mode13_p10),
        .p11(angular_mode13_p11),
        .p12(angular_mode13_p12),
        .p13(angular_mode13_p13),
        .p20(angular_mode13_p20),
        .p21(angular_mode13_p21),
        .p22(angular_mode13_p22),
        .p23(angular_mode13_p23),
        .p30(angular_mode13_p30),
        .p31(angular_mode13_p31),
        .p32(angular_mode13_p32),
        .p33(angular_mode13_p33)
    );

    angular_top_level #(WIDTH, 14) angular_mode14_block(
        .clock(clock),
        .reset(reset_registers),
        .start(start_predictors),
        .ref1(angular_mode14_ref1),
        .ref2(angular_mode14_ref2),
        .ref3(angular_mode14_ref3),
        .ref4(angular_mode14_ref4),
        .ref5(angular_mode14_ref5),
        .N(N),

        .samples(angular_mode14_samples),
        .part_done(check_angular_mode14_step),
        .done(angular_mode14_done),
        .p00(angular_mode14_p00),
        .p01(angular_mode14_p01),
        .p02(angular_mode14_p02),
        .p03(angular_mode14_p03),
        .p10(angular_mode14_p10),
        .p11(angular_mode14_p11),
        .p12(angular_mode14_p12),
        .p13(angular_mode14_p13),
        .p20(angular_mode14_p20),
        .p21(angular_mode14_p21),
        .p22(angular_mode14_p22),
        .p23(angular_mode14_p23),
        .p30(angular_mode14_p30),
        .p31(angular_mode14_p31),
        .p32(angular_mode14_p32),
```

```
        .p33(angular_mode14_p33)
    );

    angular_top_level #(WIDTH, 15) angular_mode15_block(
        .clock(clock),
        .reset(reset_registers),
        .start(start_predictors),
        .ref1(angular_mode15_ref1),
        .ref2(angular_mode15_ref2),
        .ref3(angular_mode15_ref3),
        .ref4(angular_mode15_ref4),
        .ref5(angular_mode15_ref5),
        .N(N),

        .samples(angular_mode15_samples),
        .part_done(check_angular_mode15_step),
        .done(angular_mode15_done),
        .p00(angular_mode15_p00),
        .p01(angular_mode15_p01),
        .p02(angular_mode15_p02),
        .p03(angular_mode15_p03),
        .p10(angular_mode15_p10),
        .p11(angular_mode15_p11),
        .p12(angular_mode15_p12),
        .p13(angular_mode15_p13),
        .p20(angular_mode15_p20),
        .p21(angular_mode15_p21),
        .p22(angular_mode15_p22),
        .p23(angular_mode15_p23),
        .p30(angular_mode15_p30),
        .p31(angular_mode15_p31),
        .p32(angular_mode15_p32),
        .p33(angular_mode15_p33)
    );

    angular_top_level #(WIDTH, 16) angular_mode16_block(
        .clock(clock),
        .reset(reset_registers),
        .start(start_predictors),
        .ref1(angular_mode16_ref1),
        .ref2(angular_mode16_ref2),
        .ref3(angular_mode16_ref3),
        .ref4(angular_mode16_ref4),
        .ref5(angular_mode16_ref5),
        .N(N),

        .samples(angular_mode16_samples),
        .part_done(check_angular_mode16_step),
        .done(angular_mode16_done),
        .p00(angular_mode16_p00),
        .p01(angular_mode16_p01),
        .p02(angular_mode16_p02),
        .p03(angular_mode16_p03),
        .p10(angular_mode16_p10),
        .p11(angular_mode16_p11),
        .p12(angular_mode16_p12),
        .p13(angular_mode16_p13),
```

```
        .p20(angular_mode16_p20),
        .p21(angular_mode16_p21),
        .p22(angular_mode16_p22),
        .p23(angular_mode16_p23),
        .p30(angular_mode16_p30),
        .p31(angular_mode16_p31),
        .p32(angular_mode16_p32),
        .p33(angular_mode16_p33)
    );

    angular_top_level #(WIDTH, 17) angular_mode17_block(
        .clock(clock),
        .reset(reset_registers),
        .start(start_predictors),
        .ref1(angular_mode17_ref1),
        .ref2(angular_mode17_ref2),
        .ref3(angular_mode17_ref3),
        .ref4(angular_mode17_ref4),
        .ref5(angular_mode17_ref5),
        .N(N),

