Carlos Eduardo de Campos

Projeto de um Controlador de Acesso Direto a Memória de Multi Canais

Florianópolis, Santa Catarina

2011.1

Carlos Eduardo de Campos

Projeto de um Controlador de Acesso Direto a Memória de Multi Canais

Trabalho de Conclusão de Curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Ciências da Computação.

Orientador:
Prof. Dr. José Luís Almada Güntzel

UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Florianópolis, Santa Catarina

2011.1

Trabalho de conclusão de curso sob o título *"Projeto de um Controlador de Acesso Direto a Memória de Multi Canais"*, defendido por Carlos Eduardo de Campos e aprovado em 00 de Junho de 2011, em Florianópolis, Santa Catarina, pela banca examinadora constituída pelos professores:

Prof. Dr. José Luís Almada Güntzel
Universidade Federal de Santa Catarina
Orientador

Prof. Dr. Djones Vinícius Lettnin
Universidade Federal de Santa Catarina
Membro da Banca

Bel. Vinícius dos Santos Livramento
Universidade Federal de Santa Catarina
Membro da Banca

Agradecimentos

Em primeiro lugar agradeço a Deus por tudo que ele me proporcionou.

Agradeço ao Prof. Güntzel pela oportunidade e orientação do desenvolvimento deste trabalho.

Agradeço aos membros da banca pelo prestigio da avaliação.

Agradeço aos membros do laboratório LAPS, em especial ao Jucemar Monteiro que forneceu um grande auxílio na parte de extração de resultados.

Agradeço a minha família pelo apoio .

Agradeça as bolsas de inciação científica ITI-A do CNPq via projeto NAMITEC e SiSRas.

“Tudo posso naquele que me fortalece.”

Filipenses 4:13

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos p. 11

Resumo p. 12

Abstract p. 13

1 Introdução p. 14

1.1 Objetivos . . . . . p. 15

1.1.1 Objetivos Gerais . . . . . p. 15

1.1.2 Objetivos Específicos . . . . . p. 15

1.2 Escopo Deste Trabalho . . . . . p. 15

1.3 Organização do Trabalho . . . . . p. 15

2 Transferência de E/S p. 17

2.1 E/S Programado . . . . . p. 17

2.2 Interrupção Dirigida de E/S . . . . . p. 18

2.3 Acesso Direto à Memória . . . . . p. 18

2.3.1 *DMA Framework* . . . . . p. 18

2.3.2 Modelos de DMA . . . . . p. 20

2.3.2.1 Modelo de Endereçamento Simples . . . . . p. 20

2.3.2.2 Modelo de Endereçamento Duplo . . . . . p. 21

2.3.3 Modos de Transferência . . . . . p. 22
2.3.4 Tipo de Transferência . . . . . p. 23
2.3.5 Compatibilidade de Sistemas Heterogêneos . . . . . p. 24
2.3.6 Implementações Comerciais . . . . . p. 24

3 Projeto de um DMAC para Sistemas Embarcados p. 26
3.1 Especificações . . . . . p. 26
3.2 Comportamento . . . . . p. 27
3.2.1 Configuração . . . . . p. 28
3.2.2 Operação . . . . . p. 30
3.3 Arquitetura . . . . . p. 39
3.3.1 Decodificador . . . . . p. 40
3.3.2 Configuração do Canal . . . . . p. 41
3.3.3 *Datapath* . . . . . p. 42
3.3.4 Bloco de Controle . . . . . p. 47
3.4 Geração Automática de Descrições RTL de DMAC . . . . . p. 57

4 Descrição, Síntese e Validação do DMAC p. 59
4.1 Versões de DMAC Sintetizadas . . . . . p. 59
4.2 Síntese . . . . . p. 60
4.3 Validação . . . . . p. 61

5 Conclusão e Trabalhos Futuros p. 62
5.1 Conclusão . . . . . p. 62
5.2 Trabalhos Futuros . . . . . p. 64

Referências p. 66

Anexo A – Artigo p. 68

Anexo B – Código Fonte do Gerador Automático (PHP) p. 73

B.0.1 gera_tudo.php . . . . . p. 73

B.0.2 funcoes.php . . . . . p. 76

B.0.3 maquina_estados.php . . . . . p. 87

B.0.4 centro_operacao.php (*Datapath*) . . . . . p. 154

B.0.5 dados_canal.php . . . . . p. 188

B.0.6 decodificador.php . . . . . p. 202

B.0.7 dmac.php . . . . . p. 221

B.0.8 dmac.html . . . . . p. 337

Lista de Figuras

1 Modelo de Endereçamento Simples [Heath 2002]. . . . . p. 21

2 Modelo de Endereçamento Duplo [Heath 2002]. . . . . p. 21

3 Modo de Endereçamento 1D [Heath 2002]. . . . . p. 22

4 Modo de endereçamento 2D [Heath 2002]. . . . . p. 22

5 Modos de Transferência [Heath 2002]. . . . . p. 23

6 Comportamento do DMAC. . . . . p. 28

7 Comportamento configuração. . . . . p. 28

8 Comportamento configuração: PASSO_0. . . . . p. 29

9 Comportamento configuração: PASSO_1. . . . . p. 29

10 Comportamento configuração: PASSO_2. . . . . p. 30

11 Comportamento configuração. . . . . p. 31

12 Comportamento Operação: Início Transação. . . . . p. 31

13 Comportamento Operação: Adquire Barramento. . . . . p. 32

14 Comportamento Operação: Leitura. . . . . p. 33

15 Comportamento Operação: Verifica Mais Leitura. . . . . p. 34

16 Comportamento Operação: Escrita. . . . . p. 35

17 Comportamento Operação: Verifica Mais Escrita. . . . . p. 36

18 Comportamento Operação: Verifica Término. . . . . p. 37

19 Comportamento Operação: Libera barramento. . . . . p. 38

20 Comportamento Operação: término transação. . . . . p. 39

21 Arquitetura do DMAC. . . . . p. 40

22 Arquitetura: Decodificador. . . . . p. 41

23 Arquitetura: Configuração do Canal. . . . . p. 42

24 Controle de Incremento/Decremento do Registrador “addr_f”. . . . . p. 43

25 Controle de Incremento/Decremento do Registrador “addr_b”. . . . . p. 44

26 Controle de Decremento do Registrador “counter”. . . . . p. 45

27 Controle de Incremento/Decremento do Registrador “offset”. . . . . p. 46

28 Controle da área de memória “buffer”. . . . . p. 47

29 Bloco de Controle Inicial. . . . . p. 48

30 Bloco de Controle de *Status*. . . . . p. 49

31 Bloco de Controle Configuração. . . . . p. 50

32 Bloco de Controle: Operação. . . . . p. 51

33 Bloco de Controle: Inicio Transação. . . . . p. 53

34 Bloco de Controle: Leitura. . . . . p. 55

35 Bloco de Controle: Escrita. . . . . p. 56

36 Bloco de Controle: Finalização da Transação. . . . . p. 57

37 Interface do DMAC. . . . . p. 58

Lista de Tabelas

1 Síntese das Descrições em FPGA. . . . . p. 60

2 Síntese das descriçõesm em ASIC. . . . . p. 61

3 Comparativo de área. . . . . p. 63

4 Comparativo de potência. . . . . p. 63

5 Comparativo de frequência. . . . . p. 64

Lista de Acrônimos

ASIC *Application-Specific Integrated Circuit*

CRS *Control or Status Register*

DMA *Direct Memory Access* [Acesso Direto a Memória]

DMAC *Direct Memory Access Controller* [Controlador de Acesso Direto a Memória]

ES *Embedded Systems* [Sistemas Embarcados]

ESL *Electronic System-Level* [Nível de Sistema Eletrônico]

FPGA *Field-Programmable Gate Array*

HDL *Hardware Description Language* [linguagem de descrição de hardware]

IEEE *Institute of Electrical and Electronic Engineers*

E/S Entrada / Saída

RTL *Register-Transfer Level* [Nível de Transferência entre Registradores]

SoC *System-on-Chip* [Sistema em *chip*]

UART *Universal Asynchronous Receiver/Transmitter*

USB *Universal Serial Bus*

Resumo

Sistemas Embarcados (ES - *Embedded Systems*) são sistemas que possuem em sua composição sistemas computacionais desenvolvidos para atender tarefas especificas com restrições iguais ou mais rígidas do que computadores genéricos, geralmente embutidos em um produto maior e não visíveis ao usuário. Um sistema em *chip* (SoC - *System-on-Chip*) contém componentes computacionais como processador, memória, controlador de acesso direto a memória (DMAC - *Direct Memory Access Controller*), dispositivos de entrada / saída integrados em um único chip. Exemplos de aplicações que utilizam SoC em ES são: injeção eletrônica, microondas, sistemas médicos, entre outros.

O trabalho implementa um dispositivo de acesso direto a memória (DMA - *Direct Memory Access*, cujo modelo de DMA conta com um dispositivo dedicado para tratar requisições de transferência, denominado DMAC. Necessitando do processador apenas nas etapa de configuração do canal, no início e no término da transferência, estando livre durante a ocorrência da transferência para executar outras operações.

Assim, o resultado principal deste trabalho foi a caracterização do comportamento funcional da *interface* do DMAC, definido por um conjunto de estímulos aplicado às suas entradas e o respectivo conjunto de resultados às suas saídas. Tal caracterização é crucial para o projeto de validação do DMAC em nível de transferência entre registradores (RTL - *Register-Transfer Level*) com vistas à sua prototipação em *Field-Programmable Gate Array* (FPGA) e em silício através de um circuito integrade de aplicação especifica (ASIC - *Application-Specific Integrated Circuit*), as quais são objeto do trabalho.

Abstract

Embedded Systems(ES) are computational systems developed to attend specif tasks that have the same restrictions or be stricter then home computers and be, usually, embedded in a larger product and not visible to the user. An integrated system System on Chip (SoC) contains computational components like processor, memory, DMAC, input and output devices integrated in just one chip. A typical application of a SoC is in the ES area. Examples of applications that use SoC are: fuel injection, microwave, medical systems, among others.

This work implements a direct memory access device, whose DMA model has a dedicated device to treat transfer requests, called direct memory access controller (DMAC). The processor is needed only in steps of channel configuration, at the beginning and end of transfers, and is free during transfer occurrence to execute others operations.

Therefore, the main result of this work was the functional behavior characterization of DMAC interface, defined for a stimuli set applied to its inputs and the respective results set to its outputs. This characterization is crucial for the DMAC validation project with level RT to its prototyping in FPGA and silicon, that are this work object.

1 Introdução

Sistemas Embarcados (ES - *Embedded Systems*) são sistemas que possuem em sua composição sistemas computacionais desenvolvidos para atender tarefas especificas com restrições iguais ou mais rígidas do que computadores genéricos, geralmente embutidos em um produto maior e não visíveis ao usuário. Um sistema em *chip* (SoC - *System-on-Chip*) contém componentes computacionais como processador, memória, controlador de acesso direto a memória (DMAC - *Direct Memory Access Controller*), dispositivos de entrada / saída integrados em um único chip. Exemplos de aplicações que utilizam SoC em ES são: injeção eletrônica, microondas, sistemas médicos, entre outros.

Um paradigma de projeto é o projeto baseado em plataforma, consiste de um projeto genérico que pode ser personalizado de acordo com o determinado domínio de aplicação. Os blocos que compõem uma plataforma podem ser descritos em diferentes níveis de abstração como, por exemplo: nível de transistores, nível de porta lógica, Nível de Sistema Eletrônico (ESL - *Electronic System-Level*) e Nível de Transferência entre Registradores (RTL - *Register-Transfer Level*). Este último descreve um componente sendo formado geralmente por um *datapath* comandado por uma unidade de controle, tal descrição é feita em uma linguagem de descrição de *hardware* (HDL - *Hardware Description Language*) como, por exemplo, a linguagem VHDL.

Foram estudados técnicas de implementação de controladores de acesso direto a memória, tanto o projeto de *hardware* quanto o refinamento da máquina de estados a fim de encontrar o melhor compromisso dos requisitos do projeto, voltados ao consumo de energia e vazão de dados. A arquitetura base do projeto é baseada em pesquisas das arquiteturas de *hardware* encontradas em sistemas embarcados, sendo gerado um protótipo de DMAC com saída para 2 barramentos de dados e endereços, 4 canais e um *buffer* interno de 1Kb. Baseado no protótipo foi elaborado um gerador automático em PHP que permite ao projetista selecionar o número de canais e o endereço ao qual o DMAC será mapeado, obtendo a descrição em VHDL automatizada.

Concluída a geração automática da descrição VHDL, a descrição foi validada por *testbench*

e sintetizada para FPGA e ASIC, a fim de obter resultados que permitam avaliar o projeto em termos de recursos lógicos, potência e frequência máxima obtida.

1.1 Objetivos

1.1.1 Objetivos Gerais

O objetivo deste trabalho é a especificação, projeto e verificação de um DMAC multicanal para sistemas embarcados.

1.1.2 Objetivos Específicos

- Criar uma especificação de DMAC compatível com os sistemas embarcados.
- Propor uma arquitetura em RTL.
- Descrição da arquitetura RTL em VHDL.
- Validar o DMAC por meio de verificação funcional usando *testbench*.
- Síntese da arquitetura RTL para FPGA contemporâneos a fim de obter resultados que permitam avaliar o projeto em termos de recursos lógicos, potência e frequência máxima obtida.
- Síntese da arquitetura RTL para ASIC utilizando uma ferramenta de síntese comercial para realizar as etapas de síntese lógica e física, a fim se obter resultados que permitam avaliar o projeto em termos de área, potência e frequência máxima obtida.

1.2 Escopo Deste Trabalho

Este trabalho envolve a concepção de um DMAC a ser utilizado em sistemas embarcados, iniciando pela sua especificação, e chegando até a proposta de uma arquitetura RTL.

1.3 Organização do Trabalho

O trabalho está organizado da seguinte forma. No **Capítulo 2** é efetuada uma revisão bibliografica sobre meios de controle de Entrada / Saida (E/S), contextualizando o funcionamento

do DMAC e introduzindo conceitos básicos envolvidos. No **Capítulo 3** são expostas as especificações e comportamento do projeto do DMAC. No **Capítulo 4** são apresentadas as descrições RTL geradas, os resultados de síntese e a validação do DMAC e no **Capítulo 5** são apresentadas as devidas conclusões e menções a trabalhos futuros.

2 *Transferência de E/S*

A comunicação entre os diversos blocos que compõem os sistemas computacionais se dá através de um meio físico comum, o barramento. Dentre os blocos do sistema computacional, alguns são dedicados à comunicação com outros sistemas, apresentando, por este motivo, velocidades e vazões (*throughputs*) bastante diversas. Como a comunicação entre todos os blocos exige o uso do barramento, e como os blocos com maior diversidade de velocidades e vazões são os blocos que permitem transferências de entrada e de saída, políticas de transferência de entrada e saída (E/S) precisam estar presentes nos sistemas computacionais. Cada política é adequada às características próprias de um subconjunto de componentes, de modo a maximizar o desempenho do sistema. Existem basicamente três políticas de E/S: E/S programada, Interrupção dirigida de E/S e Acesso Direto à Memória (DMA - *Direct Memory Access*) [Patterson e Hennessy 2007]. Nas próximas seções estas políticas são explicadas, sendo dada maior atenção à DMA, uma vez que esta política é o foco do presente trabalho.

2.1 E/S Programado

Esta política não utiliza nenhum *hardware* dedicado e, para execução da transferência são utilizados registradores de controle. O processador possui controle sobre as operações de E/S, desde a detecção do estado do dispositivo, o envio de comandos de leitura ou comandos de escrita até a transferência de dados. O mecanismo de transferência é ativado ao necessitar realizar alguma operação de E/S, o processador verifica o registrador de *status* do dispositivo até que o mesmo esteja pronto. Quando pronto, o processador efetua a transferência de um dado (leitura ou escrita) e após, volta a ler o *status* do dispositivo. Este processo é repetido até que todas as transferências sejam realizadas. Neste método o processador gasta muitos ciclos para fazer uma única transferência devido à diferença de velocidades de funcionamento entre o dispositivo e o processador, ficando este grande parte do tempo ocioso [Null e Lobur 2006] [Patterson e Hennessy 2007].

2.2 Interrupção Dirigida de E/S

Esta política conta com um dispositivo dedicado para tratar as interrupções, denominado controlador de interrupções. O controlador possui canais de acesso para os quais os dispositivos são devidamente mapeados e, ao necessitar realizar uma transferência, um sinal é emitido ao controlador. Ao receber uma interrupção, dependendo do estado, o processador muda de contexto para atender à interrupção, então executando as instruções relativas à interrupção. Após realizá-las, o contexto é restaurado e o processador continua sua execução normal. Ao finalizar a transferência, uma nova interrupção é sinalizada e, dependendo do estado do sistema, o processador muda o contexto, tratada a interrupção e restaura o contexto prosseguindo a execução [Null e Lobur 2006].

2.3 Acesso Direto à Memória

O acesso direto a memória (DMA - *Direct Memory Access*) exige um dispositivo dedicado para tratar requisições de transferência, denominado controlador de acesso direto a memória (DMAC - *Direct Memory Access Controller*). Nesta política de transferência, o processador tem contato com o DMAC na etapa de configuração do canal, no início e no término da transferência, ficando livre para executar outras operações enquanto a transferência ocorre [Vahid e Givargis 2002]. Para efetuar a configuração, é necessário informar um endereço de origem, no qual será efetuada a leitura, informar um endereço de destino, no qual será efetuada a escrita, informar o número de palavras a transmitir e informar os sinais de configuração específicos da transação, como tamanho da palavra, modo de transferência, entre outros. A seguir é descrito um *framework* de DMA bem como modelos de implementação, modo de operação, tipos de transferência e implementações comerciais.

2.3.1 *DMA Framework*

O mercado quando não regularizado é pré-disposto a criação de próprios padrões, dificultando a interconectabilidade de produtos de diferentes fabricantes. Para aumentar a chance de compatibilidade entre produtos de diversos fabricantes a *Institute of Electrical and Electronic Engineers* (IEEE) criou a IEEE Std 1212.1-1993 [Crowell 1994], um *framework* de comunicação entre processadores e dispositivos, utilizando memória compartilhada, estando incluso o DMAC. Este *framework* define um conjunto de características desejáveis para os sistemas computacionais, no que se refere às transferências entre os componentes, incluindo as caracter-

isticas dos DMACs.
Estas caracteristicas são:

1. Flexibilidade na configuração: deve ser capaz de ser implementado em uma grande variedade de configurações de sistema, incluindo as seguintes:

   (a) Ampla variedade de dispositivos: Além de suportar ampla variedade de dispositivos, bem como transferências memória - memória. Estes dispositivos devem ser suportados diretamentre ou indiretamente, usando *links* de comunicação compartilhada como SCSI ou IPI;

   (b) Sistema de *cache* : Permite ao dispositivo utilizar de forma eficiente, o sistema de coerência de *cache*, caso houver;

   (c) Acessibilidade: Suportar desde processadores de última geração a dispositivos antigos

   (d) Ambiente Compatível : Ser compatível com o modelo de transação do IEEE Std 1212-1991;

   (e) Endereçamento físico : Suportar endereçamento físico e virtual;

   (f) Tamanho do sistema: Suportar desde sistemas de mono processador à sistemas multiprocessados;

   (g) Confiabilidade : E/S são alimentados e possuem reinicialização independente;

   (h) Retro compatibilidade: Mesmo havendo alterações nos circuitos envolvidos, se seguirem o IEEE Std 1212.1-1993 devem ser suportados;

2. Flexibilidade na arquitetura - Suportar eficientemente uma grande variedade de dispositivos de E/S que abrange os requisitos abaixo, mas não limita-se a eles;

   (a) Transações de E/S : Um canal pode ser associado a qualquer dispositivo

   (b) Interrupções: Suportar a entrada de sinais síncronas e assíncronas

   (c) Memória contígua: Não utilizar estruturas de dados que ultrapassem o tamanho padrão de uma página de dados (4 Kbytes);

3. Desempenho - O projeto de DMAC deve ter desempenho igual ou superior aos sistemas atuais.

   (a) Leitura CRS : Evitar ao máximo que o processador tenha que verificar o estado do dispositivo;

(b) Escritas CRS :Solicitar ao processador para escrever no máximo uma vez por transação de E/S na unidade de CSR;

(c) Tamanho de transferência: Suportar eficientemente transações de todos os tamanhos.

(d) Contador de transação: Minimizar o número de transações no barramento, pois não necessariamente cada transação faz parte de uma transferência de dados contíguos, havendo uma carga extra na configuração;

(e) Contador de interrupção: Em caso de muitas requisições de interrupção, devem ser combinadas em uma só;

(f) Interrupção eficiente: O DMA deve prover informações suficientes para o processador disparar a rotina adequada de forma rápida.

(g) *Overhead* : As transferências DMA devem apresentar o menor custo possível;

(h) Otimização: Otimizar o caso mais comum e fornecer cobertura aos piores casos;

2.3.2 Modelos de DMA

Há vários modelos de DMA que podem ser implementados, indo desde modelos mais simples até modelos mais complexos.

2.3.2.1 Modelo de Endereçamento Simples

No modelo de endereçamento simples o DMAC indica uma posição de memória, que irá atuar no ciclo de memória. O DMA utiliza o sinal de leitura/escrita para ativar o dispositivo durante o ciclo. Dessa forma, os dados podem ser transferidos sem passar por dentro do DMAC. Esse modo também é conhecido como endereçamento implícito [Heath 2002]. A figura 1 ilustra o conceito.

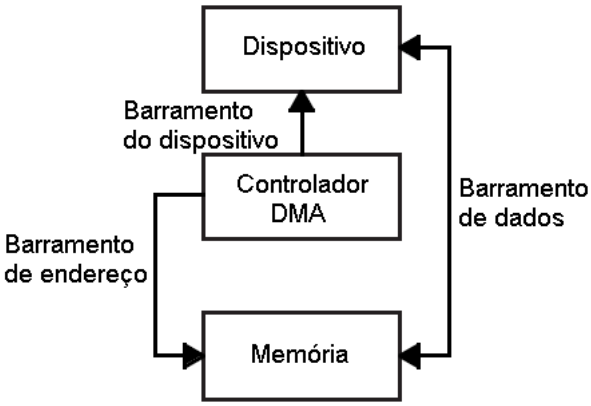


Figura 1: Modelo de Endereçamento Simples [Heath 2002].

2.3.2.2 Modelo de Endereçamento Duplo

O modelo de endereçamento duplo utiliza dois barramentos. Neste modelo o dado irá passar por dentro do DMAC [Heath 2002], sendo portanto necessário guardar o dado temporariamente, o que incorre em mais recursos de *hardware* e maior tempo de transfferência. Este modelo é ilustrado na Figura 2. No modelo de endereçamento duplo há duas formas de acessar a menória, as quais são definidas pelo modelo 1D e 2D.

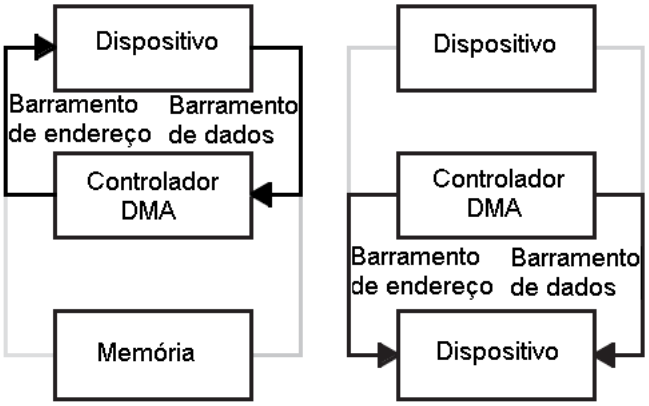


Figura 2: Modelo de Endereçamento Duplo [Heath 2002].

O modelo 1D assume uma faixa de endereço e, ilustrado na Figura 3, dada a transferência, irá escrever os dados na sequência da faixa de endereços. Esse conceito é originalmente para o *buffer*, porém, pode ser generalizado no processo de escrita do DMA.
Uma desvantagem desse uso é que ao terminar a transação, geralmente o contador é reinicializado e ao iniciar uma nova transação os dados podem ser sobrescritos [Heath 2002].

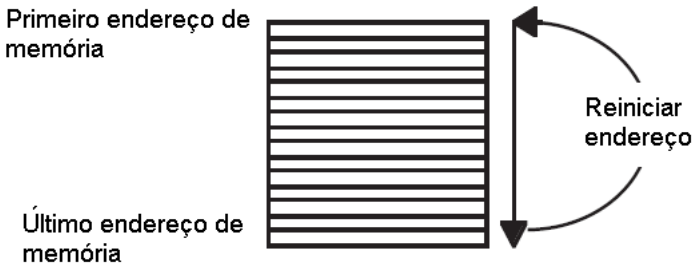


Figura 3: Modo de Endereçamento 1D [Heath 2002].

O modelo 2D adiciona um conceito de passada, onde o passo de incremento é adicionado ao endereço base no final de uma contagem, fazendo o DMA retransmitir dados não consecutivos. Ao terminar a transação, em vez reinicializado, o endereço base é somado a um deslocamento. Geralmente, há 2 registradores de controle. Este modelo é ilustrado na Figura 4 [Heath 2002]. Uma utilização para este modelo é a seleção de segmentos de dados recebidos, agindo como filtro de cabeçalhos, entregando somente o conteúdo.