        .samples(angular_mode17_samples),
        .part_done(check_angular_mode17_step),
        .done(angular_mode17_done),
        .p00(angular_mode17_p00),
        .p01(angular_mode17_p01),
        .p02(angular_mode17_p02),
        .p03(angular_mode17_p03),
        .p10(angular_mode17_p10),
        .p11(angular_mode17_p11),
        .p12(angular_mode17_p12),
        .p13(angular_mode17_p13),
        .p20(angular_mode17_p20),
        .p21(angular_mode17_p21),
        .p22(angular_mode17_p22),
        .p23(angular_mode17_p23),
        .p30(angular_mode17_p30),
        .p31(angular_mode17_p31),
        .p32(angular_mode17_p32),
        .p33(angular_mode17_p33)
    );

    angular_top_level #(WIDTH, 18) angular_mode18_block(
        .clock(clock),
        .reset(reset_registers),
        .start(start_predictors),
        .ref1(angular_mode18_ref1),
        .ref2(angular_mode18_ref2),
        .ref3(angular_mode18_ref3),
        .ref4(angular_mode18_ref4),
        .ref5(angular_mode18_ref5),
        .N(N),

        .samples(angular_mode18_samples),
        .part_done(check_angular_mode18_step),
        .done(angular_mode18_done),
        .p00(angular_mode18_p00),
```

```
        .p01(angular_mode18_p01),
        .p02(angular_mode18_p02),
        .p03(angular_mode18_p03),
        .p10(angular_mode18_p10),
        .p11(angular_mode18_p11),
        .p12(angular_mode18_p12),
        .p13(angular_mode18_p13),
        .p20(angular_mode18_p20),
        .p21(angular_mode18_p21),
        .p22(angular_mode18_p22),
        .p23(angular_mode18_p23),
        .p30(angular_mode18_p30),
        .p31(angular_mode18_p31),
        .p32(angular_mode18_p32),
        .p33(angular_mode18_p33)
    );

    angular_top_level #(WIDTH, 19) angular_mode19_block(
        .clock(clock),
        .reset(reset_registers),
        .start(start_predictors),
        .ref1(angular_mode19_ref1),
        .ref2(angular_mode19_ref2),
        .ref3(angular_mode19_ref3),
        .ref4(angular_mode19_ref4),
        .ref5(angular_mode19_ref5),
        .N(N),

        .samples(angular_mode19_samples),
        .part_done(check_angular_mode19_step),
        .done(angular_mode19_done),
        .p00(angular_mode19_p00),
        .p01(angular_mode19_p01),
        .p02(angular_mode19_p02),
        .p03(angular_mode19_p03),
        .p10(angular_mode19_p10),
        .p11(angular_mode19_p11),
        .p12(angular_mode19_p12),
        .p13(angular_mode19_p13),
        .p20(angular_mode19_p20),
        .p21(angular_mode19_p21),
        .p22(angular_mode19_p22),
        .p23(angular_mode19_p23),
        .p30(angular_mode19_p30),
        .p31(angular_mode19_p31),
        .p32(angular_mode19_p32),
        .p33(angular_mode19_p33)
    );

    angular_top_level #(WIDTH, 20) angular_mode20_block(
        .clock(clock),
        .reset(reset_registers),
        .start(start_predictors),
        .ref1(angular_mode20_ref1),
        .ref2(angular_mode20_ref2),
        .ref3(angular_mode20_ref3),
        .ref4(angular_mode20_ref4),
```

```
        .ref5(angular_mode20_ref5),
        .N(N),

        .samples(angular_mode20_samples),
        .part_done(check_angular_mode20_step),
        .done(angular_mode20_done),
        .p00(angular_mode20_p00),
        .p01(angular_mode20_p01),
        .p02(angular_mode20_p02),
        .p03(angular_mode20_p03),
        .p10(angular_mode20_p10),
        .p11(angular_mode20_p11),
        .p12(angular_mode20_p12),
        .p13(angular_mode20_p13),
        .p20(angular_mode20_p20),
        .p21(angular_mode20_p21),
        .p22(angular_mode20_p22),
        .p23(angular_mode20_p23),
        .p30(angular_mode20_p30),
        .p31(angular_mode20_p31),
        .p32(angular_mode20_p32),
        .p33(angular_mode20_p33)
    );

    angular_top_level #(WIDTH, 21) angular_mode21_block(
        .clock(clock),
        .reset(reset_registers),
        .start(start_predictors),
        .ref1(angular_mode21_ref1),
        .ref2(angular_mode21_ref2),
        .ref3(angular_mode21_ref3),
        .ref4(angular_mode21_ref4),
        .ref5(angular_mode21_ref5),
        .N(N),

        .samples(angular_mode21_samples),
        .part_done(check_angular_mode21_step),
        .done(angular_mode21_done),
        .p00(angular_mode21_p00),
        .p01(angular_mode21_p01),
        .p02(angular_mode21_p02),
        .p03(angular_mode21_p03),
        .p10(angular_mode21_p10),
        .p11(angular_mode21_p11),
        .p12(angular_mode21_p12),
        .p13(angular_mode21_p13),
        .p20(angular_mode21_p20),
        .p21(angular_mode21_p21),
        .p22(angular_mode21_p22),
        .p23(angular_mode21_p23),
        .p30(angular_mode21_p30),
        .p31(angular_mode21_p31),
        .p32(angular_mode21_p32),
        .p33(angular_mode21_p33)
    );

    angular_top_level #(WIDTH, 22) angular_mode22_block(
```

```
        .clock(clock),
        .reset(reset_registers),
        .start(start_predictors),
        .ref1(angular_mode22_ref1),
        .ref2(angular_mode22_ref2),
        .ref3(angular_mode22_ref3),
        .ref4(angular_mode22_ref4),
        .ref5(angular_mode22_ref5),
        .N(N),

        .samples(angular_mode22_samples),
        .part_done(check_angular_mode22_step),
        .done(angular_mode22_done),
        .p00(angular_mode22_p00),
        .p01(angular_mode22_p01),
        .p02(angular_mode22_p02),
        .p03(angular_mode22_p03),
        .p10(angular_mode22_p10),
        .p11(angular_mode22_p11),
        .p12(angular_mode22_p12),
        .p13(angular_mode22_p13),
        .p20(angular_mode22_p20),
        .p21(angular_mode22_p21),
        .p22(angular_mode22_p22),
        .p23(angular_mode22_p23),
        .p30(angular_mode22_p30),
        .p31(angular_mode22_p31),
        .p32(angular_mode22_p32),
        .p33(angular_mode22_p33)
    );

    angular_top_level #(WIDTH, 23) angular_mode23_block(
        .clock(clock),
        .reset(reset_registers),
        .start(start_predictors),
        .ref1(angular_mode23_ref1),
        .ref2(angular_mode23_ref2),
        .ref3(angular_mode23_ref3),
        .ref4(angular_mode23_ref4),
        .ref5(angular_mode23_ref5),
        .N(N),

        .samples(angular_mode23_samples),
        .part_done(check_angular_mode23_step),
        .done(angular_mode23_done),
        .p00(angular_mode23_p00),
        .p01(angular_mode23_p01),
        .p02(angular_mode23_p02),
        .p03(angular_mode23_p03),
        .p10(angular_mode23_p10),
        .p11(angular_mode23_p11),
        .p12(angular_mode23_p12),
        .p13(angular_mode23_p13),
        .p20(angular_mode23_p20),
        .p21(angular_mode23_p21),
        .p22(angular_mode23_p22),
        .p23(angular_mode23_p23),
```