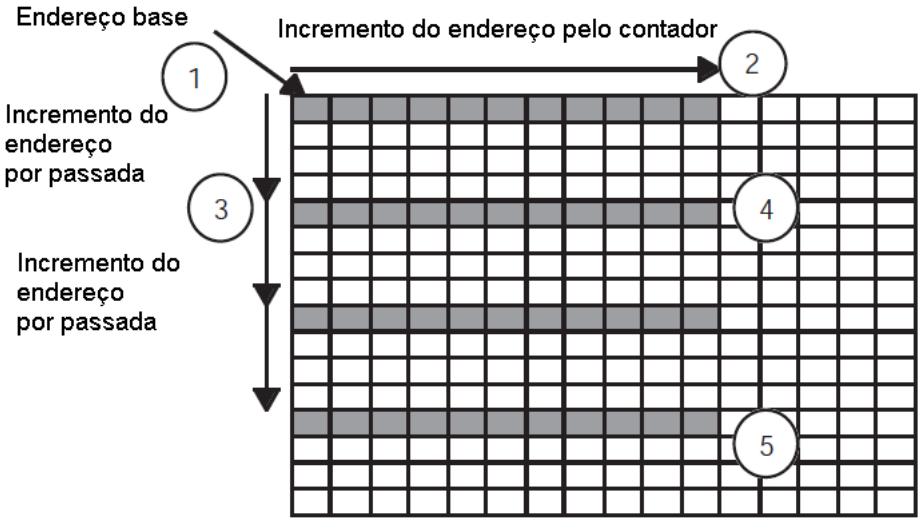


Figura 4: Modo de endereçamento 2D [Heath 2002].

2.3.3 Modos de Transferência

Um DMAC pode oferecer até três modos básicos de operação, os quais visam atender eficientemente cada classe de dispositivo. As nomenclaturas variam dependendo da bibliográfia. Neste trabalho foi assumida a nomenclatura utilizada por [Heath 2002]: Modo de Transferência Simples, Modo de Transferência em Bloco e Transferência de Transferência por Demanda.

- Modo de Transferência Simples:

Efetua uma transferência por vez. Ao receber uma requisição, o DMAC requisita a posse do barramento, ao consegui-lo, é efetuado a leitura de um dado da origem indicada, logo, o dado lido é escrito no destino indicado. Efetuada a transfêrencia de um dado, o DMAC devolve a posse do barramento ao processador e, havendo mais dados a serem transmitidos, o DMAC volta a requisitar a posse do barramento, repetindo este processo até concluir todas as transfêrencias ao qual foi programado. A Figura 5 ilustra o processo.

- Transferência em Bloco:
  Efetua a transação de uma só vez. Ao receber uma requisição, o DMAC requisita a posse do barramento, ao consegui-lo, é efetuado a leitura de um dado da origem indicada, logo, o dado lido é escrito no destino indicado. Efetuada a transfêrencia de um dado, o DMAC inicia o processo de mais uma transação, repetindo este processo até concluir todas as transfêrencias ao qual foi programado. A Figura 5 ilustra o processo.

- Transferência sob Demanda:
  Efetua a transação de uma só vez.Ao receber uma requisição, o DMAC requisita a posse do barramento, ao consegui-lo, é efetuado a leitura de um dado da origem indicada, dependendo da origem, quando o dispositivo sinalizar estar pronto, logo, o dado lido é escrito no destino indicado, dependendo do destino, quando o dispositivo sinalizar estar pronto. Efetuada a transfêrencia de um dado, o DMAC inicia o processo de mais uma transação, repetindo este processo até concluir todas as transfêrencias ao qual foi programado. A Figura 5 ilustra o processo.

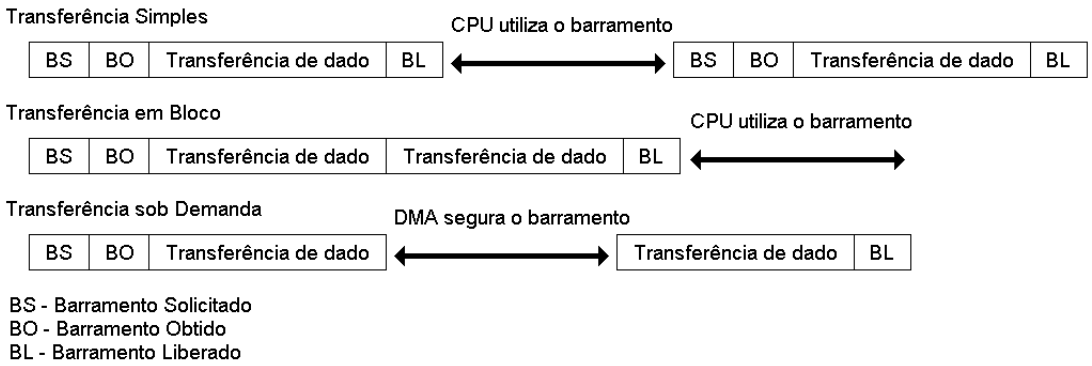


Figura 5: Modos de Transferência [Heath 2002].

2.3.4 Tipo de Transferência

O DMA pode oferecer até 4 tipos de transação: Memória para Memória, Memória para Dispositivo, Dispositivo para Memória e Dispositivo para Dispositivo.

- Memória para Memória:
  Esse modo nem sempre é disponibilizado, pois necessita que o dado seja segurado por um ciclo. Para tanto, o DMAC deve possuir *buffers* internos. Consiste no DMAC ler um dado da memória do endereço de origem, armazenar temporariamente o dado e após, escrever na memória no endereço destino o dado armazenado.

- Memória para Dispositivo: Consiste no DMAC gerar uma ordem de leitura à memória e uma ordem de escrita ao dispositivo. Dependendo das configurações, o DMAC pode armazenar temporariamente o dado durante o ciclo de transferência. Ou não.

- Dispositivo para Memória: Consiste no DMAC gerar uma ordem de leitura ao dispositivo e uma ordem de escrita à memória. Dependendo das configurações, o DMAC pode armazenar temporariamente o dado durante o ciclo de transferência. Ou não.

- Dispositivo para Dispositivo: Consiste no DMAC gerar uma ordem de leitura ao dispositivo e uma ordem de escrita ao dispositivo. Dependendo das configurações, o DMAC pode armazenar temporariamente o dado durante o ciclo de transferência. Ou não.

2.3.5 Compatibilidade de Sistemas Heterogêneos

O DMAC pode ter contato com vários sistemas com diferentes tamanhos de palavras. O controlador deve servir de ponte e conseguir fazer esses sistemas conversarem entre sí. Uma das formas possíveis de efetuar essa conversação é assumir que a troca de informações aconteça no tamanho da palavra menor. Por exemplo: um dispositivo de 16 bits conversando com o processador de 32 bits. A saída do dispositivo só consegue endereçar 16 bits, para conseguir comunicação, o processador deve escrever as informações em múltiplos de 16 bits, dependendo da arquitetura nos bits menos significatovos.

2.3.6 Implementações Comerciais

Uma implementação comercial consolidada é o Intel 8237/8237A [Intel 2011]. O intel 8237A é um DMAC de 4 canais fixos com barramento de dados e endereço de 16 bits. Ele permite cascatear controladores a fim de atender mais canais. Esta implementação tornou-se uma referência para geração de outros controladores.

A empresa Lattice possui um DMAC de multi canais, variando entre 1 a 32 canais. Permite definição do tamanho do barramento de dados e endereço além de possúir um módulo de compatibilidade com o Intel 8237A.

A empresa Inomize possui um DMAC de multi canais, variando entre 1 a 10 canais, atendendo até 32 dispositivos. O produto utiliza o barramento AMBA (Advanced High Performance Busses).

A empresa ARM possui um DMAC, chamado *PrimeCell DMA Controller*. O produto utiliza o barramento AMBA

3 *Projeto de um DMAC para Sistemas Embarcados*

Este capítulo apresenta as específicações de DMAC e seu comportamento, em um nivel de abstração de eventos. Também apresenta a arquitetura do DMAC. Finalmente, apresenta as principais caracteristicas de um gerador de descrições RTL que segue as caracteristicas de DMAC, usada para a criação de descrições de DMAC com mais canais.

3.1 Especificações

As especificações do DMAC para sistemas embarcados teve como base o DMAC do projeto Dalton [Vahid 2011], e refinadas com base em artigos [Oh Byeong-Deok Kim 2008] [O'Nils e Jantsch 1999] [Yuan, Chen e Bai 2004] [Newman e Smith 1981], livros [Vahid e Givargis 2002] [Martin e Bailey Brian 2007] [Null e Lobur 2006] [Stallings 2002] [Dandamudi 2003] [Heath 2002] [HENNESSY e PATTERSON 2003] [Wolf e Kaufmann 2001], materiais de ensino [Mano 2011] [Ball 2011] e *framework* de DMAC da IEEE 1212.1-1993 [Crowell 1994], resultando nas características abaixo:

- **Numero de canais:** Como os sistemas computacionais embarcados podem possuir mais de uma dezena de blocos que necessitem realizar transferências de dados com memória ou entre si, optou-se por definir um número de canais entre 1 e 32. Assim, o projeto baseou-se no princípio da modularidade, de modo a facilitar a adaptação da arquitetura para um número distinto de canais (desde que este número esteja entre 1 e 32). A ferramenta criada para gerar descrições RTL da arquitetura de DMAC (ver seção 3.4) possui como um de seus parâmetros de entrada o número de canais.

- **Ordem de prioridade:** No projeto do DMAC foi estipulada a ordem de prioridade sendo fixa, tendo o canal de ordem mais significativa a menor prioridade.

- **Barramento:** Dois barramentos, sendo um o barramento do CPU (*Front-end*) e outro o barramento de E/S (*Back-end*). A escolha foi uma decisão de projeto visando não compro-

meter a velocidade do barramento do CPU em detrimento da velocidade dos dispositivos de E/S.

- **Tamanho da Palavra no barramento de Entrada / Saída (E/S) (*Back-end*):** Suporte a 8, 16, 32 e 64 bits. Fornece ao sistema suporte para comunicar sistemas com tamanhos de palavras de 8, 16, 32 e 64 bits.

- **Tamanho da Palavra no barramento da CPU (*Front-end*):** Suporte 32 bits. Uma decisão de projeto que entende que no barramento *front-end* somente irá haver transações de 32 bits, isto gera simplificações na lógica resultando em menos componentes lógicos.

- **Modo de transferência:** Suporte aos três modos: Simples, Bloco e Demanda. O suporte as três modos de endereçamento aumenta o potencial de aplicação do DMAC aos sistemas embarcados.

- **Número de transferências por transação:** 2^{16} transferências. Por decisão de projeto o registrador interno que efetua o gerenciamento de transações possui 16 bits.

- **Tamanho do *buffer* interno:** 1Kb. O uso de *buffer* interno permite a transferência memória-memória, bastante comum em sistemas computacionais embarcados. O tamanho de 1kb permite que blocos de tamanho razoável, no contexto de sistemas embarcados, sejam armazenados.

- **Modo de Operação:** Suporte aos 4 modos: Memória para memória; Memória para E/S; E/S para memória; E/S para E/S.

3.2 Comportamento

O DMAC apresenta dois comportamentos básicos, denominados de CONFIGURAÇÃO e OPERAÇÃO (Figura 6), os quais podem ser atingidos a partir da do estado de inicialização. O DMAC permanece no estado de espera enquanto não ocorre nenhum evento que ative um destes comportamentos. Os eventos que ativam cada um dos dois comportamentos são detalhados a seguir.

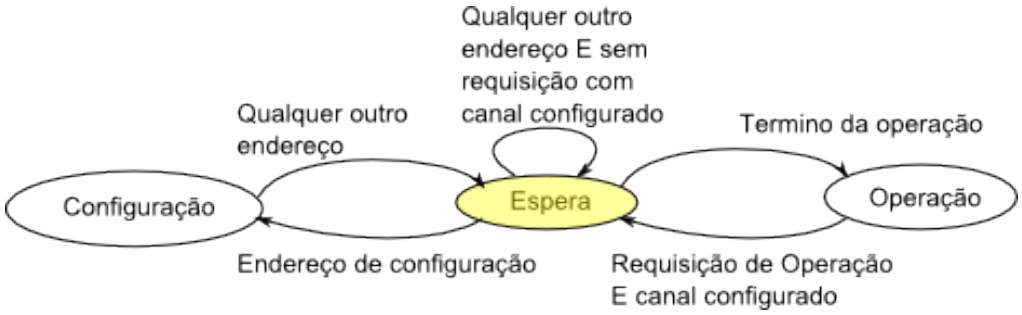


Figura 6: Comportamento do DMAC.

3.2.1 Configuração

O comportamento CONFIGURAÇÃO modela o DMAC quando é efetuada/alterada a configuração de um canal. Sua ativação ocorre quando o DMAC encontra-se no estado de espera e é escrito um endereço no barramento de endereço superior ao qual o DMAC foi mapeado, conforme ilustra a Figura 7

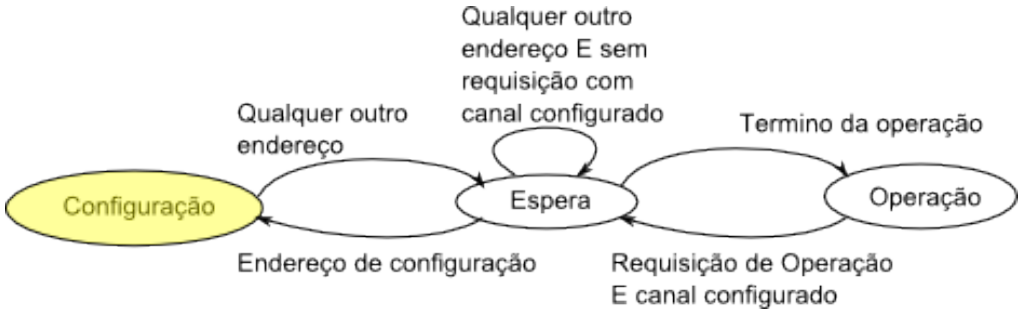


Figura 7: Comportamento configuração.

Ao iniciar o comportamento de configuração o endereço é lido e a parte menos significativa é decodificada. Para determinar o canal alvo da configuração são lidos do bit 6 ao 2 e para determinar o endereço da configuração são lidos do bit 1 ao 0.

Para sair do estado de ESPERA e ir ao estado PASSO_0, o endereço obrigatoriamente deve ter os bits um e zero em 0, a fim de definir que se trata de um endereço de configuração 0. Enquanto o endereço permanecer inalterado, o sistema permanece no estado PASSO_0. Durante o estado PASSO_0, o barramento de dados superior é lido e armazenado em um registrador interno do decodificador. O dado lido representa a configuração do endereço-base do barramento *Front-end*. A Figura 8 ilustra o processo.

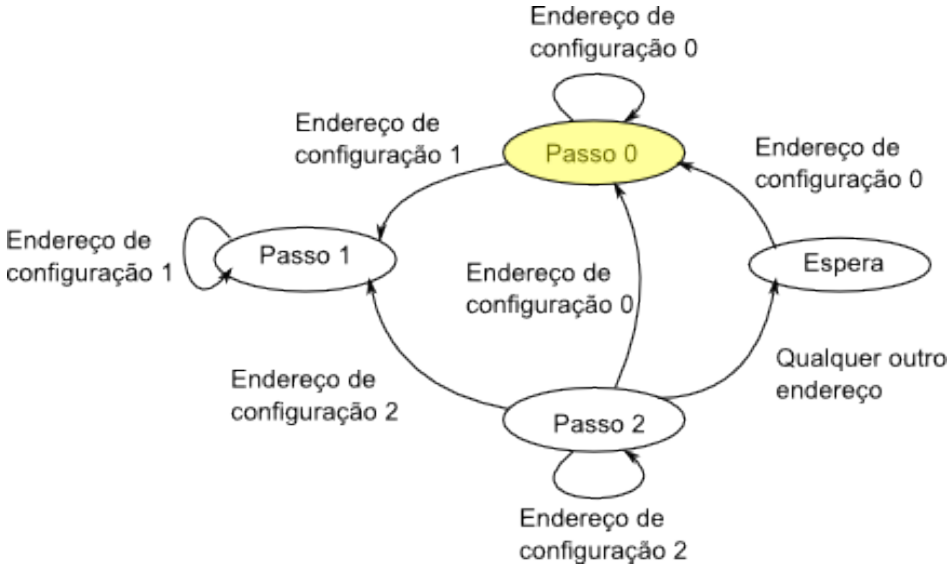


Figura 8: Comportamento configuração: PASSO_0.

Ao receber o próximo endereço de configuração, e sendo este um endereço válido, para chegar ao estado PASSO_1 o bit um deve ser 0 e o bit zero deve ser 1, definindo o endereço de configuração 1. Além disso, o bit 6 ao bit 2 devem ser iguais ao endereço anterior. Enquanto o endereço permanecer inalterado, o sistema permanece no mesmo estado. Durante o estado PASSO_1 o barramento de dados superior é lido e armazenado em um registrador interno do decodificador. O dado lido representa a configuração do endereço base do barramento *Back-end*. A Figura 9 ilustra o processo.

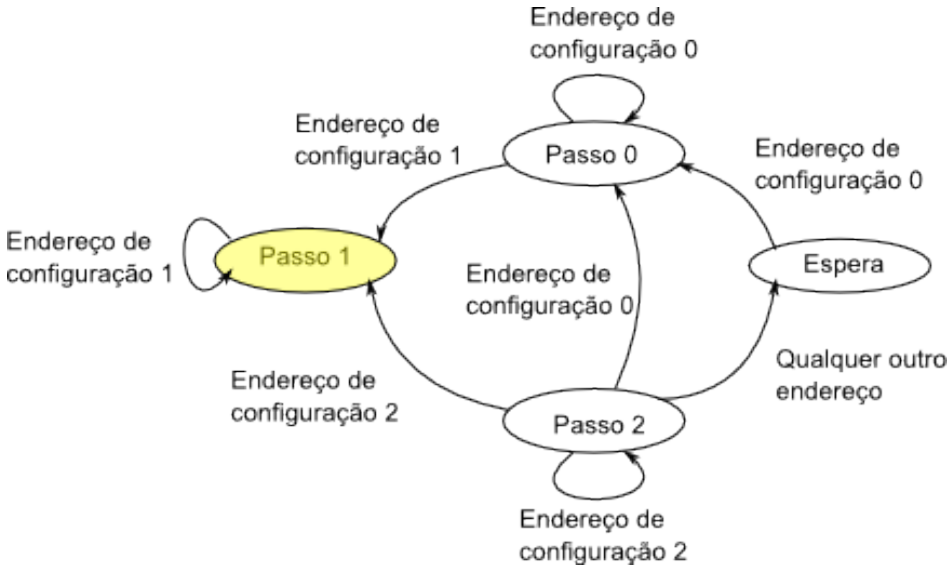


Figura 9: Comportamento configuração: PASSO_1.

Ao receber o próximo endereço de configuração, e sendo este um endereço válido, para

chegar ao estado PASSO_2, o bit um deve ser 1 e o bit zero deve ser 0, definindo o endereço de configuração 2 e do bit 6 ao bit 2 devem ser iguais aoendereço anterior. Enquanto o endereço permanecer inalterado, o sistema permanece no mesmo estado. Durante o estado PASSO_2 o barramento de dados superior é lido e o valor é armazenado em um registrador interno do decodificador. Os 16 bits mais significativos do dado lido representam o número de transações a efetuar e os 16 bits menos significativos representam as sinalizações de configuração do canal. Os bits 1 e 0 representam o tamanho da palavra associada ao barramento *back-end*. Os bits 3 e 2 representam o modo de operação, os bits 5 e 4 identificam o tipo de transação, o bit 6 identifica o incremento (0) ou decremento (1) do endereço *front-end*, o bit 7 identifica o incremento (0) ou decremento (1) do endereço *back-end*, o bit 8 identifica o não bloqueio de incremento/decremento (0) ou o bloqueio de incremento/decremento (1) do endereço *front-end*. O bit 9 identifica o não bloqueio de incremento/decremento (0) ou o bloqueio de incremento/decremento (1) do endereço *back-end* e dos bits 15 ao 10 não são utilizados. Neste estado, o canal é marcado como configurado e o local onde os dados de configuração do canal são armazenados é atualizado. A Figura 10 ilustra o processo lógico.

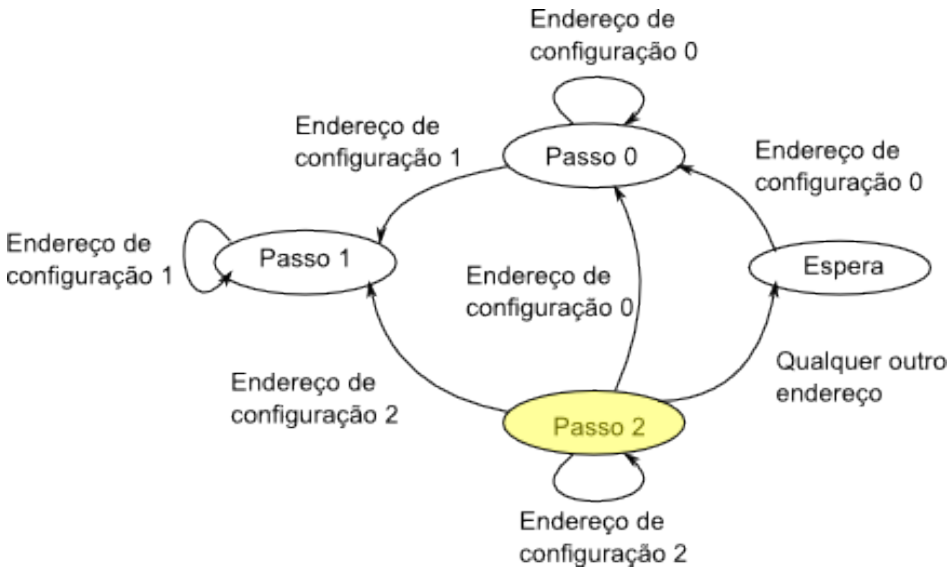


Figura 10: Comportamento configuração: PASSO_2.

3.2.2 Operação

O comportamento OPERAÇÃO modela o DMAC quando uma requisição válida é processada, sua ativação ocorre quando o DMAC está no estado de espera e a requisição válida é processada, conforme ilustra a Figura 11.

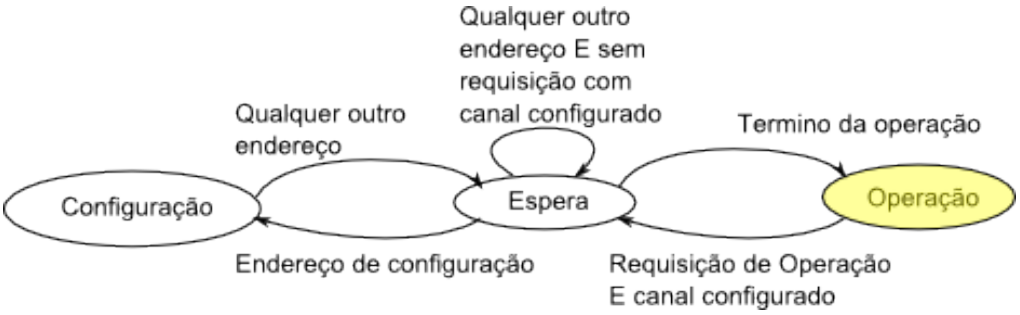


Figura 11: Comportamento configuração.

Ao receber uma requisição (sinal DREQ[i] = 1, onde i é o canal associado à requisição, e o canal associado já tendo sido configurado), o DMAC carrega as configurações do devido canal para os registradores do *datapath*. A Figura 12 ilustra o processo lógico.

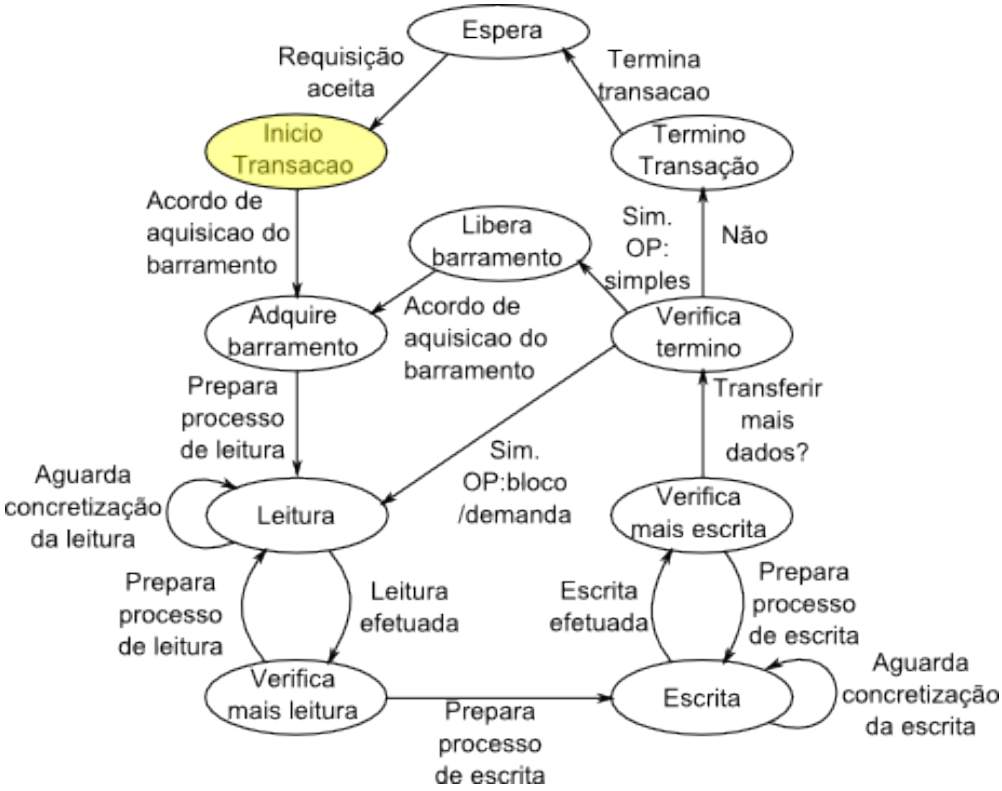


Figura 12: Comportamento Operação: Início Transação.