```
2077        .p30(angular_mode23_p30),
2078        .p31(angular_mode23_p31),
2079        .p32(angular_mode23_p32),
2080        .p33(angular_mode23_p33)
2081    );
2082
2083    angular_top_level #(WIDTH, 24) angular_mode24_block(
2084        .clock(clock),
2085        .reset(reset_registers),
2086        .start(start_predictors),
2087        .ref1(angular_mode24_ref1),
2088        .ref2(angular_mode24_ref2),
2089        .ref3(angular_mode24_ref3),
2090        .ref4(angular_mode24_ref4),
2091        .ref5(angular_mode24_ref5),
2092        .N(N),
2093
2094        .samples(angular_mode24_samples),
2095        .part_done(check_angular_mode24_step),
2096        .done(angular_mode24_done),
2097        .p00(angular_mode24_p00),
2098        .p01(angular_mode24_p01),
2099        .p02(angular_mode24_p02),
2100        .p03(angular_mode24_p03),
2101        .p10(angular_mode24_p10),
2102        .p11(angular_mode24_p11),
2103        .p12(angular_mode24_p12),
2104        .p13(angular_mode24_p13),
2105        .p20(angular_mode24_p20),
2106        .p21(angular_mode24_p21),
2107        .p22(angular_mode24_p22),
2108        .p23(angular_mode24_p23),
2109        .p30(angular_mode24_p30),
2110        .p31(angular_mode24_p31),
2111        .p32(angular_mode24_p32),
2112        .p33(angular_mode24_p33)
2113    );
2114
2115    angular_top_level #(WIDTH, 25) angular_mode25_block(
2116        .clock(clock),
2117        .reset(reset_registers),
2118        .start(start_predictors),
2119        .ref1(angular_mode25_ref1),
2120        .ref2(angular_mode25_ref2),
2121        .ref3(angular_mode25_ref3),
2122        .ref4(angular_mode25_ref4),
2123        .ref5(angular_mode25_ref5),
2124        .N(N),
2125
2126        .samples(angular_mode25_samples),
2127        .part_done(check_angular_mode25_step),
2128        .done(angular_mode25_done),
2129        .p00(angular_mode25_p00),
2130        .p01(angular_mode25_p01),
2131        .p02(angular_mode25_p02),
2132        .p03(angular_mode25_p03),
2133        .p10(angular_mode25_p10),
```

```
        .p11(angular_mode25_p11),
        .p12(angular_mode25_p12),
        .p13(angular_mode25_p13),
        .p20(angular_mode25_p20),
        .p21(angular_mode25_p21),
        .p22(angular_mode25_p22),
        .p23(angular_mode25_p23),
        .p30(angular_mode25_p30),
        .p31(angular_mode25_p31),
        .p32(angular_mode25_p32),
        .p33(angular_mode25_p33)
    );

    angular_directly_top_level #(WIDTH, 26) angular_mode26_block(
        .clock(clock),
        .reset(reset_registers),
        .start(start_predictors),
        .ref0(angular_mode26_ref0),
        .ref1(angular_mode26_ref1),
        .ref2(angular_mode26_ref2),
        .ref3(angular_mode26_ref3),
        .ref4(angular_mode26_ref4),
        .ref5(angular_mode26_ref5),
        .N(N),

        .filter_samples(filter_samples_mode26),
        .samples(angular_mode26_samples),
        .part_done(check_angular_mode26_step),
        .done(angular_mode26_done),
        .p00(angular_mode26_p00),
        .p01(angular_mode26_p01),
        .p02(angular_mode26_p02),
        .p03(angular_mode26_p03),
        .p10(angular_mode26_p10),
        .p11(angular_mode26_p11),
        .p12(angular_mode26_p12),
        .p13(angular_mode26_p13),
        .p20(angular_mode26_p20),
        .p21(angular_mode26_p21),
        .p22(angular_mode26_p22),
        .p23(angular_mode26_p23),
        .p30(angular_mode26_p30),
        .p31(angular_mode26_p31),
        .p32(angular_mode26_p32),
        .p33(angular_mode26_p33)
    );

    angular_top_level #(WIDTH, 27) angular_mode27_block(
        .clock(clock),
        .reset(reset_registers),
        .start(start_predictors),
        .ref1(angular_mode27_ref1),
        .ref2(angular_mode27_ref2),
        .ref3(angular_mode27_ref3),
        .ref4(angular_mode27_ref4),
        .ref5(angular_mode27_ref5),
        .N(N),
```