Após os dados terem sido carregados, o DMAC irá solicitar à CPU o controle do barramento *front-end* fazendo o sinal HOLD=1. Ao liberar o barramento para o DMAC, a CPU fará o sinal HLDA=1. Com posse do barramento, o DMAC irá enviar o aceite para o requisitante através do sinal DACK[i] logo após preparar os sinais para efetuar a leitura (fazendo MEN_R=1 ou IO_R=1) e escrever o endereço no respectivo barramento. A Figura 13 ilustra o processo lógico.

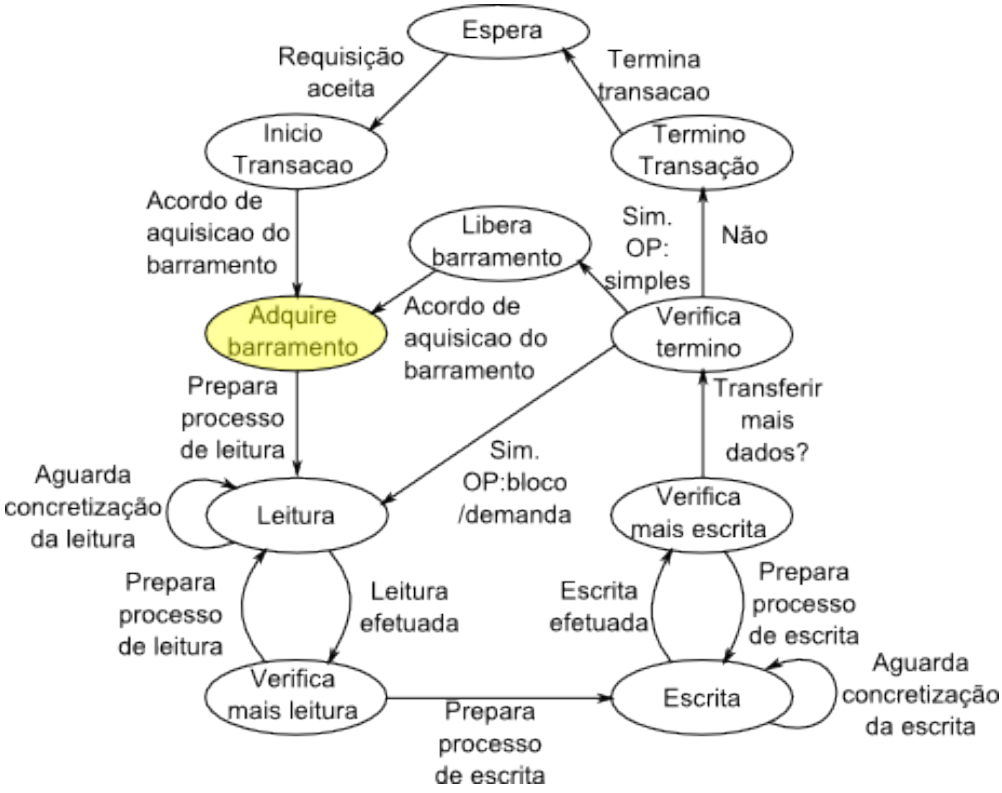


Figura 13: Comportamento Operação: Adquire Barramento.

Depois de preparado o processo de leitura, o DMAC permanece no estado de leitura, aguardando sua concretização. Para isso, o DMAC fica monitorando o barramento de dados, no qual foi escrito o endereço de leitura. Ao receber o dado, ele é armazenado no *buffer* interno do DMAC e este avança para o estado de verificação de mais leitura, desabilitando a escrita do endereço e baixando o sinal de controle de leitura. A Figura 14 ilustra o processo lógico.

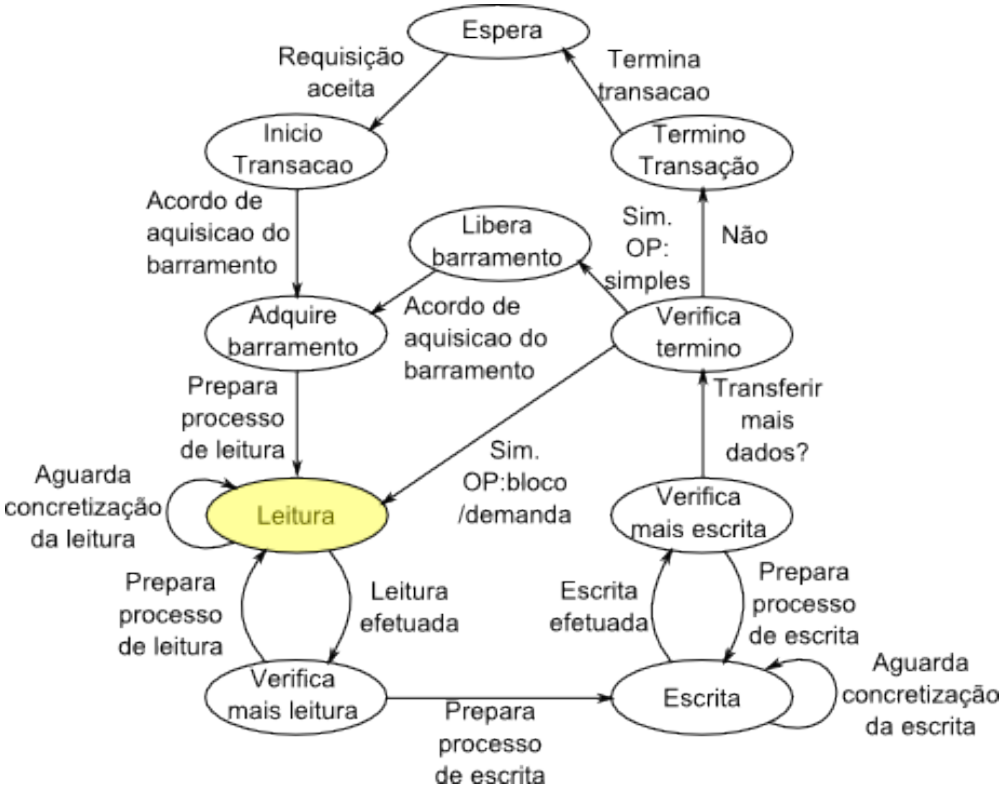


Figura 14: Comportamento Operação: Leitura.

Baixados os sinais de leitura e concluída a escrita do endereço, o controlador do *buffer* é incrementado e o número de transferências a realizar é decrementado. Logo após, o modo de operação é conferido. Caso a operação seja do tipo bloco ou demanda, os sinais de leitura são reabilitados e o endereço é escrito no devido barramento, efetuando as preparações para mais uma leitura. Este ciclo ocorrerá até o *buffer* encher ou o número de transferência chegar à zero. Caso o modo de transferência seja simples ou o *buffer* estiver cheio ou o número de transferências tiver chegado a zero, o DMAC irá escrever o endereço de escrita e o dado nos devidos barramentos, e também ativará os devidos sinais para efetuar a escrita. A Figura 15 ilustra o processo lógico.

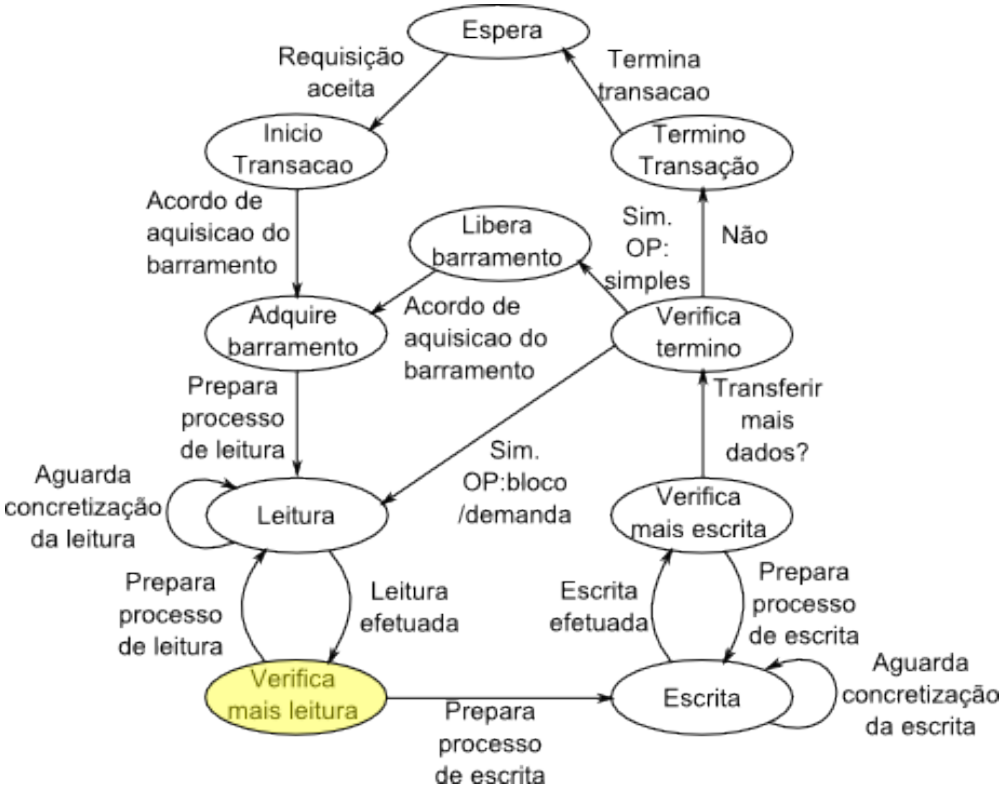


Figura 15: Comportamento Operação: Verifica Mais Leitura.

Depois de todos os sinais estarem devidamente ajustados para efetuar a escrita, o DMAC verifica o sinal READY. Enquanto este sinal estiver desabilitado, os sinais de escrita são mantidos. Após constatar a habilitação do sinal, os sinais de escrita são desabilitados, bem como a escrita do endereço e do dado no devido barramento. A Figura 16 ilustra o conceito lógico.

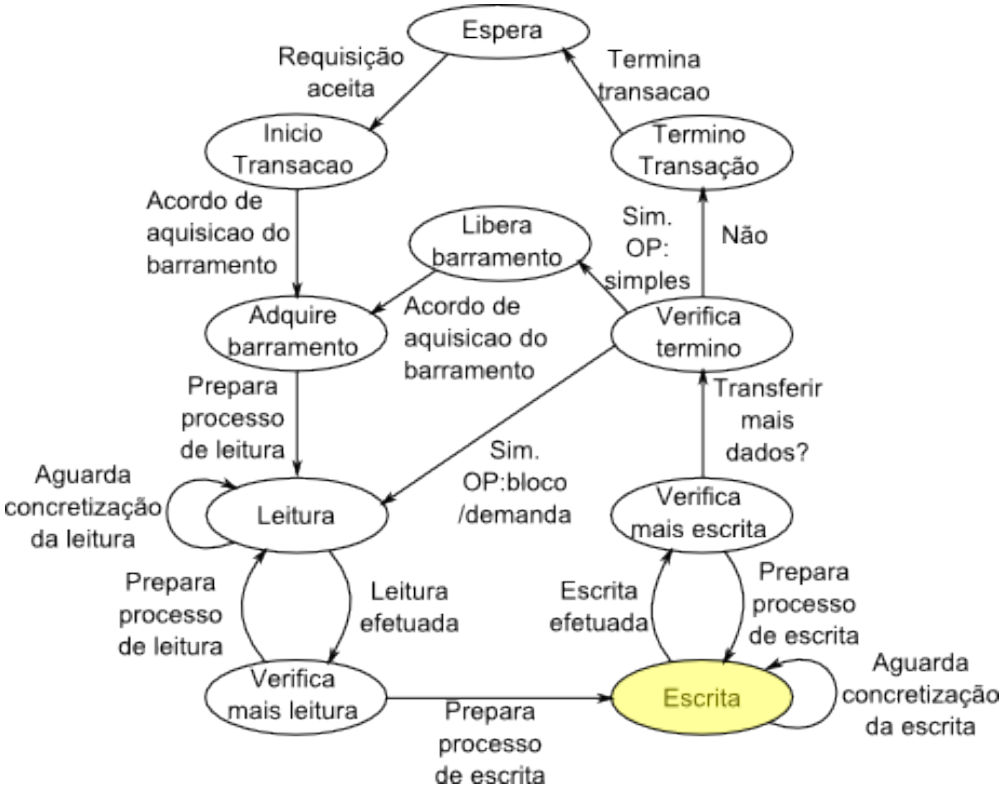


Figura 16: Comportamento Operação: Escrita.

Quando confirmada a escrita, o endereço de escrita é incrementado e o controlador do *buffer* decrementado. Enquanto o *buffer* não estiver vazio, os sinais de escrita serão devidamente habilitados e o DMAC segue para o estado de escrita. Quando o *buffer* estiver vazio, o DMAC passará ao estado que verifica se há mais dados à transmitir. A Figura 17 ilustra o conceito lógico.

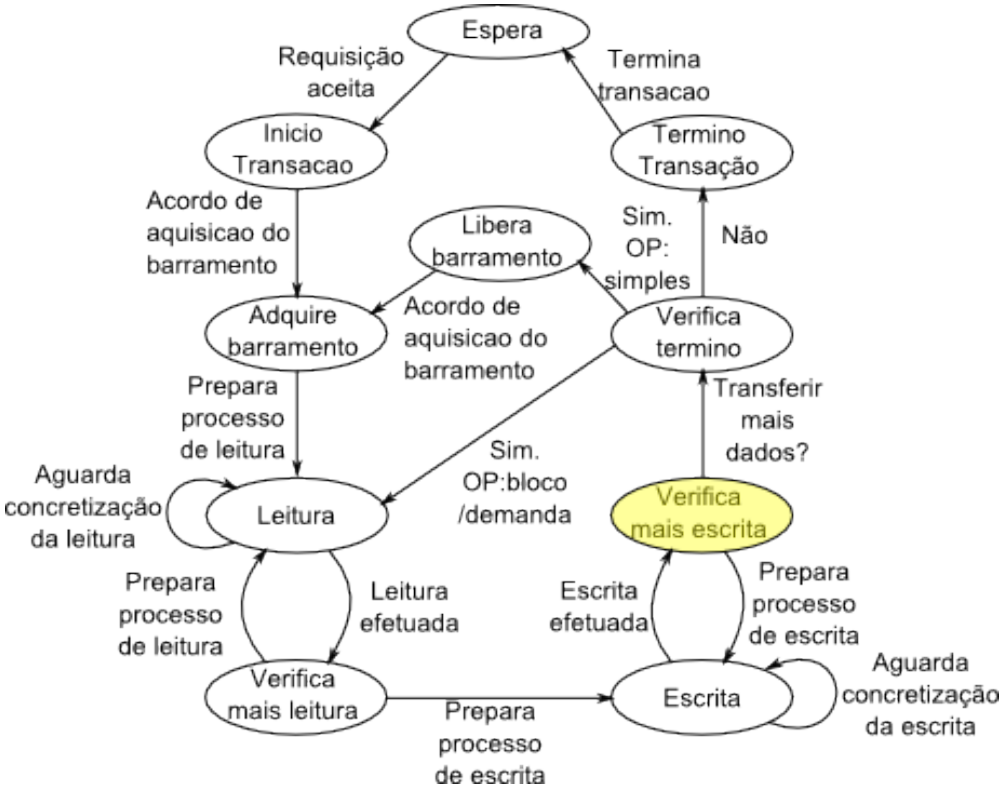


Figura 17: Comportamento Operação: Verifica Mais Escrita.

O estado “verifica término” testa o contador de transações. Se este for zero, o DMAC vai para o estado “término transação”. Porém, se este for diferente de zero, será verificado o modo de operação: caso o modo de operação seja bloco ou demanda, pela definição da operação, os sinais para efetuar mais leituras serão devidamente habilitados e o DMAC seguirá para o estado de leitura; caso seja a operação simples, pela definição da operação, o DMAC seguirá para o estado de liberação do barramento. A Figura 18 ilustra o conceito lógico.

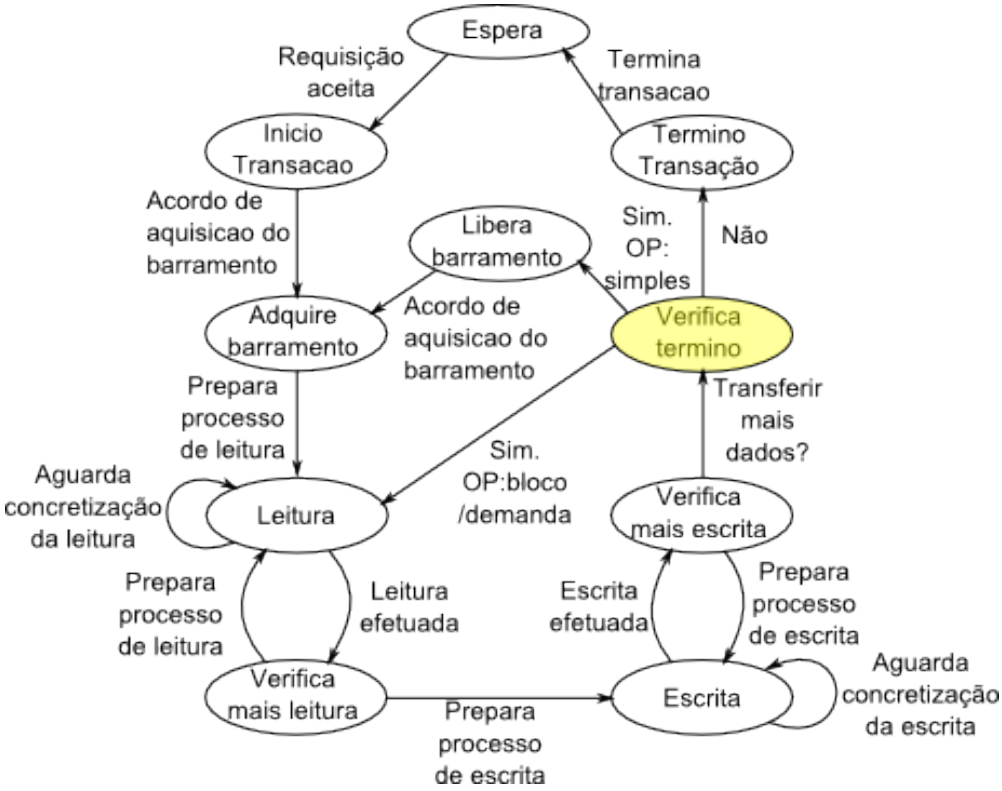


Figura 18: Comportamento Operação: Verifica Término.

Se o tipo de operação for simples e havendo mais dados a transferir, a definição da operação diz que o barramento deve ser liberado e deve ser solicitada a posse do barramento novamente. A liberação é feita desabilitando o sinal HOLD. A Figura 19 ilustra o conceito lógico.

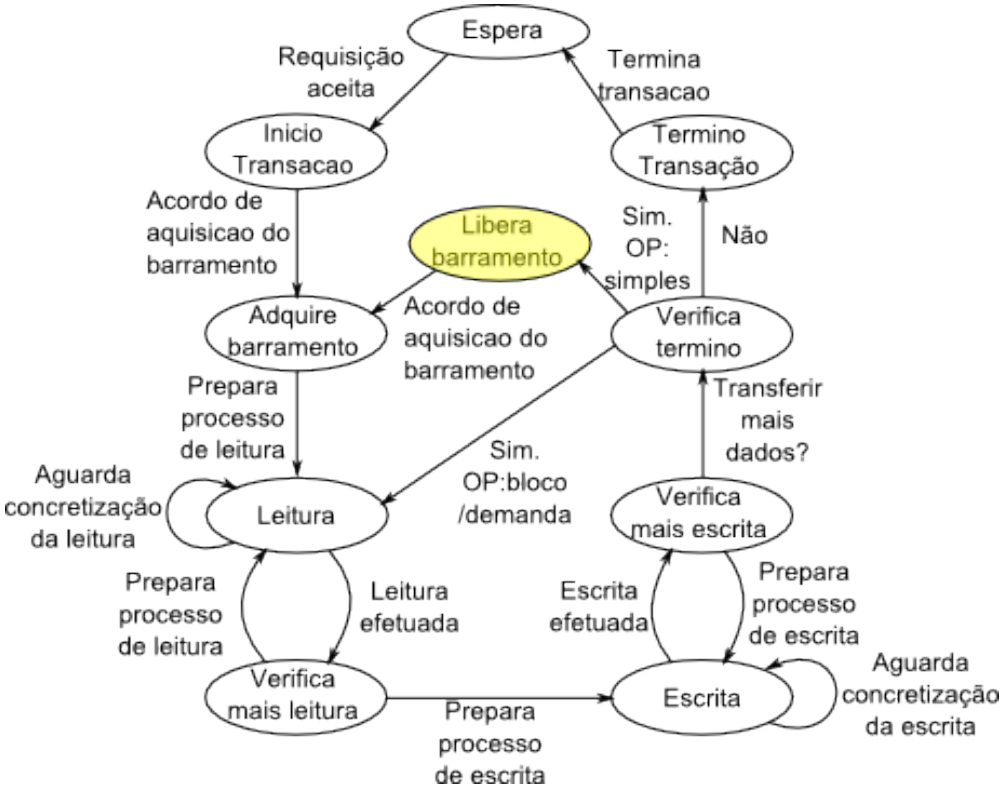


Figura 19: Comportamento Operação: Libera barramento.

Não havendo mais dados para transmitir, o DMAC prepara os sinais de finalização da transação, como desabilitar o sinal DACK[i], desabilitar o sinal HOLD e habilitar o sinal EOP. A Figura 20 ilustra o conceito lógico.

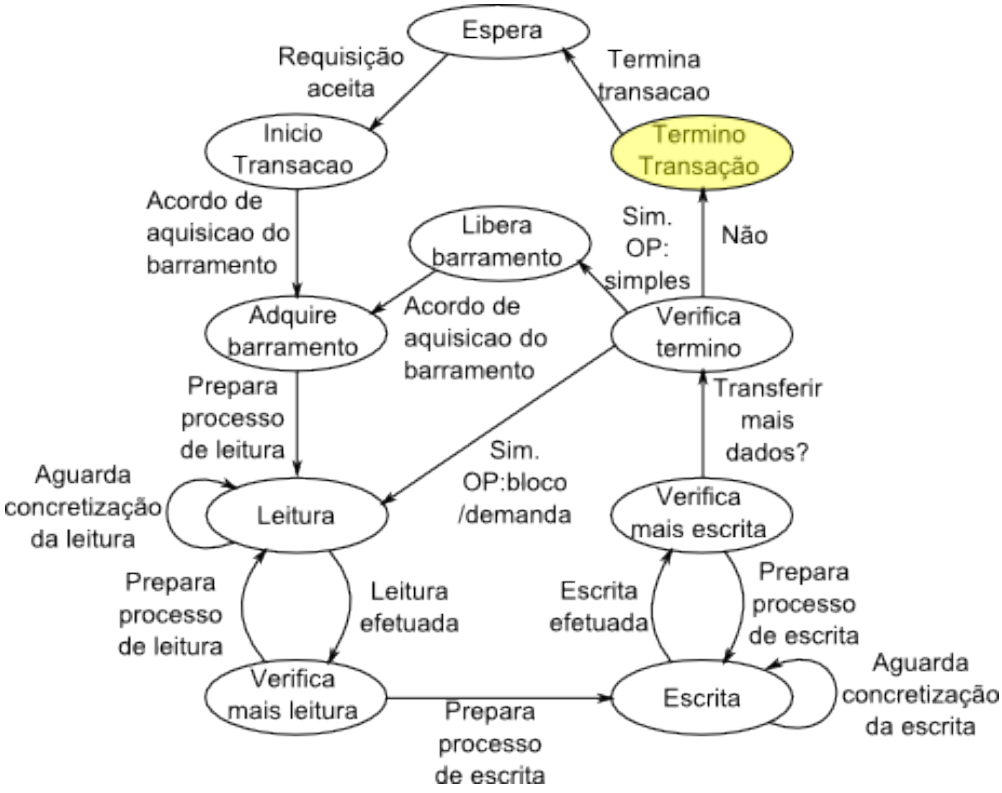


Figura 20: Comportamento Operação: término transação.

3.3 Arquitetura

A arquitetura do DMAC foi concebida tomando-se como referência a arquitetura do DMAC do Projeto Dalton [Vahid 2011], o qual é uma implementação simplificada do Intel 8237A. Logo, foi efetuado um refinamento daquela arquitetura, cujo resultado é detalhado a seguir.

A Figura 21 mostra um diagrama de blocos sintético da arquitetura proposta, considerando um DMAC de quatro canais. As conexões de dados e os sinais estão simplificados para facilitar o entendimento da arquitetura. O sentido das setas indica o sentido dos respectivos sinais, ao passo que os retângulos indicam os blocos de arquitetura que serão descritos a seguir. Os quadrados pequenos representam os registradores internos a cada bloco.

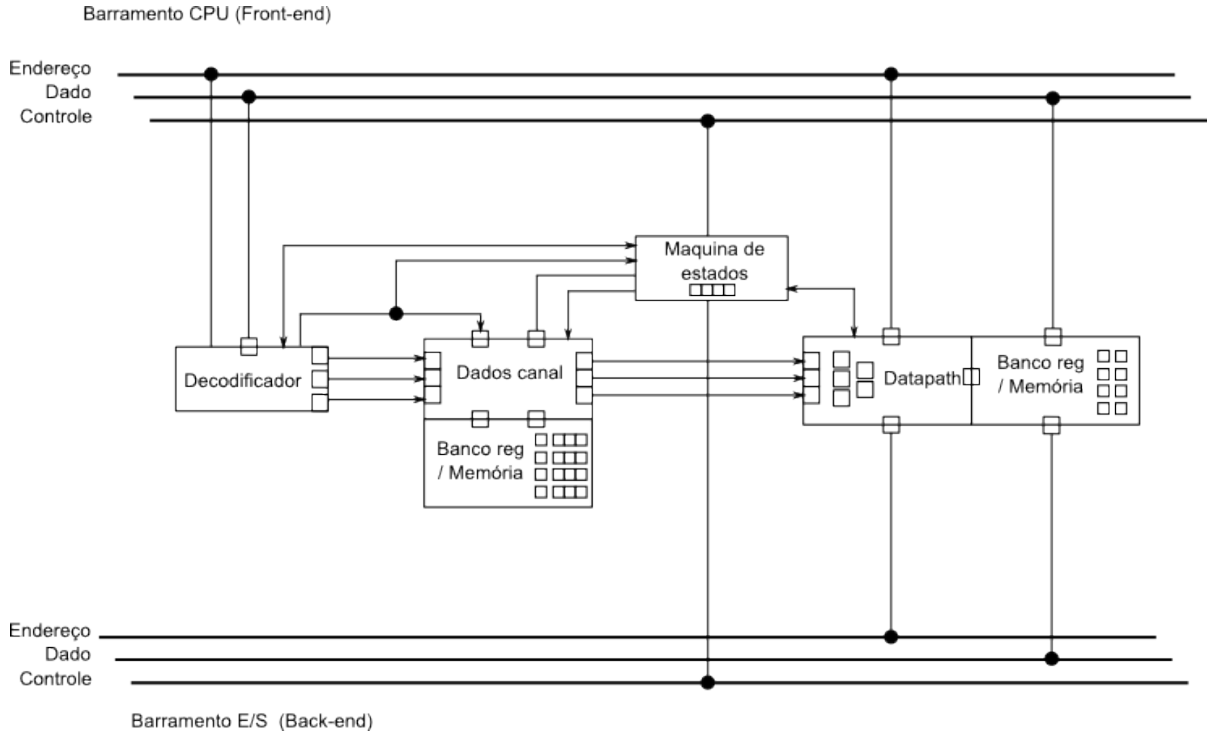


Figura 21: Arquitetura do DMAC.

3.3.1 Decodificador

O decodificador possui acesso de dados e endereço do barramento *front-end*. Ele fica habilitado durante o estado de "ESPERA"e no comportamento CONFIGURAÇÃO. Quando habilitado, fica a espera da escrita no barramento *front-end* de endereço o endereço ao qual o DMAC foi mapeado.

A Figura 22 ilustra a arquitetura proposta do decodificador, sendo: addr_f representa a entrada do barramento de endereço superior, db_f representa a entrada do barramento de dados superior, sl_w_ch_out informa o canal alvo decodificado do endereço, step informa o passo decodificado do endereço, r_status representa o sinal que é desejado ler o *status* do DMAC e out_data é a saída de dados do decodificador, concatenando a rajada de três dados de configuração. Os quadrados internos representam os registradores que armazenam os dados de configuração escritos juntamente com os endereços.

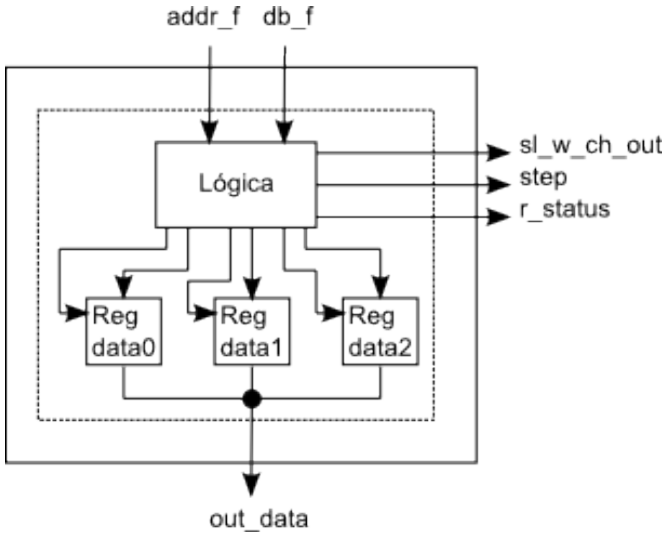


Figura 22: Arquitetura: Decodificador.

3.3.2 Configuração do Canal

Os dados de configuração do canal são armazenados internamente no DMAC para garantir velocidade na carga dos dados, tanto para efetuar configurações, quanto para carregar ao *datapath* ao iniciar uma operação. Sua implementação não possui nenhuma lógica além de uma memória de 2 portas, sendo uma porta de leitura e outra de escrita, sinal de habilitação de escrita, 5 bits de endereçamento e o tamanho da palavra de 96 bits.

A Figura 23 ilustra a arquitetura proposta, sendo: in_data representada pela entrada de dados fornecida pelo decodificador ao qual concatena as três rajadas de dados, sl_w_ch representando o sinal do endereço de escrita do canal, sl_w_ch representando o sinal do endereço de leitura do canal e en_w_data representando a habilitação de escrita no canal.

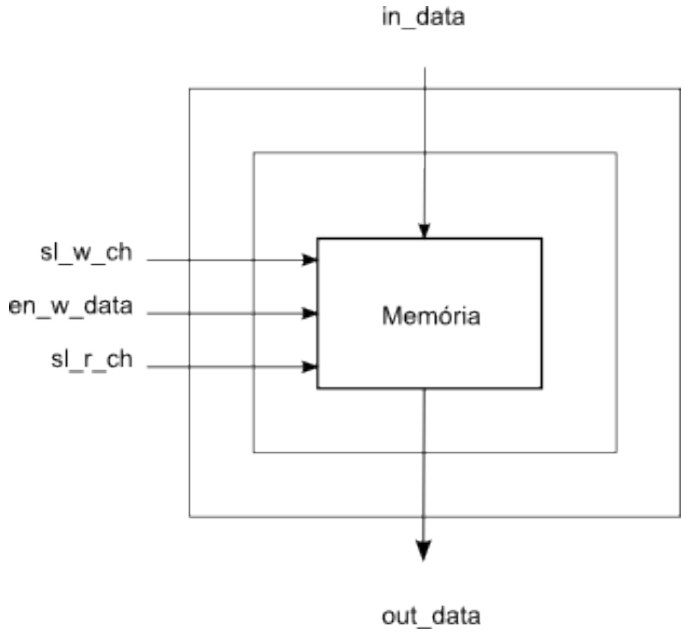


Figura 23: Arquitetura: Configuração do Canal.

3.3.3 *Datapath*

O *datapath* é o principal bloco do DMAC, uma vez que é nele que o prcesso de transferência de dados ocorre de fato. O *datapath* possui acesso de escrita aos barramentos de endereço e acesso de escrita e leitura aos barramentos de dados. Para facilitar o entendimento, o bloco foi dividido em blocos básicos.

O registrador "flags" possui 16 bits e contém todas as configurações específicas de operação relacionada ao canal a fim de utilizá-las quando o referido canal for requisitado em uma transação.

O registrador "addr_f" possui 32 bits e controla o endereço a ser escrito no barramento de dados superior. Seu dado de inicialização provém da configuração dos dados do canal. Para isso, são necessários os devidos sinais da máquina de estados. Posteriormente, o registrador é incrementado/decrementado, em função do sinal de configuração, e a atualização do registrador é feita de forma síncrona, obedecendo os devido sinais. O incremento é fixado em 4 bits pois, na definição de projeto, o registrador é conectado somente a dispositivos de 32 bis. De mesma forma ocorre o decremento.

A Figura 24 expõe a arquitetura planejada, sendo: in_data representando o endereço base do barramento superior provindo dos dados de configuração do canal, sl_w_addr_f sinal provindo da máquina de estados que seleciona a origem dos dados a serem escritos no registrador, en_w_addr_f sinal que habilita a escrita do registrador, en_w_addr_f_f sinal que permite a

escrita do endereço superior no barramento superior, en_w_addr_f_b sinal que permite a escrita do endereço superior no barramento inferior, o sinal de controle indicando o incremento/decremento (flag 7) e o sinal de bloqueio de incremento (flag 9).

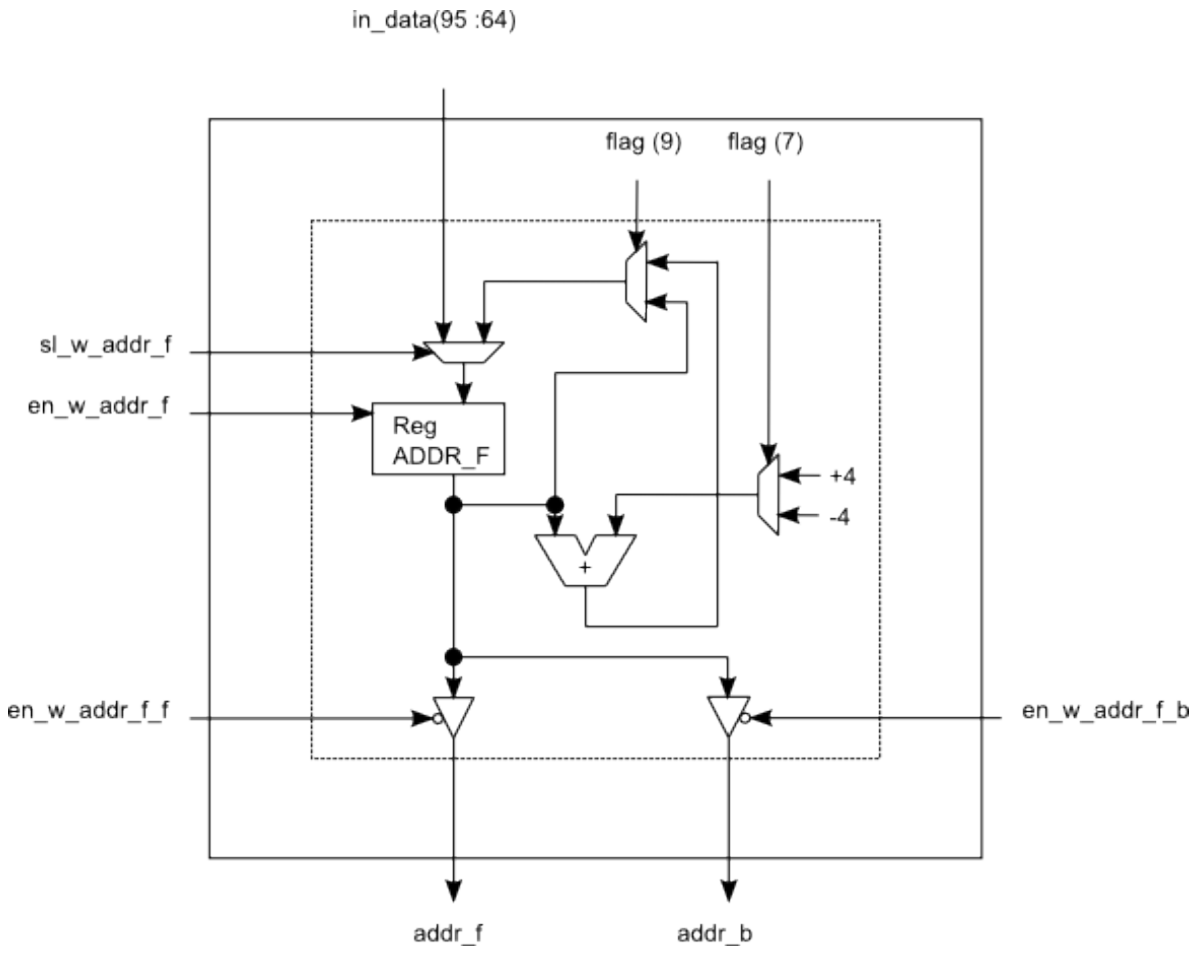


Figura 24: Controle de Incremento/Decremento do Registrador “addr_f”.

O registrador “addr_b” possui 32 bits e controla o endereço escrito no barramento de dados inferior. Seu dado de inicialização provém da configuração dos dados do canal. Para isso são necessários os devidos sinais da máquina de estados. Posteriormente o registrador é incrementado/decrementado, em função do sinal de configuração, e a atualização do registrador é feita de forma síncrona, obedecendo os devidos sinais. O incremento varia de 1 bit quando selecionado o tamanho de palavra de 08 bits, 2 bits quando selecionado o tamanho da palavra de 16 bits, 4 bits quando selecionado o tamanho de palavra de 32 bits e 8 bits quando selecionado o tamanho da palavra de 64 bits, pois na definição de projeto, o registrador pode ser conectado a dispositivos de 08,16, 32 e 64 bits, de mesma forma ocorre o decremento.

A Figura 25 expõe a arquitetura planejada, sendo: in_data representando o endereço base do barramento inferior provindo dos dados de configuração do canal, sl_w_addr_b sinal provindo

da máquina de estados que seleciona a origem dos dados a serem escritos no registrador, en_w_addr_b sinal que habilita a escrita do registrador, en_w_addr_b_b sinal que permite a escrita do endereço inferior no barramento inferior, en_w_addr_b_b sinal que permite a escrita do endereço inferior no barramento superior, sinal de controle indicando o incremento/decremento (flag 6), o sinal de bloqueio de incremento (flag 8) e o sinal de controle que indica o tamanho do incremento do endereço, dado o tamanho da palavra do dispositivo (flag (1:0).

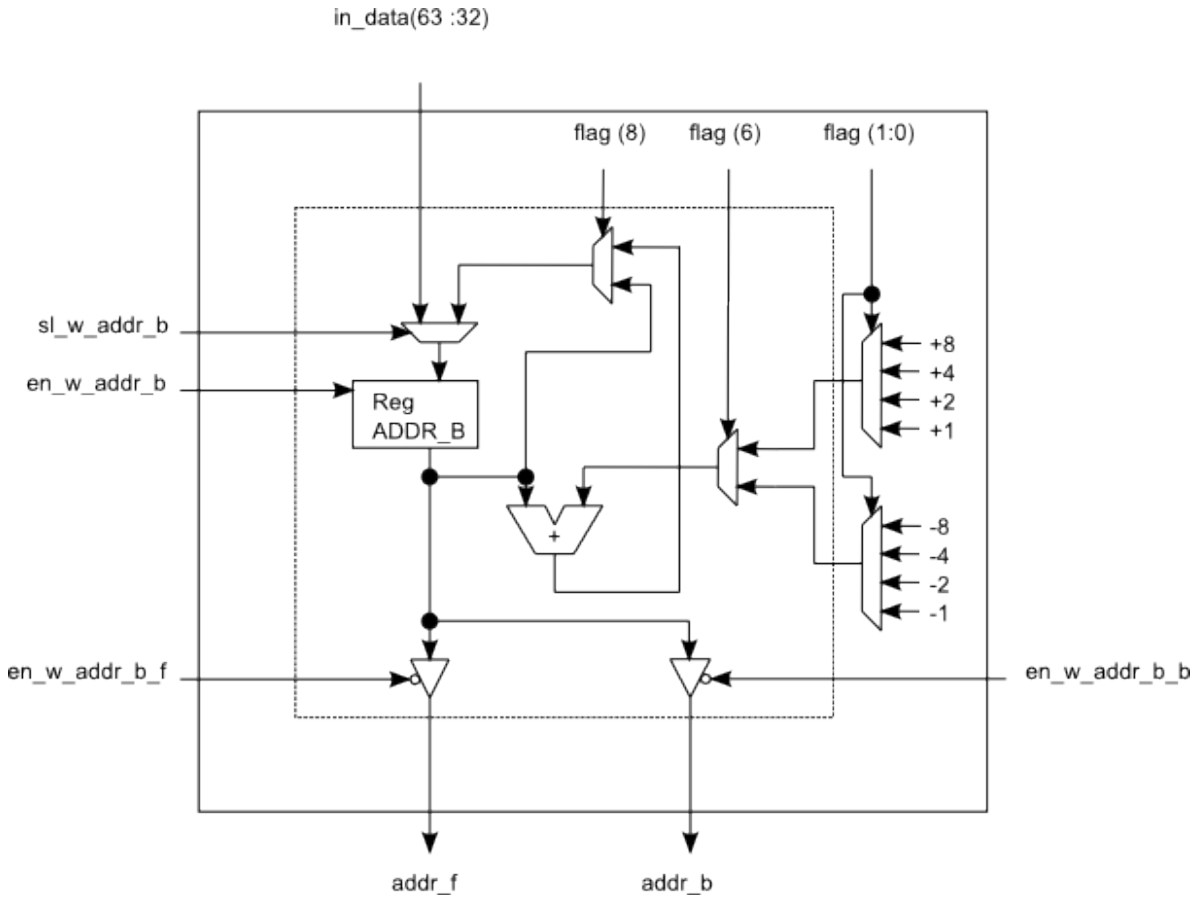


Figura 25: Controle de Incremento/Decremento do Registrador “addr_b”.

O registrador “counter” possui 16 bits e controla a quantidade de transações a serem realizadas. Seu dado de inicialização provém da configuração dos dados do canal. Para isso, são necessários os devidos sinais da máquina de estados. Posteriormente, o contador é decrementado em 1 bit na realização de cada leitura, e a atualização do contador é feita de forma síncrona, obedecendo os devidos sinais. A definição do registrador indica o número absoluto de transações, indiferente ao tamanho da palavra dos dados da transação. Quando o contador chega a 0, é enviado um sinal à máquina de estados informando que a transação do canal já alcançou seu objetivo. O tamanho máximo de transações é de 2^{16}-1.

A Figura 26 expõe a arquitetura planejada, sendo: in_data representando o número de

transações a efetuar, sl_w_counter sinal provindo da máquina de estados que seleciona a origem dos dados a serem escritos no registrador, en_w_counter sinal que habilita a escrita do registrador, counter_out representa a saída do registrador “counter”, counter_is_empty sinal que informa quando o contador chega à zero.

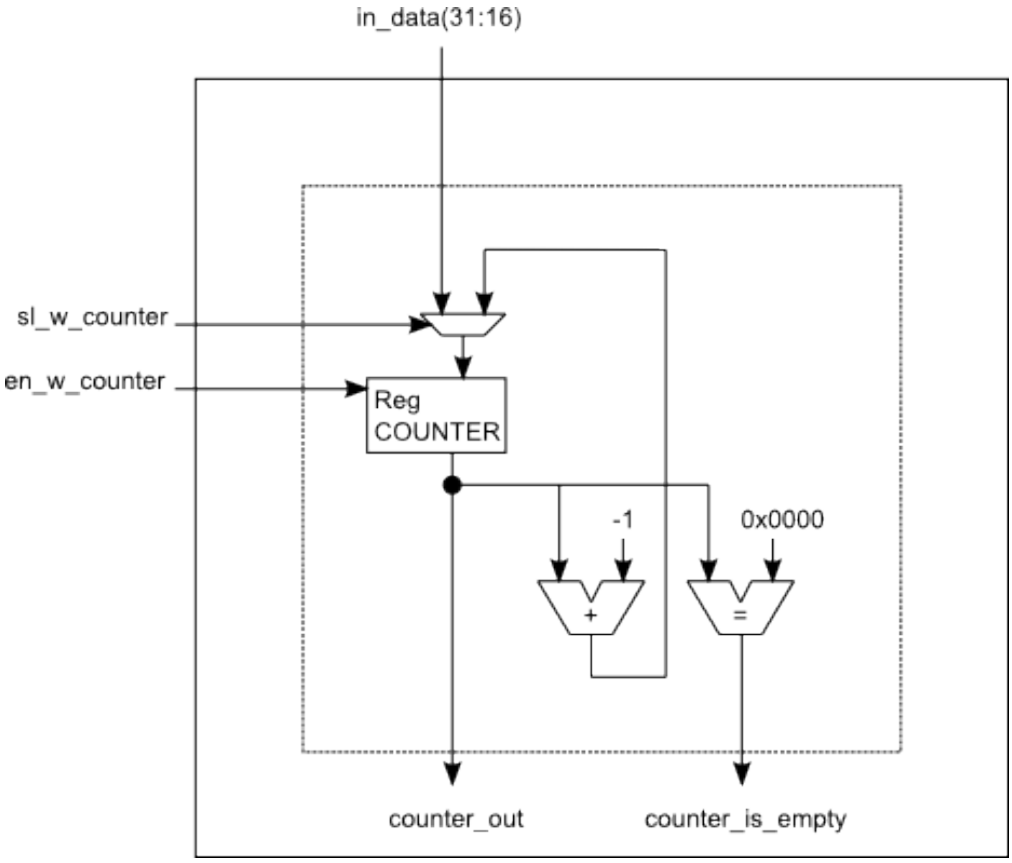


Figura 26: Controle de Decremento do Registrador “counter”.

O registrador “offset” possui 16 bits e controla o endereço de escrita/leitura do *buffer* de dados. Seu valor de inicialização é 1. Para a inicialização são necessários os devidos sinais da máquina de estados. Dependendo das configurações de operação, o registrador não sofre alterações, mas quando a operação solicita a contagem, o registrador é incrementado em 1 bit a cada operação de leitura e decrementado em 1 bit a cada operação de escrita. A atualização do endereço é feita de forma síncrona, obedecendo os sinais relacionados. Quando o tamanho armazenado pelo registrador chega ao endereçamento máximo do *buffer*, no caso 0xFF, é enviado um sinal à máquina de estados informando que o *buffer* está cheio, forçando o DMAC a descarregar o *buffer*. De forma análoga, quando o tamanho for 0, é informado à máquina de estados que o *buffer* está vazio.

A Figura 27 expõe a arquitetura planejada, sendo: sl_w_offset sinal provindo da máquina de estados que seleciona a origem dos dados a serem escritos no registrador, en_w_offset sinal

que habilita a escrita do registrador, sl_op_mode sinal que indica se o DMAC está em modo leitura ou escrita, offset_out representa a saída do registrador “offset”, offset_is_empty sinal que informa quando o registrador “offset” chega a zero e offset_is_full sinal que informa quando o registrador “offset” chega ao limite programado.

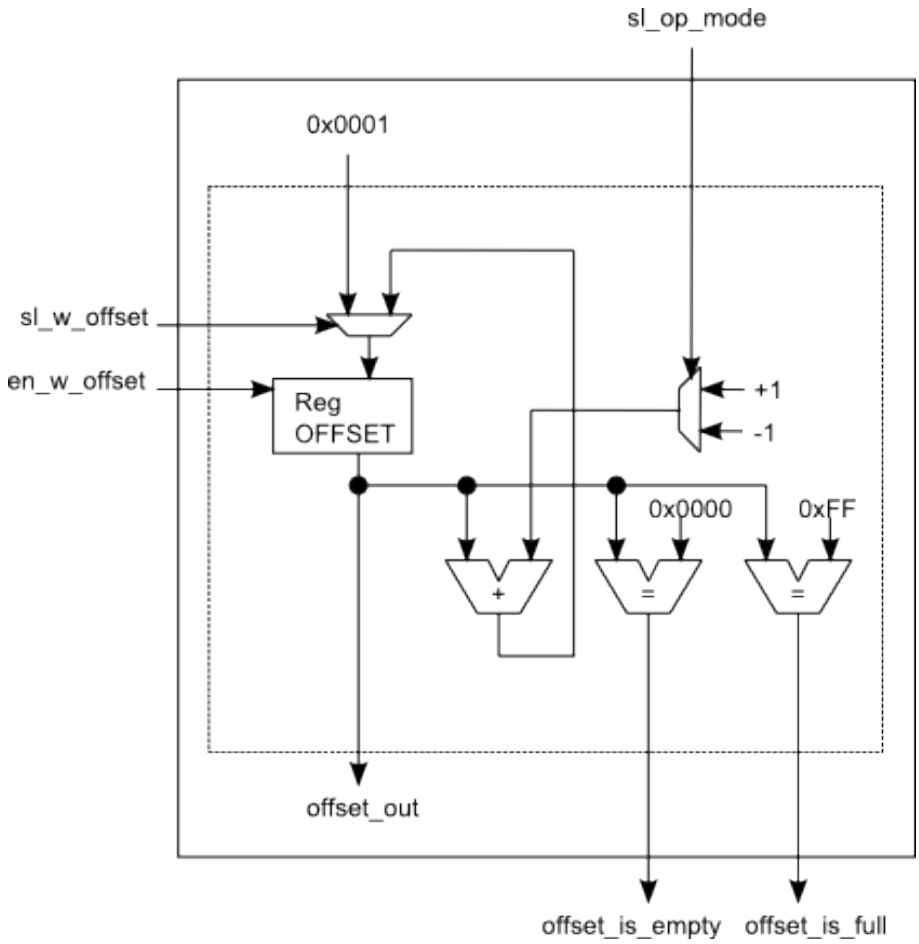


Figura 27: Controle de Incremento/Decremento do Registrador “offset”.