```
2191
2192        .samples(angular_mode27_samples),
2193        .part_done(check_angular_mode27_step),
2194        .done(angular_mode27_done),
2195        .p00(angular_mode27_p00),
2196        .p01(angular_mode27_p01),
2197        .p02(angular_mode27_p02),
2198        .p03(angular_mode27_p03),
2199        .p10(angular_mode27_p10),
2200        .p11(angular_mode27_p11),
2201        .p12(angular_mode27_p12),
2202        .p13(angular_mode27_p13),
2203        .p20(angular_mode27_p20),
2204        .p21(angular_mode27_p21),
2205        .p22(angular_mode27_p22),
2206        .p23(angular_mode27_p23),
2207        .p30(angular_mode27_p30),
2208        .p31(angular_mode27_p31),
2209        .p32(angular_mode27_p32),
2210        .p33(angular_mode27_p33)
2211    );
2212
2213    angular_top_level #(WIDTH, 28) angular_mode28_block(
2214        .clock(clock),
2215        .reset(reset_registers),
2216        .start(start_predictors),
2217        .ref1(angular_mode28_ref1),
2218        .ref2(angular_mode28_ref2),
2219        .ref3(angular_mode28_ref3),
2220        .ref4(angular_mode28_ref4),
2221        .ref5(angular_mode28_ref5),
2222        .N(N),
2223
2224        .samples(angular_mode28_samples),
2225        .part_done(check_angular_mode28_step),
2226        .done(angular_mode28_done),
2227        .p00(angular_mode28_p00),
2228        .p01(angular_mode28_p01),
2229        .p02(angular_mode28_p02),
2230        .p03(angular_mode28_p03),
2231        .p10(angular_mode28_p10),
2232        .p11(angular_mode28_p11),
2233        .p12(angular_mode28_p12),
2234        .p13(angular_mode28_p13),
2235        .p20(angular_mode28_p20),
2236        .p21(angular_mode28_p21),
2237        .p22(angular_mode28_p22),
2238        .p23(angular_mode28_p23),
2239        .p30(angular_mode28_p30),
2240        .p31(angular_mode28_p31),
2241        .p32(angular_mode28_p32),
2242        .p33(angular_mode28_p33)
2243    );
2244
2245    angular_top_level #(WIDTH, 29) angular_mode29_block(
2246        .clock(clock),
2247        .reset(reset_registers),
```

```
        .start(start_predictors),
        .ref1(angular_mode29_ref1),
        .ref2(angular_mode29_ref2),
        .ref3(angular_mode29_ref3),
        .ref4(angular_mode29_ref4),
        .ref5(angular_mode29_ref5),
        .N(N),

        .samples(angular_mode29_samples),
        .part_done(check_angular_mode29_step),
        .done(angular_mode29_done),
        .p00(angular_mode29_p00),
        .p01(angular_mode29_p01),
        .p02(angular_mode29_p02),
        .p03(angular_mode29_p03),
        .p10(angular_mode29_p10),
        .p11(angular_mode29_p11),
        .p12(angular_mode29_p12),
        .p13(angular_mode29_p13),
        .p20(angular_mode29_p20),
        .p21(angular_mode29_p21),
        .p22(angular_mode29_p22),
        .p23(angular_mode29_p23),
        .p30(angular_mode29_p30),
        .p31(angular_mode29_p31),
        .p32(angular_mode29_p32),
        .p33(angular_mode29_p33)
    );

    angular_top_level #(WIDTH, 30) angular_mode30_block(
        .clock(clock),
        .reset(reset_registers),
        .start(start_predictors),
        .ref1(angular_mode30_ref1),
        .ref2(angular_mode30_ref2),
        .ref3(angular_mode30_ref3),
        .ref4(angular_mode30_ref4),
        .ref5(angular_mode30_ref5),
        .N(N),

        .samples(angular_mode30_samples),
        .part_done(check_angular_mode30_step),
        .done(angular_mode30_done),
        .p00(angular_mode30_p00),
        .p01(angular_mode30_p01),
        .p02(angular_mode30_p02),
        .p03(angular_mode30_p03),
        .p10(angular_mode30_p10),
        .p11(angular_mode30_p11),
        .p12(angular_mode30_p12),
        .p13(angular_mode30_p13),
        .p20(angular_mode30_p20),
        .p21(angular_mode30_p21),
        .p22(angular_mode30_p22),
        .p23(angular_mode30_p23),
        .p30(angular_mode30_p30),
        .p31(angular_mode30_p31),
```

```
        .p32(angular_mode30_p32),
        .p33(angular_mode30_p33)
    );

    angular_top_level #(WIDTH, 31) angular_mode31_block(
        .clock(clock),
        .reset(reset_registers),
        .start(start_predictors),
        .ref1(angular_mode31_ref1),
        .ref2(angular_mode31_ref2),
        .ref3(angular_mode31_ref3),
        .ref4(angular_mode31_ref4),
        .ref5(angular_mode31_ref5),
        .N(N),