A memória “buffer” possui 8 bits de endereçamento e cada endereço armazena 32 bits. Sua função é de armazenar os dados de um dos barramentos de dados e escrever em um dos barramentos de dados. O controle do endereço de leitura é fornecido pelos 8 bits menos significativos do registrador offset. Os sinais habilitação de escrita, seleção do barramento aser lido e seleção do barramento a ser escrito provém da máquina de estados.

A Figura 28 expõe a arquitetura planejada, sendo: db_b representa o barramento de dados inferior, db_f representa o barramento de dados superior, en_r_db_f sinal que seleciona o dado do barramento superior como entrada de dados do “buffer”, en_r_db_b sinal que seleciona o dado do barramento inferior como entrada de dados do “buffer”, offset_out representa o endereço da memória, en_w_db_f sinal que habilita a escrita do conteúdo do “buffer” no barra-

mento superior, en_w_db_b sinal que habilita a escrita do conteúdo do “buffer” no barramento inferior.

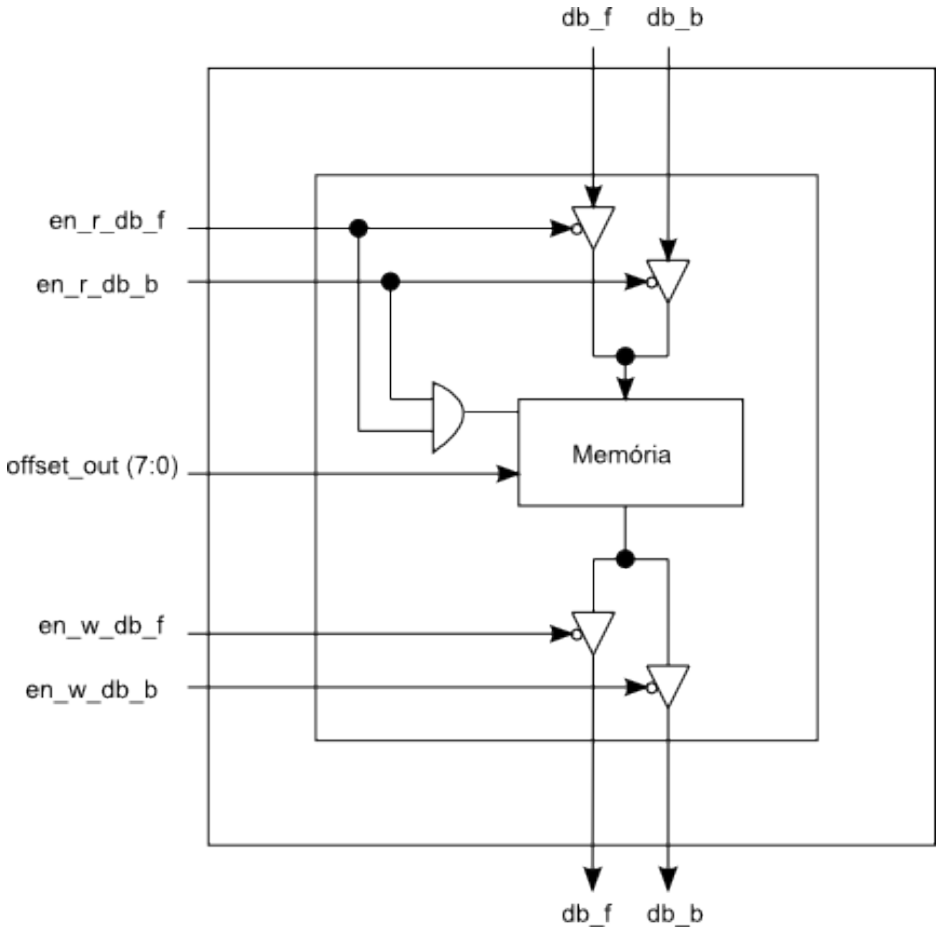


Figura 28: Controle da área de memória “buffer”.

3.3.4 Bloco de Controle

Vistas as especificações de comportamento (de alto nível) e da arquitetura do DMAC proposto, o bloco de controle será detalhado, a partir do comportamento da máquina de estados implementada. A máquina de estados contém 20 estados, sendo 3 relacionados ao comportamento CONFIGURAÇÃO, 14 relacionados ao comportamento OPERAÇÃO, 1 relacionado à inicialização, 1 relacionado à recuperação de erros e 1 dedicado a reportar o *status* do DMAC.

O estado relacionado à inicialização e espera é “ESPERA”, o qual é alcançado de forma assíncrona quando o sinal “RESET” está habilitado, ou de forma síncrona a partir dos estados: “PASSO_2”, ao término da etapa de configuração; “FINALIZACAO”, ao término da transferência; “STATUS”, ao cessar a requisição de leitura de status; ”ERRO” após registrado o erro. Os estados alcançados por “ESPERA” são: “PASSO_0”, através da escrita no barramento superior do endereço no qual o DMAC está mapeado; “TRANSACAO_SOLICITADA”, através

do levante do sinal DREQ [i] (sendo i o canal associado à requisição) e, o canal associado já haver sido configurado; “STATUS” ao ser escrito o endereço no barramento superior de endereço ao qual requisita a leitura de *status*. No estado “ESPERA” os sinais de controle estão desabilitados aguardando algum evento para transacionar o estado. A Figura 29 ilustra as transições de alcance deste estado.

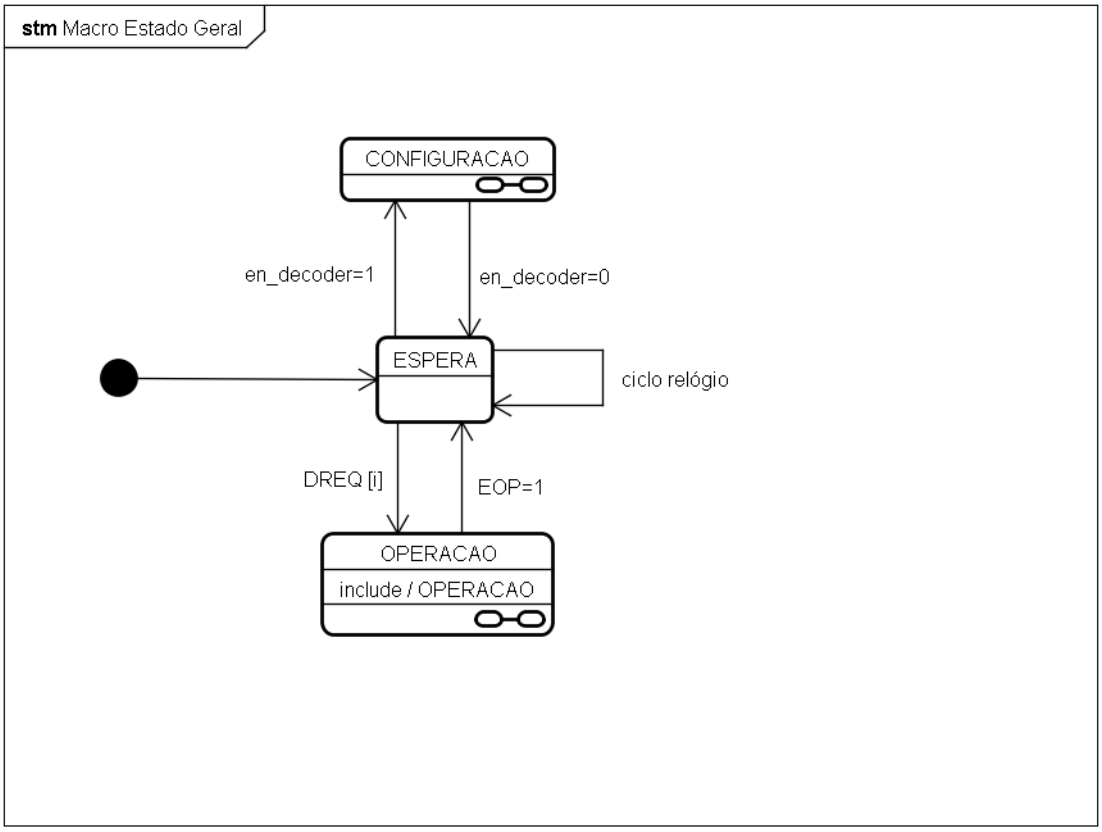


Figura 29: Bloco de Controle Inicial.

O estado relacionado ao *status* é “STATUS”, o qual é alcançado a partir do estado “ESPERA” através da escrita no barramento superior de endereços do endereço para o qual o DMAC está mapeado, com os dois bits menos significativos habilitados. O estado alcançado por “STATUS” é “ESPERA”, ao ser escrito um endereço diferente do endereço de ativação de leitura de status. No estado são habilitados sinais de escrita no barramento superior de dados escrevendo o conteúdo do registrador de status. A Figura 30 ilustra as transições de alcance deste estado.

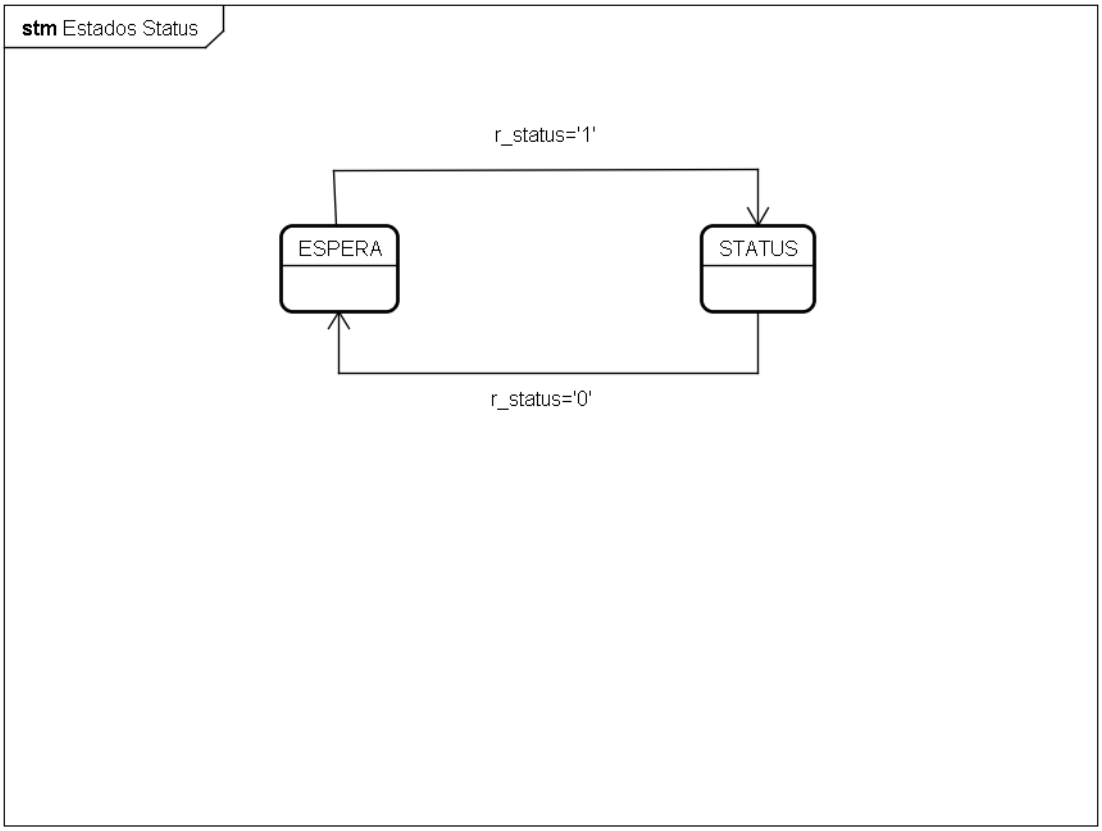


Figura 30: Bloco de Controle de *Status*.

O estado relacionado ao erro é “ERRO”, o qual pode ser alcançado por qualquer dos demais estados através da violação de alguma verificação de estado corrente. A partir de “ERRO” é possível alcançar o estado “ESPERA” após um ciclo de relógio.

Os estados relacionados ao comportamento CONFIGURAÇÃO são: “PASSO_0”, “PASSO_1”, “PASSO_2”.

O estado “PASSO_0” é alcançado a partir do estado de “ESPERA” através da escrita no barramento superior do endereço no qual o DMAC está mapeado e do estado “PASSO_2”, quando são realizadas configurações consecutivas de canal, através da escrita no barramento de endereço superior do endereço para o qual o DMAC está mapeado. A partir de “PASSO_0” é possível alcançar o estado “PASSO_1”, através da variação dos dois bits menos significativos do endereço. Neste estado são ativados sinais que habilitam a leitura e o armazenamento temporário do dado lido do barramento superior de dados e a seleção do canal alvo. A Figura 31 ilustra as transições de alcance deste estado.

O estado “PASSO_1” é alcançado a partir do estado “PASSO_0”, através da variação dos dois bits menos significativos do endereço. A partir de “PASSO_1” é possível alcançar o estado

“PASSO_2” a partir da variação dos dois bits menos significativos do endereço. No estado são ativados sinais que habilitam a leitura e o armazenamento temporário do dado lido no barramento superior de dados. A Figura 31 ilustra as transições de alcance deste estado. A Figura 31 ilustra as transições de alcance deste estado.

O estado “PASSO_2” é alcançado a partir do estado “PASSO_1”, através da variação dos dois bits menos significativos do endereço. Os estados alcançados a partir do estado “PASSO_2” são: O estado “PASSO_2”, através da variação dos dois bits menos significativos do endereço, ao qual informa a etapa da configuração; O estado “ESPERA” através da escrita de um endereço no barramento superior de endereço ao qual não seja o mapeado pelo DMAC. No estado são ativados sinais que habilitam a leitura e o armazenamento temporário do dado lido no barramento superior de dados, habilitação de escrita dos dados de configuração do canal na área de configuração do canal e marcação do canal alvo como configurado. A Figura 31 ilustra as transições de alcance deste estado.

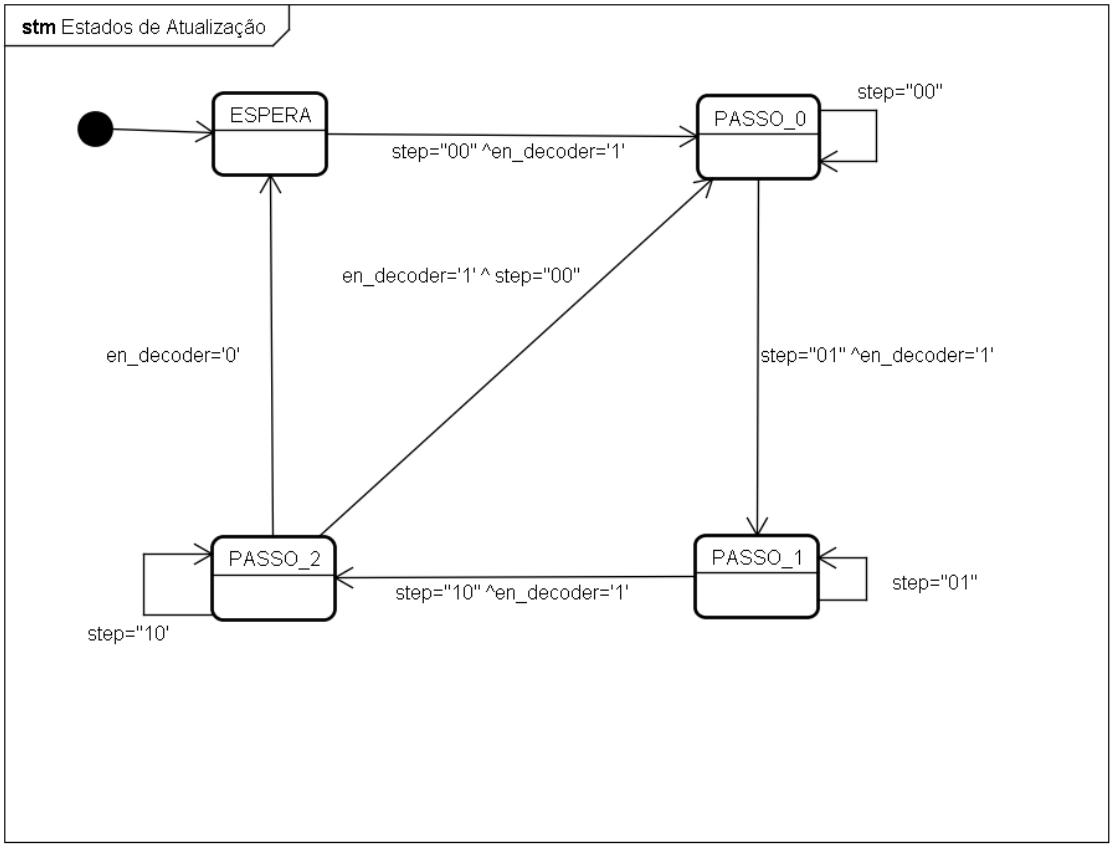


Figura 31: Bloco de Controle Configuração.

Os estados relacionados ao comportamento OPERAÇÃO são: “TRANSACAO_SOLICITADA”, “BARRAMENTO_SOLICITADO”, “BARRAMENTO_GARANTIDO”, “TRANSACAO_CONFIRMADA”, “OP_LEITURA_A”, “OP_LEITURA_B”, “OP_CICLO_LEITURA”,

“VERIFICA_MAIS_LEITURA”, “OP_ESCRITA_A”, “OP_ESCRITA_B”, “OP_CICLO_ESCRITA”, “FAZER_MAIS_ESCRITA”, “TESTA_CONTADOR” e “FINALIZACAO”. Os estados listados acima estão contidos em um dos macro estados listados na Figura 32, sendo detalhados ao serem listados.

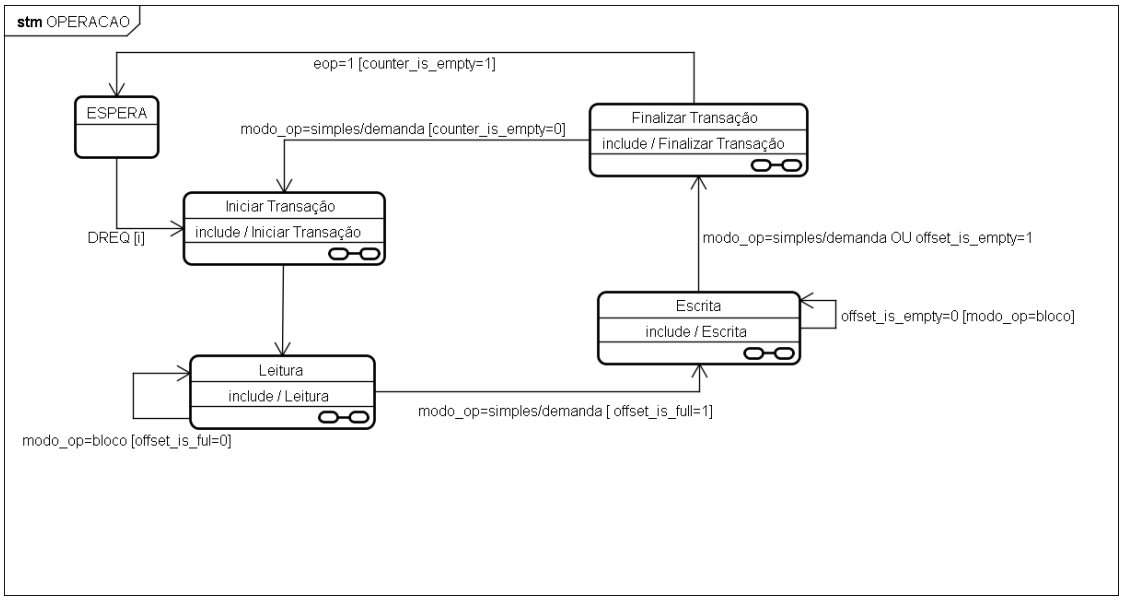


Figura 32: Bloco de Controle: Operação.

O estado “TRANSACAO_SOLICITADA” é alcançado a partir do estado “ESPERA”, através do levante do sinal DREQ [i] (sendo i o canal associado à requisição) e, o canal associado já haver sido configurado. O estado alcançado a partir do estado “TRANSACAO_SOLICITADA” é o estado “BARRAMENTO_SOLICITADO”, através da habilitação do sinal “HOLD” ao qual é responsável de requisitar ao processador a utilização do barramento superior de endereço e dados. No estado são ativados sinais que efetuam a cópia dos dados da configuração do canal ao *datapath* e levante do sinal “HOLD”. A Figura 33 ilustra as transições de alcance deste estado.

O estado “BARRAMENTO_SOLICITADO” é alcançado a partir do estado “TRANSACAO_SOLICITADA”, através da habilitação do sinal “HOLD”. O estado alcançado a partir do estado “BARRAMENTO_SOLICITADO” é o estado “BARRAMENTO_GARANTIDO”, através da constatação da habilitação do sinal “HLDA” ao qual é responsável de informar o aceite da requisição da utilização do barramento superior de endereço e dados. No estado são desativados os sinais que efetuam a cópia dos dados da configuração do canal ao *datapath*. A Figura 33 ilustra as transições de alcance deste estado.

O estado “BARRAMENTO_GARANTIDO” é alcançado a partir do estado “BARRAMENTO_SOLICITADO”, através da constatação da habilitação do sinal “HLDA”. O estado

alcançado a partir do estado “BARRAMENTO_GARANTIDO” é o estado “TRANSACAO-_CONFIRMADA”, através da habilitação do sinal “DACK [i]” (sendo i o canal associado à requisição), ao qual é responsável de informar o aceite da requisição ao dispositivo solicitante. No estado não há alteração de sinais além do levante do sinal “DACK [i]”. A Figura 33 ilustra as transições de alcance deste estado.

O estado “TRANSACAO_CONFIRMADA” é alcançado a partir do estado “BARRAMENTO_GARANTIDO”, através da habilitação do sinal “DACK[i]”. Os estados alcançados a partir do estado “TRANSACAO_CONFIRMADA” são: O estado “OP_LEITURA_A” no próximo ciclo de relógio, dado que a configuração do canal indique uma operação do tipo memória – memória ou memória – dispositivo, ou seja, uma leitura no barramento superior; O estado “OP_LEITURA_B” no próximo ciclo de relógio, dado que a configuração do canal indique uma operação do tipo dispositivo – memória ou dispositivo – dispositivo, ou seja, uma leitura no barramento inferior. No estado são preparados sinais para efetuar uma leitura, por exemplo, uma leitura na memória, a habilitação do sinal de escrita no barramento superior de endereço (en_w_addr_f_f), leitura da memória (mem_r), a leitura do barramento superior de dados (en_r_db_f), entre outros. A Figura 33 ilustra as transições de alcance deste estado.

O estado “OP_LEITURA_A” é alcançado a partir do estado “BARRAMENTO_GARANTIDO” passado um ciclo de relógio, ou “VERIFICA_MAIS_LEITURA” caso a operação seja do tipo bloco ou demanda e existam mais dados a serem lidos, ou do estado “TESTA_CONTADOR” caso existam mais dados a serem transmitidos, e dado que a configuração do canal indique uma operação do tipo memória – memória ou memória – dispositivo. O estado alcançado a partir do estado “OP_LEITURA_A” é o estado “OP_CICLO_LEITURA”. Caso a operação seja do tipo simples, a transição de estado ocorre em um ciclo de relógio, caso seja do tipo bloco ou demanda, é aguardada a habilitação do sinal “READY”, cuja função é sincronizar o DMAC com dispositivos mais lentos. No estado são mantidos os sinais de leitura ativados no estado anterior. A Figura 33 e Figura 34 ilustram as transições de alcance deste estado.

O estado “OP_LEITURA_B” é alcançado a partir do estado “BARRAMENTO_ GARANTIDO” passado um ciclo de relógio, ou “VERIFICA_MAIS_LEITURA” caso a operação seja do tipo bloco ou demanda e existam mais dados a serem lidos, ou do estado “TESTA_CONTADOR” caso existam mais dados a serem transmitidos, e dado que a configuração do canal indique uma operação do tipo dispositivo – memória ou dispositivo – dispositivo. O estado alcançado a partir do estado “OP_LEITURA_B” é o estado “OP_CICLO_LEITURA”. Caso a operação seja do tipo simples, a transição de estado ocorre em um ciclo de relógio, caso seja do tipo bloco ou demanda, é aguardada a habilitação do sinal “READY”, cuja função é sincronizar o DMAC com dispositivos mais lentos. No estado são mantidos os sinais de leitura ativados no

estado anterior. A Figura 33 e Figura 34 ilustram as transições de alcance deste estado.

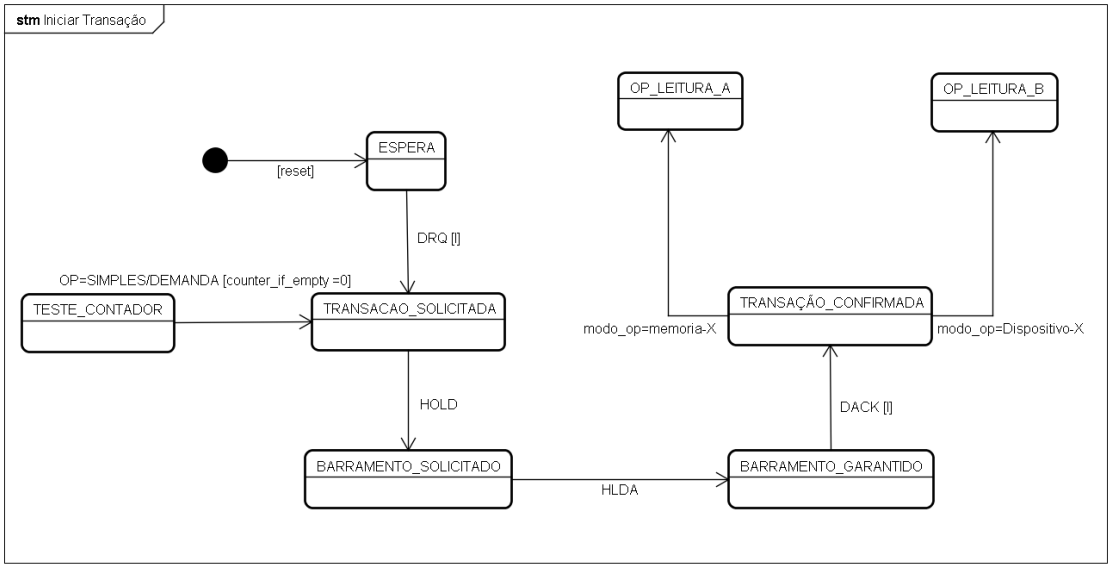


Figura 33: Bloco de Controle: Inicio Transação.

O estado “OP_CICLO_LEITURA” é alcançado a partir do estados “OP_LEITURA_A” ou “OP_LEITURA_B”, passando um ciclo de relógio ou habilitação do sinal “READY”. O estado alcançado a partir do estado “OP_CICLO_LEITURA” é o estado “VERIFICA_MAIS-_LEITURA”, através de um ciclo de relógio. No estado são desabilitados os sinais de leitura e ativados os sinais de controle interno do DMAC para calculo do próximo endereço de leitura, incrementar o *offset*, decrementar o numero de palavras a transmitir, entre outros. A Figura 34 ilustra as transições de alcance deste estado.

O estado “VERIFICA_MAIS_LEITURA” é alcançado a partir do estado “OP_CICLO_-LEITURA”, passando um ciclo de relógio. Os estados alcançados a partir do estado “VERIFICA_MAIS_LEITURA” são: O estado “OP_LEITURA_A”, através de um ciclo de relógio, caso a operação seja do tipo bloco ou demanda e existam mais dados a transmitir e o *buffer* não estiver cheio e a transferência for do tipo memória – memória ou memória – dispositivo; O estado “OP_LEITURA_B”, através de um ciclo de relógio, caso a operação seja do tipo bloco ou demanda e existam mais dados a transmitir e o *buffer* não estiver cheio e a transferência for do tipo dispositivo – memória ou dispositivo – dispositivo; O estado “OP_ESCRITA_A”, através de um ciclo de relógio, caso a operação seja do tipo simples ou o *buffer* estiver cheio e a transferência for do tipo memória – memória ou dispositivo – memória; O estado “OP_ES-CRITA_B”, através de um ciclo de relógio, caso a operação seja do tipo simples ou o *buffer* estiver cheio e a transferência for do tipo memória – dispositivo ou dispositivo – dispositivo. No estado são preparados os sinais para efetuar uma nova leitura, caso a transição de próximo

estado seja do tipo leitura ou preparar os sinais de escrita, caso a transação de próximo estado seja de escrita. A Figura 34 ilustra as transições de alcance deste estado.

O estado “OP_ESCRITA_A” é alcançado a partir do estado “VERIFICA_MAIS_LEITURA”, passando um ciclo de relógio e a operação seja do tipo dispositivo – memória ou memória – memória. O estado alcançado a partir do estado “OP_ESCRITA_A” é o estado “OP_CICLO_ESCRITA”, através de um ciclo de relógio caso a transferência seja do tipo simples ou no caso de transferências do tipo bloco ou demanda, aguardar a habilitação do sinal “READY” informando que o dispositivo está pronto. No estado são mantidos os sinais de escrita habilitados pelo estado anterior. A Figura 34 e Figura 35 ilustram as transições de alcance deste estado.

O estado “OP_ESCRITA_B” é alcançado a partir do estado “VERIFICA_MAIS_LEITURA”, passando um ciclo de relógio e a operação seja do tipo dispositivo – dispositivo ou memória – dispositivo. O estado alcançado a partir do estado “OP_ESCRITA_B” é o estado “OP_CICLO_ESCRITA”, através de um ciclo de relógio caso a transferência seja do tipo simples ou no caso de transferências do tipo bloco ou demanda, aguardar a habilitação do sinal “READY” informando que o dispositivo está pronto. No estado são mantidos os sinais de escrita habilitados pelo estado anterior. A Figura 34 e Figura 35 ilustram as transições de alcance deste estado.

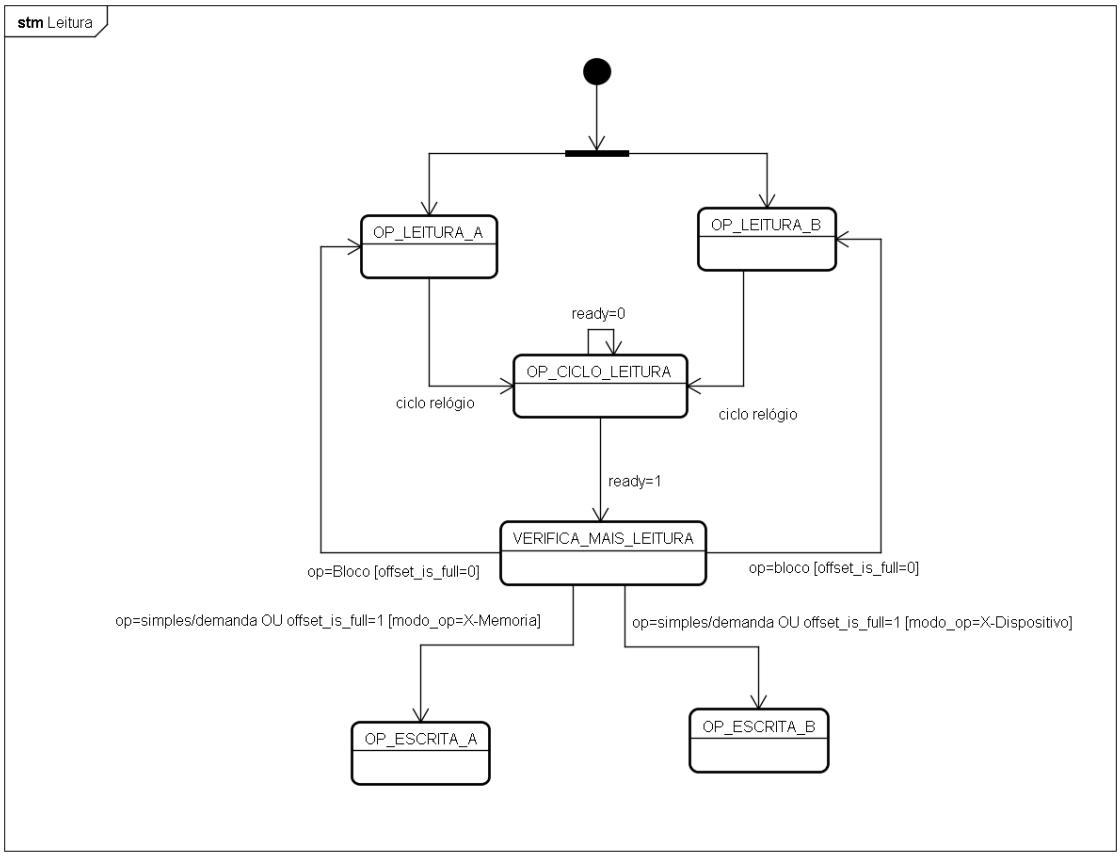


Figura 34: Bloco de Controle: Leitura.

O estado “OP_CICLO_ESCRITA” é alcançado a partir dos estados “OP_ESCRITA_A” ou “OP_ESCRITA_B”, passando um ciclo de relógio ou habilitação do sinal “READY”. O estado alcançado a partir do estado “OP_CICLO_ESCRITA” é o estado “VERIFICA_MAIS_ ESCRITA”, através de um ciclo de relógio. No estado são desabilitados os sinais de escrita e ativados os sinais de controle interno do DMAC para calculo do próximo endereço de escrita como decrementar o *offset*, entre outros. A Figura 35 ilustra as transições de alcance deste estado.

O estado “VERIFICA_MAIS_ESCRITA” é alcançado a partir do estado “OP_CICLO_ESCRITA”, passando um ciclo de relógio. Os estados alcançados a partir do estado “VERIFICA-_MAIS_ESCRITA” são: O estado “OP_ESCRITA_A”, através de um ciclo de relógio, caso a operação seja do tipo bloco ou demanda, o *buffer* não estiver vazio e a transferência for do tipo memória – memória ou dispositivo – memória; O estado “OP_ESCRITA_B”, através de um ciclo de relógio, caso a operação seja do tipo bloco ou demanda, o *buffer* não estiver vazio e a transferência for do tipo dispositivo – dispositivo ou dispositivo – memória; O estado “TESTA_CONTADOR”, através de um ciclo de relógio, caso a operação seja do tipo simples ou o *buffer* estiver vazio. No estado são preparados os sinais para efetuar uma nova escrita, caso a

transação de próximo estado seja de escrita ou desabilitar os sinais de escrita caso a transação de próximo estado seja para o teste do contador. A Figura 35 ilustra as transições de alcance deste estado.

O estado “TESTA_CONTADOR” é alcançado a partir do estado “VERIFICA_MAIS_ESCRITA”, passando um ciclo de relógio. O estado alcançado a partir do estado “TESTA_CONTADOR” são: O estado “OP_LEITURA_A”, através de um ciclo de relógio, caso o contador seja diferente de zero e a transferência seja do tipo memória – memória ou memória – dispositivo; O estado “OP_LEITURA_B”, através de um ciclo de relógio, caso o contador seja diferente de zero e a transferência seja do tipo dispositivo – memória ou dispositivo – dispositivo; O estado “FINALIZACAO” caso o contador seja zero. No estado são habilitados os sinais de leitura caso a transição de estado vá para um estado de leitura ou habilita o sinal de término do processo de transferência “EOP” juntamente com a desabilitação do sinal “DACK [i]” caso a transição vá para o estado de término. A Figura 35 e a Figura 36 ilustram as transições de alcance deste estado.

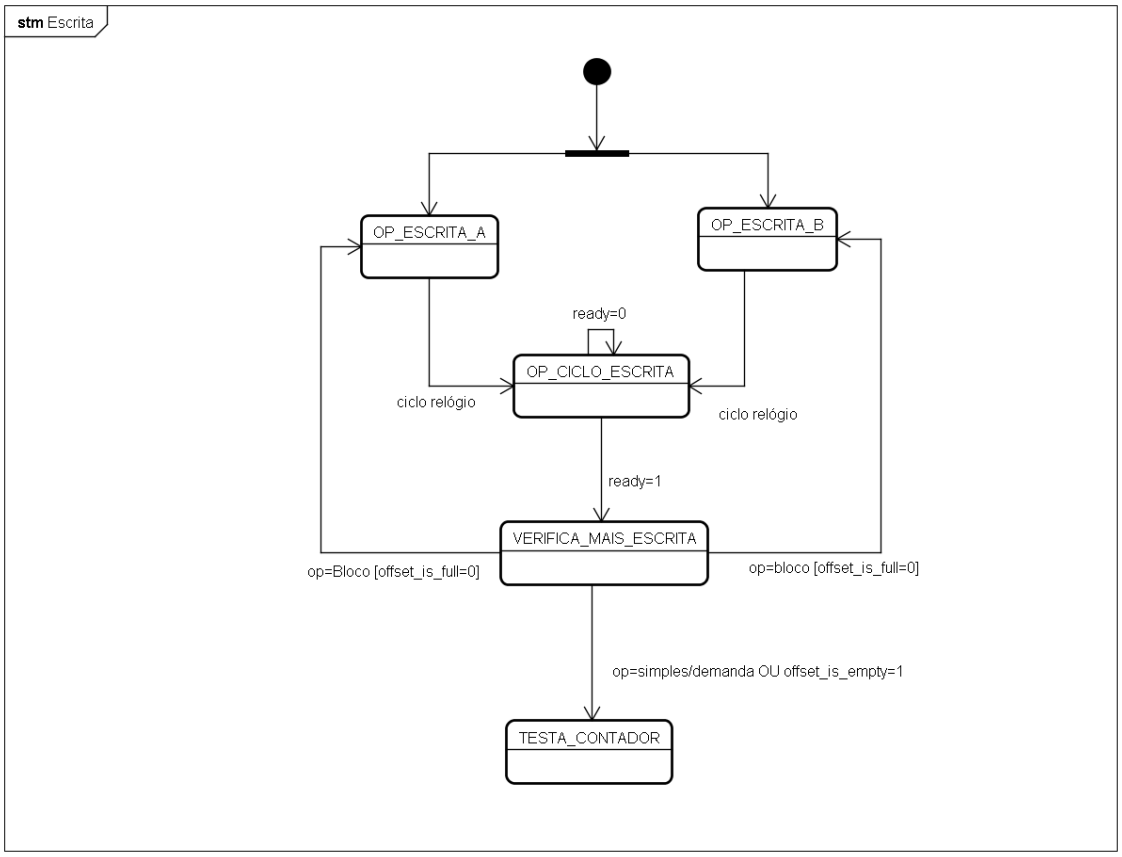


Figura 35: Bloco de Controle: Escrita.

O estado “FINALIZACAO” é alcançado a partir do estado “TESTA_CONTADOR”, passando um ciclo de relógio. O estado alcançado a partir do estado “FINALIZACAO” é o estado

“ESPERA”, através da detecção da desabilitação do sinal “HLDA”. No estado são mantidos os sinais até a detecção da desabilitação do sinal “HLDA”. A Figura 36 ilustra as transições de alcance deste estado.

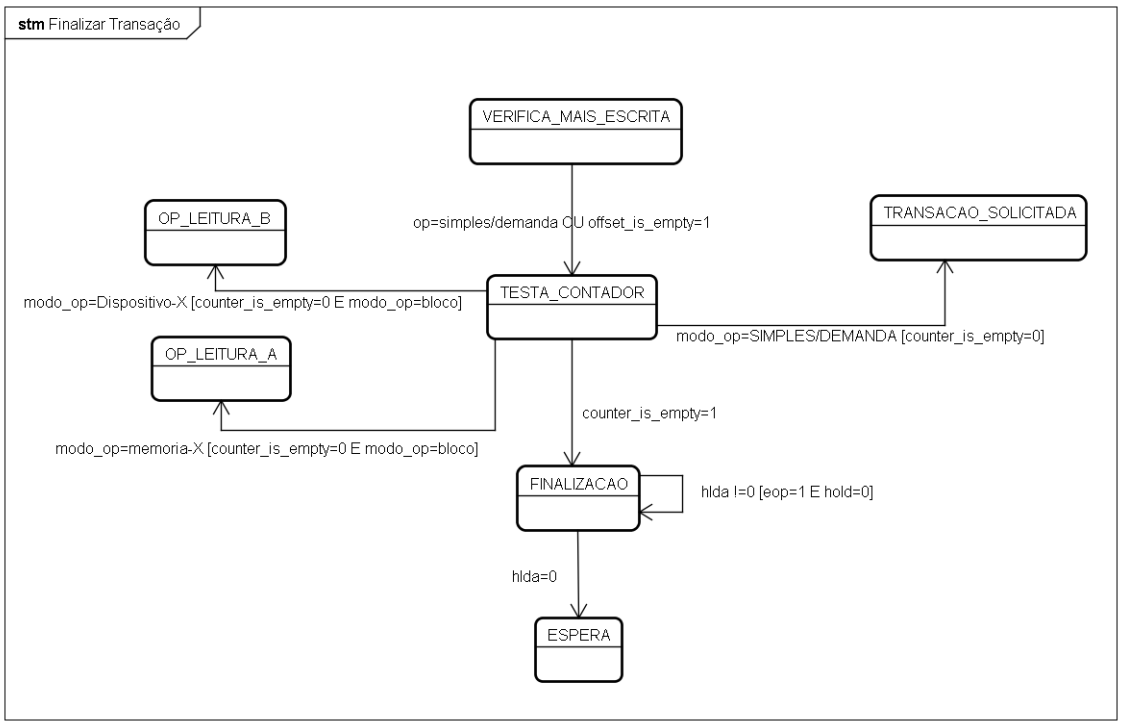


Figura 36: Bloco de Controle: Finalização da Transação.

3.4 Geração Automática de Descrições RTL de DMAC

Conforme foi ressaltado nas seções anteriores, o projeto do DMAC foi realizado de maneira modular, o que favorece a alteração de suas características, como por exemplo, o número de canais. Assim, a fim de facilitar as descrições RTL em VHDL de outras versões do DMAC deste trabalho, alterando o número de canais, foi desenvolvido um gerador de código VHDL. Este gerador foi feito em linguagem PHP que utiliza código desenvolvido manualmente para os blocos básicos do DMAC, instanciado-os em número adequado, conforme os parâmetros fornecidos pelo usuário. Tais parâmetros são:

- número de canais;
- mapeamento da área de buffer (memória do FPGA ou banco de registradores, gerado automaticamente);
- endereço para o qual o DMSC está mapeado. Existe ainda uma opção que pode ser

habilitada ou não, a qual prevê a geração automática do projeto para a ferramenta Quartus II , da Altera.

Informado os dados, todos os arquivos são gerados e disponibilizado através de um arquivo compactado, contendo todos os arquivos gerados. A mudança na interface do DMAC é dado pela letra "N" na Figura 37, os demais sinais permanecem inalterados.

Este gerador foi utilizado para criar a descrições do DMAC que foram sintetizadas no contexto deste trabalho.

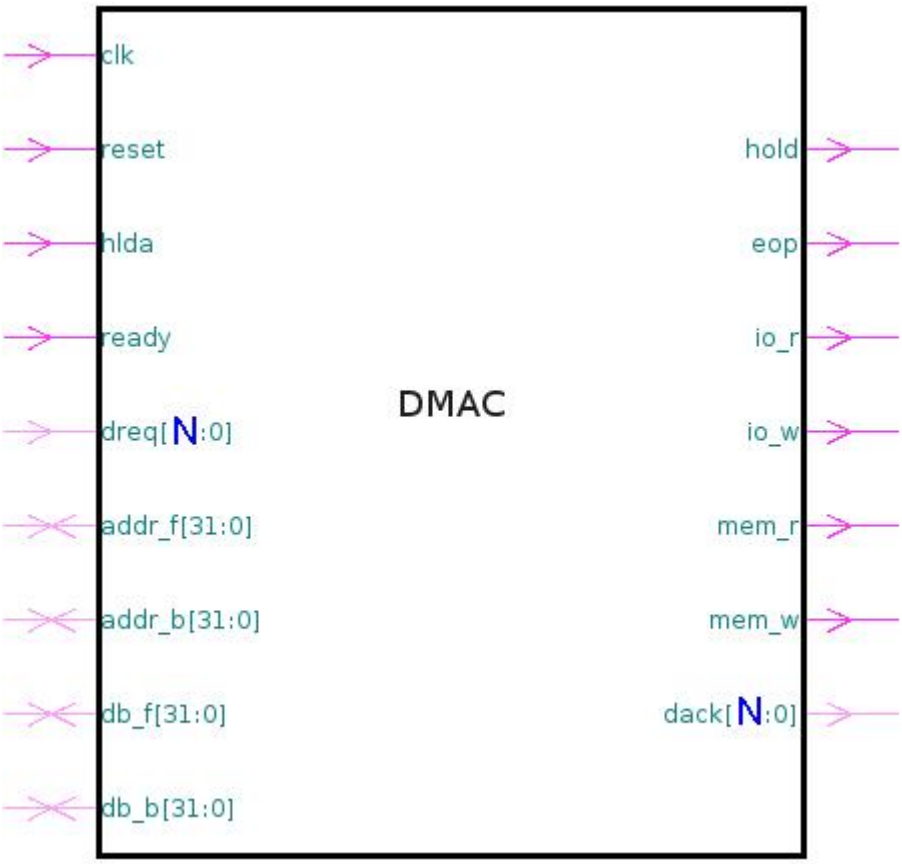


Figura 37: Interface do DMAC.

4 *Descrição, Síntese e Validação do DMAC*

Este capítulo apresenta os resultados da síntese de duas versões de DMAC, uma com 4 canais e outra com 32 canais, para ASICs e para FPGA da Altera. No caso da síntese para FPGAs, cada uma destas duas versões foi sintetizada de duas formas: uma fazendo uso de blocos SRAM do FPGA para implementar o *buffer* de dados e outra usando banco de registradores. Os resultados da síntese são apresentados e comentados.

4.1 Versões de DMAC Sintetizadas

Para permitir a síntese das duas versões de DMAC para ASIC e para FPGA foram criadas 4 descrições de DMAC:

- **DMAC A:** Versão voltada para síntese para FPGA. Possui 4 canais e o *buffer* de dados é implementado como um bloco de memória SRAM do FPGA.
- DMAC B: Versão voltada para síntese para FPGA. Possui 32 canais e o *buffer* de dados é implementado como um bloco de memória SRAM do FPGA.
- DMAC C: Versão que pode ser usada tanto para síntese para ASIC quanto para FPGA. Possui 4 canais e o *buffer* de dados é implementado como um banco de registradores.
- DMAC D: Versão que pode ser usada tanto para síntese para ASIC quanto para FPGA. Possui 32 canais e o *buffer* de dados é implementado como um banco de registradores.

As descrições A e C foram geradas com parâmetros similares ao DMAC 8237A da Intel (por exemplo, número de canais), de modo a permitir uma análise do impacto no uso de recursos, no desempenho e na potência em decorrência da maneira como o *buffer* de dados é implementado. As descrições B e D tem o intuito de permitir uma análise do impacto do número de canais no uso de recursos, no desempenho e na potência.

4.2 Síntese

As descrições do DMAC foram sintetizadas para FPGA e ASIC. A síntese para FPGA utilizou a ferramenta Quartus II Web Edition da Altera, assumindo o componente EP2C35F672C6 da família Cyclone II da Altera. A escolha deste componente deveu-se à disponibilidade de placas de prototipação DE2, as quais utilizam tal componente (embora não tenha sido possível realizar a validação do DMAC proposto em placa). A síntese para ASIC utilizou a ferramenta Synopsys Design Compiler em modo Topographical, utilizando uma biblioteca Standard-Cells de 45nm da TSMC.

A potência para síntese em FPGA foi gerada através de atividade de chaveamento com os seguintes parâmetros: 12,5% do tempo com probabilidade de 50% de 0->1 e 50% de 1->0 e para síntese em ASIC foi gerada através de atividade de chaveamento com parâmetros padrões da ferramenta 12,5% do tempo com 50% de 0->1 e 50% de 1->0.

A Tabela 1 mostra os resultados obtidos pela síntese das descrições em FPGA. O impacto do tipo de implementação do *buffer* é facilmente percebido, como pode ser visto comparando as versões A e C, cujo único diferencial é o mapeamento da área de buffer. A utilização de recursos lógicos passa de 1431 para 13392, um aumento de aproximadamente 835,85%. Ao comparar a potência dissipada, o aumento é considerável, passando de 160,75 mW para 494,74 mW, um acréscimo de aproximadamente 207,77%. A frequência de funcionamento também é afetada, passando de 84,76 MHz para 59,06 MHz, uma redução de aproximadamente 30,32%. Os resultados da comparação das versões B e D seguem a mesma tendência da comparação entre as versões A e C. Comparando as versões A e B, cujo único diferencial é a definição do número de canais, a utilização de unidades lógicas aumenta de 1431 para 4596, um aumento de aproximadamente 221,17% enquanto o número de canais passa de 4 para 32, um acréscimo de 700%. A dissipação de potência passa de 160,75 mW para 198,04 mW, um acréscimo de aproximadamente 23,19%. A frequência de funcionamento passa de 84,76 MHz para 72,71 MHz, uma redução de 14,21%

Tabela 1: Síntese das Descrições em FPGA.

	DMAC A	DMAC B	DMAC C	DMAC D
Área (LUTs[1])	1431 (4,30%)	4596 (13,83%)	13392 (40,32%)	16697 (50,27%)
Potência (mW)	160,75	198,04	494,74	520,37
Frequência (MHz)	84,76	72,71	59,06	56,19

A Tabela 2 mostra os resultados obtidos da síntese das versões C e D para ASIC. As versões A e B não foram sintetizadas uma vez que elas pressupõem o uso de recursos específicos do FPGA. As comparações que seguem tomam a versão C como referência. A área aumenta aproximadamente 29,25% enquanto o número de canais aumenta 7 vezes. A potência total aumenta aproximadamente em 37,30%. A frequência cai em aproximadamente 29,55%.