        .samples(angular_mode31_samples),
        .part_done(check_angular_mode31_step),
        .done(angular_mode31_done),
        .p00(angular_mode31_p00),
        .p01(angular_mode31_p01),
        .p02(angular_mode31_p02),
        .p03(angular_mode31_p03),
        .p10(angular_mode31_p10),
        .p11(angular_mode31_p11),
        .p12(angular_mode31_p12),
        .p13(angular_mode31_p13),
        .p20(angular_mode31_p20),
        .p21(angular_mode31_p21),
        .p22(angular_mode31_p22),
        .p23(angular_mode31_p23),
        .p30(angular_mode31_p30),
        .p31(angular_mode31_p31),
        .p32(angular_mode31_p32),
        .p33(angular_mode31_p33)
    );

    angular_top_level #(WIDTH, 32) angular_mode32_block(
        .clock(clock),
        .reset(reset_registers),
        .start(start_predictors),
        .ref1(angular_mode32_ref1),
        .ref2(angular_mode32_ref2),
        .ref3(angular_mode32_ref3),
        .ref4(angular_mode32_ref4),
        .ref5(angular_mode32_ref5),
        .N(N),

        .samples(angular_mode32_samples),
        .part_done(check_angular_mode32_step),
        .done(angular_mode32_done),
        .p00(angular_mode32_p00),
        .p01(angular_mode32_p01),
        .p02(angular_mode32_p02),
        .p03(angular_mode32_p03),
        .p10(angular_mode32_p10),
        .p11(angular_mode32_p11),
        .p12(angular_mode32_p12),
```

```
2362        .p13(angular_mode32_p13),
2363        .p20(angular_mode32_p20),
2364        .p21(angular_mode32_p21),
2365        .p22(angular_mode32_p22),
2366        .p23(angular_mode32_p23),
2367        .p30(angular_mode32_p30),
2368        .p31(angular_mode32_p31),
2369        .p32(angular_mode32_p32),
2370        .p33(angular_mode32_p33)
2371    );
2372
2373    angular_top_level #(WIDTH, 33) angular_mode33_block(
2374        .clock(clock),
2375        .reset(reset_registers),
2376        .start(start_predictors),
2377        .ref1(angular_mode33_ref1),
2378        .ref2(angular_mode33_ref2),
2379        .ref3(angular_mode33_ref3),
2380        .ref4(angular_mode33_ref4),
2381        .ref5(angular_mode33_ref5),
2382        .N(N),
2383
2384        .samples(angular_mode33_samples),
2385        .part_done(check_angular_mode33_step),
2386        .done(angular_mode33_done),
2387        .p00(angular_mode33_p00),
2388        .p01(angular_mode33_p01),
2389        .p02(angular_mode33_p02),
2390        .p03(angular_mode33_p03),
2391        .p10(angular_mode33_p10),
2392        .p11(angular_mode33_p11),
2393        .p12(angular_mode33_p12),
2394        .p13(angular_mode33_p13),
2395        .p20(angular_mode33_p20),
2396        .p21(angular_mode33_p21),
2397        .p22(angular_mode33_p22),
2398        .p23(angular_mode33_p23),
2399        .p30(angular_mode33_p30),
2400        .p31(angular_mode33_p31),
2401        .p32(angular_mode33_p32),
2402        .p33(angular_mode33_p33)
2403    );
2404
2405    angular_top_level #(WIDTH, 34) angular_mode34_block(
2406        .clock(clock),
2407        .reset(reset_registers),
2408        .start(start_predictors),
2409        .ref1(angular_mode34_ref1),
2410        .ref2(angular_mode34_ref2),
2411        .ref3(angular_mode34_ref3),
2412        .ref4(angular_mode34_ref4),
2413        .ref5(angular_mode34_ref5),
2414        .N(N),
2415
2416        .samples(angular_mode34_samples),
2417        .part_done(check_angular_mode34_step),
2418        .done(angular_mode34_done),
```

```
2419        .p00(angular_mode34_p00),
2420        .p01(angular_mode34_p01),
2421        .p02(angular_mode34_p02),
2422        .p03(angular_mode34_p03),
2423        .p10(angular_mode34_p10),
2424        .p11(angular_mode34_p11),
2425        .p12(angular_mode34_p12),
2426        .p13(angular_mode34_p13),
2427        .p20(angular_mode34_p20),
2428        .p21(angular_mode34_p21),
2429        .p22(angular_mode34_p22),
2430        .p23(angular_mode34_p23),
2431        .p30(angular_mode34_p30),
2432        .p31(angular_mode34_p31),
2433        .p32(angular_mode34_p32),
2434        .p33(angular_mode34_p33)
2435    );
2436
2437    assign read_def            = def_enable;
2438
2439 endmodule
```