Tabela 2: Síntese das descriçõesm em ASIC.

	DMAC C	DMAC D
Área (μm^2)	60697	78453
Potência dinâmica (mW)	2,06	2,87
Potência estática (mW)	0,54	0,70
Frequência (MHz)	322,58	227,27

4.3 Validação

Idealmente, a etapa de verificação visa provar que o projeto é isento de erros. Entretanto, em geral é difícil, senão impossível, garantir que o projeto esteja isento de erros. Assim, a verificação busca encontrar erros, a partir da simulação dos casos mais usuais de funcionamento. Tal procedimento é similar à estatística hipotética [Martin e Bailey Brian 2007].

A validação do bloco DMAC foi efetuada através de eventos dirigidos, no objetivo de efetuar a cobertura da máquina de estados. As comparações foram feitas utilizando o modelo assert, sendo um dado escrito na entrada e, a saída comparada com este dado. Para auxiliar no processo de validação, foi utilizado a ferramenta ModelSim da Mentor (versão Altera).

O processo de validação seguiu as seguintes etapas: O Canal é devidamente configurado, após é solicitada a leitura do status do DMAC, logo são gerados sinais para iniciar a transação sendo devidamente monitorados. Quando efetuado a requisição à leitura, os dados são escritos no devido barramento e ao efetuar a escrita, os dados são comparados com os dados lidos. No final da transação são gerados os sinais de término, sendo devidamente monitorados.

Durante a fase de validação, foram encontrados alguns erros na emissão de sinais por parte da máquina de estados, sendo tais erros corrigidos e seu funcionamento normalizado. A cobertura atingida pelo conjunto de entradas fornecidos foi de cobrir todos os estados da máquina de estados.

5 *Conclusão e Trabalhos Futuros*

5.1 Conclusão

Através das descrições geradas foi possível estimar o impacto da implementação do buffer de dados no uso de recursos, desempenho e potência do DMAC. Também foi possível estimar o impacto do número de canais sobre estes mesmos parâmetros.

Entre as descrições sintetizadas em FPGA, relativos à área: Nas descrições de 4 canais, observou-se o decréscimo de aproximadamente 89,31% da descrição A em relação à descrição C. Nas descrições de 32 canais, observou-se o decréscimo de aproximadamente 72,47% da descrição B em relação à descrição D. Nas descrições ao qual a área de *buffer* foi mapeada para SRAM da FPGA, observou-se o decréscimo de aproximadamente 68,86% da descrição A em relação à descrição B. Nas descrições ao qual a área de *buffer* foi mapeada para banco de registradores, observou-se o decréscimo de aproximadamente 19,79% da descrição C em relação à descrição D. Entre as descrições sintetizadas em ASIC, relativos à frequência: Nas descrições ao qual a área de *buffer* foi mapeada para banco de registradores, observou-se o decréscimo de aproximadamente 22,63% da descrição C em relação à descrição D. Não foi possível efetuar comparações entre as versões C e D de ASIC relativo à FPGA devido a não possuirem a mesma métrica de medida. Tais resultados são observados na Tabela 3.

A Tabela 3 expõe os comparativos relativos à área, as regiões em que não possuem dados preenchidos não possuem relações que permitam efetuar comparações.

Tabela 3: Comparativo de área.

Referência		FPGA			ASIC
		DMAC A	DMAC B	DMAC C	DMAC C
FPGA	DMAC B	-68,86%	-	-	-
	DMAC C	-89,31%	-	-	-
	DMAC D	-	-72,47%	-19,79%	-
ASIC	DMAC D	-	-	-	-22,63%

Entre as descrições sintetizadas em FPGA, relativos à potência: Nas descrições de 4 canais, observou-se o decréscimo de aproximadamente 67,51% da descrição A em relação à descrição C. Nas descrições de 32 canais, observou-se o decréscimo de aproximadamente 61,94% da descrição B em relação à descrição D. Nas descrições ao qual a área de *buffer* foi mapeada para SRAM da FPGA, observou-se o decréscimo de aproximadamente 18,83% da descrição A em relação à descrição B. Nas descrições ao qual a área de *buffer* foi mapeada para banco de registradores, observou-se o decréscimo de aproximadamente 4,93% da descrição C em relação à descrição D. Entre as descrições sintetizadas em ASIC, relativos à frequência: Nas descrições ao qual a área de *buffer* foi mapeada para banco de registradores, observou-se o decréscimo de aproximadamente 27,17% da descrição C em relação à descrição D. Comparando os resultados entre as duas sínteses: Para descrição C observou-se o decréscimo de aproximadamente 99,47% do resultado da síntese ASIC em relação à síntese FPGA. Para descrição D observou-se o decréscimo de aproximadamente 99,31% do resultado da síntese ASIC em relação à síntese FPGA. Tais resultados são observados na Tabela 4.

A Tabela 4 expõe os comparativos relativos à potência, as regiões em que não possuem dados preenchidos não possuem relações que permitam efetuar comparações.

Tabela 4: Comparativo de potência.

Referência		FPGA			ASIC	
		DMAC A	DMAC B	DMAC C	DMAC C	DMAC D
FPGA	DMAC B	-18,83%	-	-	-	-
	DMAC C	-67,51%	-	-	-99,47%	-
	DMAC D	-	-61,94%	-4,93%	-	-99,31%
ASIC	DMAC D	-	-	-	-27,17%	-

Entre as descrições sintetizadas em FPGA, relativos à frequência: Nas descrições de 4 canais, observou-se o acréscimo de aproximadamente 43,52% da descrição A em relação à descrição C. Nas descrições de 32 canais, observou-se o acréscimo de aproximadamente 29,40% da descrição B em relação à descrição D. Nas descrições ao qual a área de *buffer* foi mapeada para SRAM da FPGA, observou-se o acréscimo de aproximadamente 16,57% da descrição A em relação à descrição B. Nas descrições ao qual a área de *buffer* foi mapeada para banco de registradores, observou-se o acréscimo de aproximadamente 5,11% da descrição C em relação à descrição D. Entre as descrições sintetizadas em ASIC, relativos à frequência: Nas descrições ao qual a área de *buffer* foi mapeada para banco de registradores, observou-se o acréscimo de aproximadamente 41,94% da descrição C em relação à descrição D. Comparando os resultados entre as duas sínteses: Para descrição C observou-se o acréscimo de aproximadamente 446,19% do resultado da síntese ASIC em relação à síntese FPGA. Para descrição D observou-se o acréscimo de aproximadamente 304,47% do resultado da síntese ASIC em relação à síntese FPGA. Tais resultados são observados na Tabela 5.

A Tabela 5 expõe os comparativos relativos à frequência, as regiões em que não possuem dados preenchidos não possuem relações que permitam efetuar comparações.

Tabela 5: Comparativo de frequência.

Referência		FPGA			ASIC	
		DMAC A	DMAC B	DMAC C	DMAC C	DMAC D
FPGA	DMAC B	16,57%	-	-	-	-
	DMAC C	43,52%	-	-	446,19%	-
	DMAC D	-	29,40%	5,11%	-	304,47%
ASIC	DMAC D	-	-	-	41,94%	-

5.2 Trabalhos Futuros

Como possibilidade de trabalhos futuros, a arquitetura do DMAC poderia ser modificada para permitir ao barramento superior trabalhar com tamanhos de palavras diferentes de 32 bits, visto a necessidade de operar com unidades de dados diferentes de 32 bits e/ou unidades de processamento diferentes de 32 bits. Tal alteração arquitetural proporcionaria maior portabilidade ao DMAC. Outra possíbilidade é a prototipação do DMAC na placa de FPGA, com 2 ou mais blocos de memória, alguns blocos aceleradores e/ou blocos de interfaceamento (como *Universal Asynchronous Receiver/Transmitter* (UART), *Universal Serial Bus* (USB)) e o DMAC

projetado.

Referências

BALL, S. *Introduction to direct memory access*. 2011. [Online; accessed 16-Maio-2011]. Disponível em: <http://www.eetimes.com/discussion/beginner-s-corner/4024879/Introduction-to-direct-memory-access/>.

CROWELL, e. a. J. Ieee standard for communicating among processors and peripherals using shared memory (dma). *IEEE Std 1212.1-1993*, 1994.

DANDAMUDI, S. *Fundamentals of computer organization and design*. Springer, 2003. (Texts in computer science). ISBN 9780387952116. Disponível em: <http://books.google.com/books?id=qug840wejVQC>.

HEATH, S. *Embedded Systems Design*. 2nd. ed. Newton, MA, USA: Butterworth-Heinemann, 2002. ISBN 0750655461.

HENNESSY, J. L.; PATTERSON, D. A. *Arquitetura de Computadores: Uma Abordagem Quantitativa*. 3nd. ed. [S.l.]: Editora Campus, 2003. ISBN 8535211101.

INTEL. *8237/8237-2 HIGH PERFORMANCE PROGRAMMABLE DMA CONTROLLER (data sheet)*. 2011. [Online; accessed 16-Maio-2011]. Disponível em: <http://zet.aluzina.org/images/8/8c/Intel-8237-dma.pdf>.

MANO, R. *Métodos de Execução de Entradas e Saídas - DMA*. 2011. [Online; accessed 16-Maio-2011]. Disponível em: <http://wwwusers.rdc.puc-rio.br/rmano/intedma.html>.

MARTIN, G.; BAILEY BRIAN, P. A. *ESL Design and Verification: A Prescription for Electronic System Level Methodology*. USA: Morgan Kaufmann, 2007. 488p.

NEWMAN, M.; SMITH, D. The mc6809 in dma mode on the ieee-488 bus. *Micro, IEEE*, v. 1, n. 4, p. 56 –75, nov. 1981. ISSN 0272-1732.

NULL, L.; LOBUR, J. *The essentials of computer organization and architecture (2. ed.)*. [S.l.]: Jones and Bartlett Publishers, 2006. I-XXXII, 1-799 p. ISBN 978-0-7637-3769-6.

OH BYEONG-DEOK KIM, M.-K. Y. G.-Y. S. Y.-J. implementation of a dmac using systemc. *The 23rd International Technical Conference on Circuits/Systems, Computers and Communications*, p. 777–780, 2008.

O'NILS, M.; JANTSCH, A. Synthesis of dma controllers from architecture independent descriptions of hw/sw communication protocols. In: *VLSI Design, 1999. Proceedings. Twelfth International Conference On*. [S.l.: s.n.], 1999. p. 138 –145.

PATTERSON, D. A.; HENNESSY, J. L. *Computer Organization and Design: The Hardware/Software Interface*. 3rd. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007. ISBN 0123706068, 9780123706065.

STALLINGS, W. *Computer Organization and Architecture*. 6th. ed. [S.l.]: Prentice Hall Professional Technical Reference, 2002. ISBN 0130351199.

VAHID, F. *Dalton Project*. 2011. [Online; accessed 16-Maio-2011]. Disponível em: <http://www.cs.ucr.edu/ dalton/>.

VAHID, F.; GIVARGIS, T. *Embedded system design - a unified hardware / software introduction.* [S.l.]: Wiley-VCH, 2002. I-XXI, 1-324 p. ISBN 978-0-471-45303-1.

WOLF, W.; KAUFMANN, M. *Computers as Components*. [S.l.]: Morgan Kaufmann, 2001. ISBN 1558606939.

YUAN, H.; CHEN, H.; BAI, G. An improved dma controller for high speed data transfer in mpu based soc. In: *Solid-State and Integrated Circuits Technology, 2004. Proceedings. 7th International Conference on*. [S.l.: s.n.], 2004. v. 2, p. 1372 – 1375 vol.2.

ANEXO A – Artigo

Projeto de um Controlador de Acesso Direto à Memória de Multi Canais

Carlos E. de Campos, José Luís Güntzel
{cadu903,guntzel}@inf.ufsc.br
Laboratório de Automação de Projetos de Sistemas. (LAPS)
Universidade Federal de Santa Catarina, Florianópolis, Brasil

Abstract

Este artigo mostra a importância de um controlador DMA, as definições utilizadas para elaboração do controlador and presents a implementation of Direct Memory Access Controler (DMAC) IP. Os dados de prototipação para FPGA foram utilizados a ferramenta Quartus II da Altera para o componente EP2C35F672C6 da familia Cyclone II e prototipado para ASIC através da ferramenta Design Compiler da Synopsys utilizando uma biblioteca Standard-Cells TSMC 45 nm.

1. Introdução

Sistemas Embarcados (ES - *Embedded Systems*) são sistemas que possuem em sua composição sistemas computacionais desenvolvidos para atender tarefas especificas com restrições iguais ou mais rígidas do que computadores genéricos, geralmente embutidos em um produto maior e não visíveis ao usuário. Um sistema em chip (SoC - *System-on-Chip*) contém componentes computacionais como processador, memória, controlador de acesso direto a memória (DMAC - *Direct Memory Access Controller*), dispositivos de entrada / saída integrados em um único *chip*. Exemplos de aplicações que utilizam SoC em ES são: injeção eletrônica, microondas, sistemas médicos, entre outros. O trabalho implementa um dispositivo de acesso direto a memória (DMA - *Direct Memory Access*, cujo modelo de DMA conta com um dispositivo dedicado para tratar requisições de transferência, denominado DMAC. Necessitando do processador apenas nas etapa de configuração do canal, no início e no término da transferência, estando livre durante a ocorrência da transferência para executar outras operações.

Assim, o resultado principal deste trabalho foi a caracterização do comportamento funcional da interface do DMAC, definido por um conjunto de estímulos aplicado às suas entradas e o respectivo conjunto de resultados às suas saídas. Tal caracterização é crucial para o projeto de validação do DMAC em nível de transferência entre registradores (RTL - *Register-Transfer Level*) com vistas à sua prototipação em *Field-Programmable Gate Array* (FPGA) e em silício através de um circuito integrado de aplicação especifica (ASIC - *Application-Specific Integrated Circuit*), as quais são objetos do trabalho.

2. Acesso Direto a Memória

O acesso direto a memória (DMA - Direct Memory Access) exige um dispositivo dedicado para tratar requisições de transferência, denominado controlador de acesso direto a memória (DMAC - Direct Memory Access Controller). Nesta política de transferência, o processador tem contato com o DMAC na etapa de configuração do canal, no início e no término da transferência, ficando livre para executar outras operações enquanto a transferência ocorre [1][4]. Para efetuar a configuração, é necessário informar um endereço de origem, no qual será efetuada a leitura, informar um endereço de destino, no qual será efetuada a escrita, informar o número de palavras a transmitir e informar os sinais de configuração específicos da transação, como tamanho da palavra, modo de transferência, entre outros.

Um DMAC pode oferecer até três modos básicos de operação, os quais visam atender eficientemente cada classe de dispositivo. As nomenclaturas variam dependendo da bibliográfia. Foi assumida a nomenclatura utilizada por [3]: Modo de Transferência Simples, Modo de Transferência em Bloco e Transferência de Transferência por Demanda.

O DMA pode oferecer até quatro tipos de transação: Memória para Memória, Memória para Dispositivo, Dispositivo para Memória e Dispositivo para Dispositivo.

3. Implementação

A especificação do DMAC para sistemas embarcados teve como base o DMAC do projeto Dalton [11], e refinadas com base no artigo [9], livros [1] [2] [3] [4] [5] [6] [7] [8], e *framework* de DMAC da IEEE 1212.1-1993 [10], resultando nas características abaixo:

Numero de canais: Como os sistemas computacionais embarcados podem possuir mais de uma dezena de blocos que necessitem realizar transferências de dados com memória ou entre si, optou-se por definir um número de canais entre 1 e 32. Assim, o projeto baseou-se no princípio da modularidade, de modo a facilitar a adaptação da arquitetura para um número distinto de canais (desde que este número esteja entre 1 e 32). A

ferramenta criada para gerar descrições RTL da arquitetura de DMAC possui como um de seus parâmetros de entrada o número de canais.

Ordem de prioridade: No projeto do DMAC foi estipulada a ordem de prioridade sendo fixa, tendo o canal de ordem mais significativa a menor prioridade.

Barramento: Dois barramentos, sendo um o barramento do CPU (*Front-end*) e outro o barramento de E/S (*Back-end*). A escolha foi uma decisão de projeto visando não comprometer a velocidade do barramento do CPU em detrimento da velocidade dos dispositivos de E/S.

Tamanho da Palavra no barramento de Entrada / Saída (E/S) (*Back-end*): Suporte a 8, 16, 32 e 64 bits. Fornece ao sistema suporte para comunicar sistemas com tamanhos de palavras de 8, 16, 32 e 64 bits.

Tamanho da Palavra no barramento da CPU (*Front-end*): Suporte 32 bits. Uma decisão de projeto que entende que no barramento front-end somente irá haver transações de 32 bits, isto gera simplificações na lógica resultando em menos componentes lógicos.

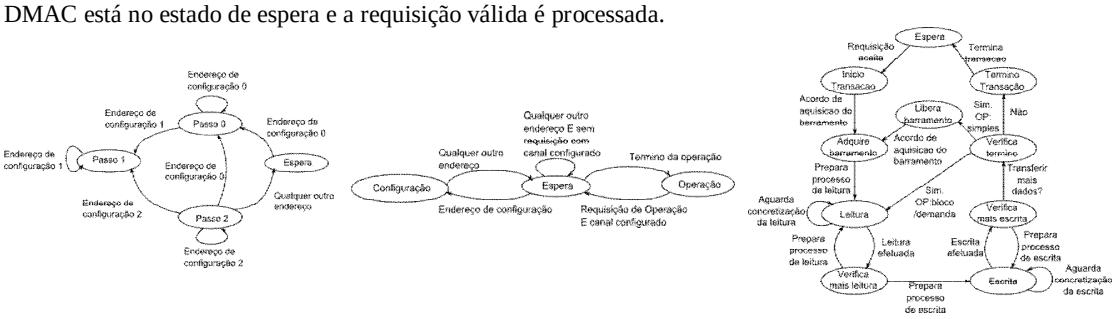
Modo de transferência: Suporte aos três modos: Simples, Bloco e Demanda. O suporte as três modos de endereçamento aumenta o potencial de aplicação do DMAC aos sistemas embarcados.

Número de transferências por transação: 2^{16} transferências. Por decisão de projeto o registrador interno que efetua o gerenciamento de transações possui 16 bits.

Tamanho do *buffer* interno: 1Kb. O uso de buffer interno permite a transferência memória-memória, bastante comum em sistemas computacionais embarcados. O tamanho de 1kb permite que blocos de tamanho razoável, no contexto de sistemas embarcados, sejam armazenados.

Modo de Operação: Suporte aos 4 modos: Memória para memória; Memória para E/S; E/S para memória; E/S para E/S.

O DMAC apresenta dois comportamentos básicos, denominados de CONFIGURAÇÃO (Figura 1 a) e OPERAÇÃO (Figura 1 c), os quais podem ser atingidos a partir da do estado de inicialização (Figura 1 b). O DMAC permanece no estado de espera enquanto não ocorre nenhum evento que ative um destes comportamentos. O comportamento CONFIGURAÇÃO modela o DMAC quando é efetuada/alterada a configuração de um canal. Sua ativação ocorre quando o DMAC encontra-se no estado de espera e é escrito um endereço no barramento de endereço superior ao qual o DMAC foi mapeado. O comportamento OPERAÇÃO modela o DMAC quando uma requisição válida é processada, sua ativação ocorre quando o DMAC está no estado de espera e a requisição válida é processada.



(a) (b) (c)

Figura 1 – Comportamento do DMAC.

A Figura 2 ilustra a disposição geral da arquitetura proposta e expõe a arquitetura de um DMAC de quatro canais, contendo um decodificador, uma região de armazenamento de dados do canal e o *datapath* com uma região de *buffer* acoplado. A ligação dos sinais está simplificada para facilitar o entendimento da arquitetura, o sentido das fechas indicam o sentido do sinal, os retângulos indicam os blocos de arquitetura que serão descritos abaixo e os quadrados representam os registradores internos dos blocos.

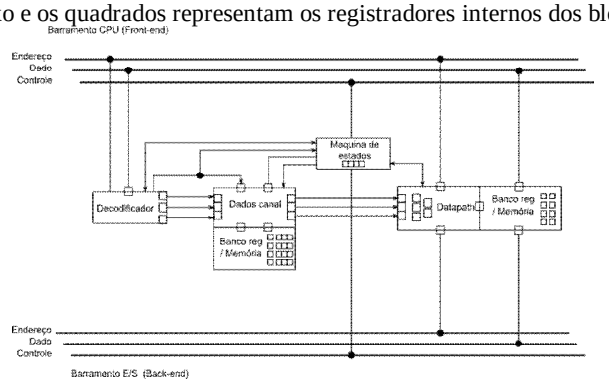


Figura 2 – Arquitetura do DMAC.

4. Resultados de Síntese

Este capítulo apresenta os resultados da síntese de duas versões de DMAC, uma com 4 canais e outra com 32 canais, para ASICs e para FPGA da Altera. No caso da síntese para FPGAs, cada uma destas duas versões foi sintetizada de duas formas: uma fazendo uso de blocos SRAM do FPGA para implementar o *buffer* de dados e outra usando banco de registradores. Os resultados da síntese são apresentados e comentados. Para permitir a síntese das duas versões de DMAC para ASIC e para FPGA foram criadas 4 descrições de DMAC:

DMAC A: Versão voltada para síntese para FPGA. Possui 4 canais e o buffer de dados é implementado como um bloco de memória SRAM do FPGA.

DMAC B: Versão voltada para síntese para FPGA. Possui 32 canais e o buffer de dados é implementado como um bloco de memória SRAM do FPGA.

DMAC C: Versão que pode ser usada tanto para síntese para ASIC quanto para FPGA. Possui 4 canais e o buffer de dados é implementado como um banco de registradores.

DMAC D: Versão que pode ser usada tanto para síntese para ASIC quanto para FPGA. Possui 32 canais e o buffer de dados é implementado como um banco de registradores.

As descrições A e C foram geradas com parâmetros similares ao DMAC 8237ª[12] da Intel (por exemplo, número de canais), de modo a permitir uma análise do impacto no uso de recursos, no desempenho e na potência em decorrência da maneira como o buffer de dados é implementado. As descrições B e D tem o intuito de permitir uma análise do impacto do número de canais no uso de recursos, no desempenho e na potência. As descrições do DMAC foram sintetizadas para FPGA e ASIC. A síntese para FPGA utilizou a ferramenta Quartus IIWeb Edition da Altera, assumindo o componente EP2C35F672C6 da família Cyclone II da Altera. A escolha deste componente deveu-se à disponibilidade de placas de prototipação DE2, as quais utilizam tal componente (embora não tenha sido possível realizar a validação do DMAC proposto em placa). A síntese para ASIC utilizou a ferramenta *Synopsys Design Compiler* em modo *Topographical*, utilizando uma biblioteca *Standard-Cells* de 45nm da TSMC.

A Tabela 1 mostra os resultados obtidos pela síntese das descrições em FPGA. O impacto do tipo de implementação do *buffer* é facilmente percebido, como pode ser visto comparando as versões A e C, cujo único diferencial é o mapeamento da área de *buffer*. A utilização de recursos lógicos passa de 1431 para 13392, um aumento de aproximadamente 835,85%. Ao comparar a potência dissipada, o aumento é considerável, passando de 160,75 mW para 494,74 mW, um acréscimo de aproximadamente 207,77%. A frequência de funcionamento também é afetada, passando de 84,76 MHz para 59,06 MHz, uma redução de aproximadamente 30,32%. Os resultados da comparação das versões B e D seguem a mesma tendência da comparação entre as versões A e C. Comparando as versões A e B, cujo único diferencial é a definição do número de canais, a utilização de unidades lógicas aumenta de 1431 para 4596, um aumento de aproximadamente 221,17% enquanto o número de canais passa de 4 para 32, um acréscimo de 700%. A dissipação de potência passa de 160,75 mW para 198,04 mW, um acréscimo de aproximadamente 23,19%. A frequência de funcionamento passa de 84,76 MHz para 72,71 MHz, uma redução de 14,21%.

	DMAC A	DMAC B	DMAC C	DMAC D
Área (LUT[1])	1431 (4,30%)	4596 (13,83%)	13392 (40,32%)	16697 (50,27%)
Potência (mW)	160,75	198,04	494,74	520,37
Frequência (MHz)	84,76	72,71	59,06	56,19

Tabela 1: Síntese das descrições em FPGA.

A Tabela 2 mostra os resultados obtidos da síntese das versões C e D para ASIC. As versões A e B não foram sintetizadas uma vez que elas pressupõem o uso de recursos específicos do FPGA. As comparações que seguem tomam a versão C como referência. A área aumenta aproximadamente 29,25% enquanto o número de canais aumenta 7 vezes. A potência total aumenta aproximadamente em 37,30%. A frequência cai em aproximadamente 29,55%.

	DMAC C	DMAC D
Área (μm^2)	60697	78453
Potência Total (mW)	2,60	3,57
Frequência (MHz)	322,58	227,27

Tabela 2: Síntese das descrições em ASIC.

5. Conclusões e Trabalhos Futuros

Através das descrições geradas foi possível estimar o impacto da implementação do *buffer* de dados no uso de recursos, desempenho e potência do DMAC. Também foi possível estimar o impacto do número de canais sobre estes mesmos parâmetros. Entre as descrições sintetizadas em FPGA, relativos à área: Nas descrições de 4 canais, observou-se o decréscimo de aproximadamente 89,31% da descrição A em relação à descrição C. Nas descrições de 32 canais, observou-se o decréscimo de aproximadamente 72,47% da descrição B em relação à descrição D. Nas descrições ao qual a área de buffer foi mapeada para SRAM da FPGA,

observou-se o decréscimo de aproximadamente 68,86% da descrição A em relação à descrição B. Nas descrições ao qual a área de buffer foi mapeada para banco de registradores, observou-se o decréscimo de aproximadamente 19,79% da descrição C em relação à descrição D. Entre as descrições sintetizadas em ASIC, relativos à frequência: Nas descrições ao qual a área de buffer foi mapeada para banco de registradores, observou-se o decréscimo de aproximadamente 22,63% da descrição C em relação à descrição D. Não foi possível efetuar comparações entre as versões C e D de ASIC relativo à FPGA devido a não possuírem a mesma métrica de medida.

Entre as descrições sintetizadas em FPGA, relativos à potência: Nas descrições de 4 canais, observou-se o decréscimo de aproximadamente 67,51% da descrição A em relação à descrição C. Nas descrições de 32 canais, observou-se o decréscimo de aproximadamente 61,94% da descrição B em relação à descrição D. Nas descrições ao qual a área de buffer foi mapeada para SRAM da FPGA, observou-se o decréscimo de aproximadamente 18,83% da descrição A em relação à descrição B. Nas descrições ao qual a área de buffer foi mapeada para banco de registradores, observou-se o decréscimo de aproximadamente 4,93% da descrição C em relação à descrição D. Entre as descrições sintetizadas em ASIC, relativos à frequência: Nas descrições ao qual a área de buffer foi mapeada para banco de registradores, observou-se o decréscimo de aproximadamente 27,17% da descrição C em relação à descrição D. Comparando os resultados entre as duas sínteses: Para descrição C observou-se o decréscimo de aproximadamente 99,47% do resultado da síntese ASIC em relação a síntese FPGA. Para descrição D observou-se o decréscimo de aproximadamente 99,31% do resultado da síntese ASIC em relação à síntese FPGA.

Entre as descrições sintetizadas em FPGA, relativos à frequência: Nas descrições de 4 canais, observou-se o acréscimo de aproximadamente 43,52% da descrição A em relação à descrição C. Nas descrições de 32 canais, observou-se o acréscimo de aproximadamente 29,40% da descrição B em relação à descrição D. Nas descrições ao qual a área de buffer foi mapeada para SRAM da FPGA, observou-se o acréscimo de aproximadamente 16,57% da descrição A em relação à descrição B. Nas descrições ao qual a área de buffer foi mapeada para banco de registradores, observou-se o acréscimo de aproximadamente 5,11% da descrição C em relação à descrição D. Entre as descrições sintetizadas em ASIC, relativos à frequência: Nas descrições ao qual a área de buffer foi mapeada para banco de registradores, observou-se o acréscimo de aproximadamente 41,94% da descrição C em relação à descrição D. Comparando os resultados entre as duas sínteses: Para descrição C observou-se o acréscimo de aproximadamente 446,19% do resultado da síntese ASIC em relação a síntese FPGA. Para descrição D observou-se o acréscimo de aproximadamente 304,47% do resultado da síntese ASIC em relação a síntese FPGA.

Como possibilidade de trabalhos futuros, a arquitetura do DMAC poderia ser modificada para permitir ao barramento superior trabalhar com tamanhos de palavras diferentes de 32 bits, visto a necessidade de operar com unidades de dados diferentes de 32 bits e/ou unidades de processamento diferentes de 32 bits. Tal alteração arquitetural proporcionaria maior portabilidade ao DMAC. Outra possibilidade é a prototipação do DMAC na placa de FPGA, com 2 ou mais blocos de memória, alguns blocos aceleradores e/ou blocos de interfaceamento (como Universal Asynchronous Receiver/Transmitter (UART), Universal Serial Bus (USB)) e o DMAC projetado.

6. Referencias

[1] VAHID, F.; GIVARGIS, T. Embedded system design - a unified hardware / software introduction. 2002.

[2] MARTIN, G.; ESL Design and Verification: A Prescription for Electronic System Level Methodology. 2007.

[3] Heath, Steve, Embedded Systems Design, 2nd Edition, Publisher: Newlines, 2003

[4] Null, Linda, The essentials of computer organization and architecture, Jonas and Bartlett, 2005

[5] PATTERSON, D. A.; HENNESSY, J. L. Computer Organization and Design: The Hardware/Software Interface. 3rd. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007

[6] STALLINGS, W. Computer Organization and Architecture. 6th, 2002.

[7] DANDAMUDI, S. Fundamentals of computer organization and design. Springer,2003.

[8] WOLF, W.; KAUFMANN, M. Computers as Components. [S.l.]: Morgan Kaufmann, 2001.

[9] O'NILS, M, A. Synthesis of dma controllers from architecture independent descriptions of hw/sw communication protocols. In: VLSI Design, 1999. Proceedings. Twelfth International Conference On. 1999. p. 138 –145.

[10] CROWELL, e. a. J. Ieee standard for communicating among processors and peripherals using shared memory (dma). IEEE Std 1212.1-1993, 1994.

[11] Vahid, Frank, John Dalton Project http://www.cs.ucr.edu/~dalton/ access at 16/05/2011

[12] Datasheet Intel 8237A – DMA Controller.

ANEXO B – Código Fonte do Gerador Automático (PHP)

B.0.1 gera_tudo.php

```
<?
include ("funcoes.php");
include ("decodificador.php");
include ("dados_canal.php");
include ("centro_operacao.php");
include ("maquina_estados.php");
include ("dmac.php");
include ("utilidades.php");
$nro_canal=$_GET['nro_canal'];
$gerar_tb=$_GET['gerar_tb'];
$projeto_altera=$_GET['projeto_altera'];
$mapeamento=$_GET['mapeamento'];
$memoria=$_GET['memoria'];

echo "
<html>
␣␣␣␣␣␣␣␣<title>␣Auto␣gera&ccedil;&atilde;o␣de␣controlador␣DMA␣
    </title>
␣␣␣␣␣␣␣␣<pre>

␣␣␣␣␣␣␣␣Clique␣no␣link␣abaixo␣e␣fa&ccedil;a␣o␣download␣do␣
    arquivo␣com␣o␣projeto␣do␣IP␣do␣controlador␣DMA

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<a␣href=\"arquivos/dmac_ip_core.zip\">
    dmac_ip_core.zip</a>
```

```
";
apagar(); // apaga tudo
//gerar decodificador
decodificador($mapeamento,$gerar_tb,$nro_canal,true,false);
echo "  <a href=\"./arquivos/decoder.vhd\">decoder.vhd</a>";
if ($gerar_tb)
        echo "\t\t-\t<a href=\"./arquivos/tb_decoder.vhd\">
            tb_decoder.vhd</a>\n";
else
        echo"\n";
dados_canal($gerar_tb,$nro_canal,1,false);
echo "  <a href=\"./arquivos/channel_data.vhd\">channel_data.
    vhd</a>";
if ($gerar_tb)
        echo "\t-\t<a href=\"./arquivos/tb_channel_data.vhd\">
            tb_channel_data.vhd</a>\n";
else
        echo"\n";
centro_operacao($gerar_tb,$nro_canal,1,0);
echo "  <a href=\"./arquivos/main_operation.vhd\">
    main_operation.vhd</a>";
if ($gerar_tb)
        echo "\t-\t<a href=\"./arquivos/tb_main_operation.vhd
            \">tb_main_operation.vhd</a>\n";
else
        echo"\n";
maquina_estados($gerar_tb,$nro_canal,1,0);
echo "  <a href=\"./arquivos/state_machine.vhd\">state_machine.
    vhd</a>";
if ($gerar_tb)
        echo "\t-\t<a href=\"./arquivos/tb_state_machine.vhd\">
            tb_state_machine.vhd</a>\n";
else
        echo"\n";
```

```
dmac($gerar_tb,$nro_canal,1,0);
echo "  <a href=\"./arquivos/dmac.vhd\">dmac.vhd</a>";
if ($gerar_tb)
        echo "\t\t-\t<a href=\"./arquivos/tb_dmac.vhd\">tb_dmac
            .vhd</a>\n";
else
        echo"\n";
if ($memoria=="altera"){
mem_altera($gerar_tb,$nro_canal,1,0);
echo "  <a href=\"./arquivos/mem.vhd\">mem.vhd</a>";
if ($gerar_tb)
        echo "\t\t\t-\t<a href=\"./arquivos/tb_mem.vhd\">tb_mem
            .vhd</a>\n";
else
        echo"\n";
}else{
mem_reg($gerar_tb, $nro_canal,1,0);
echo "  <a href=\"./arquivos/mem.vhd\">mem.vhd</a>";
if ($gerar_tb)
        echo "\t\t\t-\t<a href=\"./arquivos/tb_mem.vhd\">tb_mem
            .vhd</a>\n";
else
        echo"\n";
echo "  <a href=\"./arquivos/buffer_block.vhd\">buffer_block.
   vhd</a>
";
}
reg($gerar_tb,$nro_canal,1,0);
echo "  <a href=\"./arquivos/reg.vhd\">reg.vhd</a>";
if ($gerar_tb)
        echo "\t\t\t-\t<a href=\"./arquivos/tb_reg.vhd\">tb_reg
            .vhd</a>\n";
else
        echo"\n";
if ($projeto_altera){
        projeto_altera($gerar_tb,$nro_canal,1,0,$memoria);
```

```
        echo "  <a href=\"./arquivos/dmac.qpf\">dmac.qpf</a>\n"
            ;
        echo "  <a href=\"./arquivos/dmac.qsf\">dmac.qsf</a>\n"
            ;
        echo "  <a href=\"./arquivos/dmac.sdc\">dmac.sdc</a>\n"
            ;
}
compactar("dmac_ip_core.zip");
echo"
        </pre>
</html>
";
?>
```

B.0.2 funcoes.php

```
<?
ini_set('display_errors', 1);
function gravar ($nome_arquivo, $conteudo){
        $caminho="./arquivos/$nome_arquivo";
        $arquivo = fopen ($caminho, "w");
        if (!$arquivo){
                echo "Erro ao abrir o arquivo $nome_arquivo\n";
                return 0;
        }
        $gravou = fwrite ( $arquivo, $conteudo);
        if (!$gravou){
                echo "Erro ao gerar arquivo $nome_arquivo\n";
                fclose($arquivo);
                return 0;
        }
        fclose($arquivo);
        chmod($caminho, 0666);
        return 1;
}
function tela ($nome_arquivo, $conteudo){
echo "
```

```
<pre>
--Arquivo␣:␣$nome_arquivo
----------␣INICIO␣DO␣ARQUIVO----------
$conteudo
----------␣TERMINO␣DO␣ARQUIVO␣----------
</pre>
";
}
function testecompacta(){
//$output = shell_exec('zip a.zip a.txt');
//echo "<pre>{$output}</pre>";
$output = shell_exec('pwd');
echo "<pre>{$output}</pre>";
$output = shell_exec('./comp.sh');
echo "<pre>{$output}</pre>";
exec("whoami",$output);
echo "<pre>{".var_dump($output)."}</pre>";
 }
function compactar($nome_arquivo){
$dir = "./arquivos/";//dirname(__FILE__)."/";
$arquivos[0]="channel_data.vhd";
$arquivos[1]="decoder.vhd";
$arquivos[2]="dmac.qpf";
$arquivos[3]="dmac.qsf";
$arquivos[4]="dmac.vhd";
$arquivos[5]="main_operation.vhd";
$arquivos[6]="mem.vhd";
$arquivos[7]="reg.vhd";
$arquivos[8]="state_machine.vhd";
$arquivos[9]="tb_decoder.vhd";
$arquivos[10]="tb_dmac.vhd";
$arquivos[11]="tb_main_operation.vhd";
$arquivos[12]="tb_mem.vhd";
$arquivos[13]="tb_reg.vhd";
$arquivos[14]="tb_state_machine.vhd";
$arquivos[15]="buffer_block.vhd";
```

```
$arquivos[16]="dmac.sdc";

$zip = new ZipArchive();
if($zip->open($dir . $nome_arquivo, ZIPARCHIVE::CREATE))
for($i=0;$i<sizeof($arquivos);$i++)
        if (file_exists($dir . $arquivos[$i])){
                $zip->addfile($dir . $arquivos[$i]);
        }
$zip->close();
        return 1;
}
function apagar(){
$dir = "./arquivos/";//dirname(__FILE__)."/";
$arquivos[0]="channel_data.vhd";
$arquivos[1]="decoder.vhd";
$arquivos[2]="dmac.qpf";
$arquivos[3]="dmac.qsf";
$arquivos[4]="dmac.vhd";
$arquivos[5]="main_operation.vhd";
$arquivos[6]="mem.vhd";
$arquivos[7]="reg.vhd";
$arquivos[8]="state_machine.vhd";
$arquivos[9]="tb_decoder.vhd";
$arquivos[10]="tb_dmac.vhd";
$arquivos[11]="tb_main_operation.vhd";
$arquivos[12]="tb_mem.vhd";
$arquivos[13]="tb_reg.vhd";
$arquivos[14]="tb_state_machine.vhd";
$arquivos[15]="buffer_block.vhd";
$arquivos[16]="dmac_ip_core.zip";
$arquivos[17]="dmac.sdc";

for($i=0;$i<sizeof($arquivos);$i++)
        if (file_exists($dir . $arquivos[$i])){
                unlink($dir . $arquivos[$i]);
        }
```

```
}
function int_para_binario($inteiro){
        switch ($inteiro){
                case 0:
                        return "00000";
                case 1:
                        return "00001";
                case 2:
                        return "00010";
                case 3:
                        return "00011";
                case 4:
                        return "00100";
                case 5:
                        return "00101";
                case 6:
                        return "00110";
                case 7:
                        return "00111";
                case 8:
                        return "01000";
                case 9:
                        return "01001";
                case 10:
                        return "01010";
                case 11:
                        return "01011";
                case 12:
                        return "01100";
                case 13:
                        return "01101";
                case 14:
                        return "01110";
                case 15:
                        return "01111";
                case 16:
```

```
                        return "10000";
                    case 17:
                        return "10001";
                    case 18:
                        return "10010";
                    case 19:
                        return "10011";
                    case 20:
                        return "10100";
                    case 21:
                        return "10101";
                    case 22:
                        return "10110";
                    case 23:
                        return "10111";
                    case 24:
                        return "11000";
                    case 25:
                        return "11001";
                    case 26:
                        return "11010";
                    case 27:
                        return "11011";
                    case 28:
                        return "11100";
                    case 29:
                        return "11101";
                    case 30:
                        return "11110";
                    case 31:
                        return "11111";
//                  case default:
//                      return "00000";
            }
}
```

```
function vetor_verificacao($valor_antes, $valor_depois,
   $nro_canal){
        for ($i=0;$i<$nro_canal;$i++){
                for ($j=0;$j<$nro_canal;$j++){
                        if ($i<$j)
                                $ret[$i][$j]=$valor_antes;
                        else
                                $ret[$i][$j]=$valor_depois;
                }
        }
        return $ret;
}
function binario_para_degray($binario,$digitos){
        $ret="";
                if ($binario== "00000")//00
                        $ret="00000000000000000000000000000001"
                           ;
                else if ($binario=="00001")//01
                        $ret="00000000000000000000000000000010"
                           ;
                else if ($binario=="00010")//02
                        $ret="00000000000000000000000000000100"
                           ;
                else if ($binario=="00011")//03
                        $ret="00000000000000000000000000001000"
                           ;
                else if ($binario=="00100")//04
                        $ret="00000000000000000000000000010000"
                           ;
                else if ($binario=="00101")//05
                        $ret="00000000000000000000000000100000"
                           ;
                else if ($binario=="00110")//06
                        $ret="00000000000000000000000001000000"
                           ;
                else if ($binario=="00111")//07
```

```
        $ret="00000000000000000000000010000000"
            ;
else if ($binario=="01000")//08
        $ret="00000000000000000000000100000000"
            ;
else if ($binario=="01001")//09
        $ret="00000000000000000000001000000000"
            ;
else if ($binario=="01010")//10
        $ret="00000000000000000000010000000000"
            ;
else if ($binario=="01011")//11
        $ret="00000000000000000000100000000000"
            ;
else if ($binario=="01100")//12
        $ret="00000000000000000001000000000000"
            ;
else if ($binario=="01101")//13
        $ret="00000000000000000010000000000000"
            ;
else if ($binario=="01110")//14
        $ret="00000000000000000100000000000000"
            ;
else if ($binario=="01111")//15
        $ret="00000000000000001000000000000000"
            ;
else if ($binario=="10000")//16
        $ret="00000000000000010000000000000000"
            ;
else if ($binario=="10001")//17
        $ret="00000000000000100000000000000000"
            ;
else if ($binario=="10010")//18
        $ret="00000000000001000000000000000000"
            ;
else if ($binario=="10011")//19
```

```
        $ret="00000000000010000000000000000000"
            ;
else if ($binario=="10100")//20
        $ret="00000000000100000000000000000000"
            ;
else if ($binario=="10101")//21
        $ret="00000000001000000000000000000000"
            ;
else if ($binario=="10110")//22
        $ret="00000000010000000000000000000000"
            ;
else if ($binario=="10111")//23
        $ret="00000000100000000000000000000000"
            ;
else if ($binario=="11000")//24
        $ret="00000001000000000000000000000000"
            ;
else if ($binario=="11001")//25
        $ret="00000010000000000000000000000000"
            ;
else if ($binario=="11010")//26
        $ret="00000100000000000000000000000000"
            ;
else if ($binario=="11011")//27
        $ret="00001000000000000000000000000000"
            ;
else if ($binario=="11100")//28
        $ret="00010000000000000000000000000000"
            ;
else if ($binario=="11101")//29
        $ret="00100000000000000000000000000000"
            ;
else if ($binario=="11110")//30
        $ret="01000000000000000000000000000000"
            ;
else if ($binario=="11111")//31
```

```
                        $ret="10000000000000000000000000000000"
                            ;
/*
        $j=0;
        for ($i=32-$digitos;$i<32;$i++){
                $novo_ret[$j]=$ret[$i];
                $j++;
        }
        return $novo_ret;
*/
        return substr($ret,-$digitos);
}
function int_para_degray($int,$tamanho){
        $ret=binario_para_degray(int_para_binario($int),
           $tamanho);
        return $ret;
}
function degray_para_binario($degray){
        $tamanho=strlen($degray)-1;
        $ok=0;
        for ($i=$tamanho;$i>=0;$i--){
                if ($degray[$i]=="1"){

                        return int_para_binario($ok);
                }
                $ok++;
        }
}
function lista_binario_dado_inteiro($inteiro){
        $ret=array();
        for ($i=0;$i<=$inteiro;$i++)
                array_push($ret,int_para_binario($i));
        return $ret;
}
function lista_habilitados_dados_canal($habilita, $sinal){
```

```
$ret[0] = "(not␣$sinal(4))␣and␣(not␣$sinal(3))␣and␣(not
    ␣$sinal(2))␣and␣(not␣$sinal(1))␣and␣(not␣$sinal(0))"
    ;//00000
$ret[1] = "(not␣$sinal(4))␣and␣(not␣$sinal(3))␣and␣(not
    ␣$sinal(2))␣and␣(not␣$sinal(1))␣and␣($sinal(0))";//
    00001
$ret[2] = "(not␣$sinal(4))␣and␣(not␣$sinal(3))␣and␣(not
    ␣$sinal(2))␣and␣($sinal(1))␣and␣(not␣$sinal(0))";//
    00010
$ret[3] = "(not␣$sinal(4))␣and␣(not␣$sinal(3))␣and␣(not
    ␣$sinal(2))␣and␣($sinal(1))␣and␣($sinal(0))";//00011
$ret[4] = "(not␣$sinal(4))␣and␣(not␣$sinal(3))␣and␣(
    $sinal(2))␣and␣(not␣$sinal(1))␣and␣(not␣$sinal(0))";
    //00100
$ret[5] = "(not␣$sinal(4))␣and␣(not␣$sinal(3))␣and␣(
    $sinal(2))␣and␣(not␣$sinal(1))␣and␣($sinal(0))";//
    00101
$ret[6] = "(not␣$sinal(4))␣and␣(not␣$sinal(3))␣and␣(
    $sinal(2))␣and␣($sinal(1))␣and␣(not␣$sinal(0))";//
    00110
$ret[7] = "(not␣$sinal(4))␣and␣(not␣$sinal(3))␣and␣(
    $sinal(2))␣and␣($sinal(1))␣and␣($sinal(0))";//00111
$ret[8] = "(not␣$sinal(4))␣and␣($sinal(3))␣and␣(not␣
    $sinal(2))␣and␣(not␣$sinal(1))␣and␣(not␣$sinal(0))";
    //01000
$ret[9] = "(not␣$sinal(4))␣and␣($sinal(3))␣and␣(not␣
    $sinal(2))␣and␣(not␣$sinal(1))␣and␣($sinal(0))";//
    01001
$ret[10] = "(not␣$sinal(4))␣and␣($sinal(3))␣and␣(not␣
    $sinal(2))␣and␣($sinal(1))␣and␣(not␣$sinal(0))";//
    01010
$ret[11] = "(not␣$sinal(4))␣and␣($sinal(3))␣and␣(not␣
    $sinal(2))␣and␣($sinal(1))␣and␣($sinal(0))";//01011
$ret[12] = "(not␣$sinal(4))␣and␣($sinal(3))␣and␣($sinal
    (2))␣and␣(not␣$sinal(1))␣and␣(not␣$sinal(0))";//
    01100
```

```
$ret[13] = "(not␣$sinal(4))␣and␣($sinal(3))␣and␣($sinal
    (2))␣and␣(not␣$sinal(1))␣and␣($sinal(0))";//01101
$ret[14] = "(not␣$sinal(4))␣and␣($sinal(3))␣and␣($sinal
    (2))␣and␣($sinal(1))␣and␣(not␣$sinal(0))";//01110
$ret[15] = "(not␣$sinal(4))␣and␣($sinal(3))␣and␣($sinal
    (2))␣and␣($sinal(1))␣and␣($sinal(0))";//01111
$ret[16] = "($sinal(4))␣and␣(not␣$sinal(3))␣and␣(not␣
    $sinal(2))␣and␣(not␣$sinal(1))␣and␣(not␣$sinal(0))";
    //10000
$ret[17] = "($sinal(4))␣and␣(not␣$sinal(3))␣and␣(not␣
    $sinal(2))␣and␣(not␣$sinal(1))␣and␣($sinal(0))";//
    10001
$ret[18] = "($sinal(4))␣and␣(not␣$sinal(3))␣and␣(not␣
    $sinal(2))␣and␣($sinal(1))␣and␣(not␣$sinal(0))";//
    10010
$ret[19] = "($sinal(4))␣and␣(not␣$sinal(3))␣and␣(not␣
    $sinal(2))␣and␣($sinal(1))␣and␣($sinal(0))";//10011
$ret[20] = "($sinal(4))␣and␣(not␣$sinal(3))␣and␣($sinal
    (2))␣and␣(not␣$sinal(1))␣and␣(not␣$sinal(0))";//
    10100
$ret[21] = "($sinal(4))␣and␣(not␣$sinal(3))␣and␣($sinal
    (2))␣and␣(not␣$sinal(1))␣and␣($sinal(0))";//10101
$ret[22] = "($sinal(4))␣and␣(not␣$sinal(3))␣and␣($sinal
    (2))␣and␣($sinal(1))␣and␣(not␣$sinal(0))";//10110
$ret[23] = "($sinal(4))␣and␣(not␣$sinal(3))␣and␣($sinal
    (2))␣and␣($sinal(1))␣and␣($sinal(0))";//10111
$ret[24] = "($sinal(4))␣and␣($sinal(3))␣and␣(not␣$sinal
    (2))␣and␣(not␣$sinal(1))␣and␣(not␣$sinal(0))";//
    11000
$ret[25] = "($sinal(4))␣and␣($sinal(3))␣and␣(not␣$sinal
    (2))␣and␣(not␣$sinal(1))␣and␣($sinal(0))";//11001
$ret[26] = "($sinal(4))␣and␣($sinal(3))␣and␣(not␣$sinal
    (2))␣and␣($sinal(1))␣and␣(not␣$sinal(0))";//11010
$ret[27] = "($sinal(4))␣and␣($sinal(3))␣and␣(not␣$sinal
    (2))␣and␣($sinal(1))␣and␣($sinal(0))";//11011
```

```
        $ret[28] = "($sinal(4))␣and␣($sinal(3))␣and␣($sinal(2))
           ␣and␣(not␣$sinal(1))␣and␣(not␣$sinal(0))"; //11100
        $ret[29] = "($sinal(4))␣and␣($sinal(3))␣and␣($sinal(2))
           ␣and␣(not␣$sinal(1))␣and␣($sinal(0))"; //11101
        $ret[30] = "($sinal(4))␣and␣($sinal(3))␣and␣($sinal(2))
           ␣and␣($sinal(1))␣and␣(not␣$sinal(0))"; //11110
        $ret[31] = "($sinal(4))␣and␣($sinal(3))␣and␣($sinal(2))
           ␣and␣($sinal(1))␣and␣($sinal(0))"; //11111
        return $ret;
}
?>
```

B.0.3 maquina_estados.php

```
<?
//include ("funcoes.php");
$gerar_tb=$_GET['gerar_tb'];
$nro_canal=$_GET['nro_canal'];
$grava=$_GET['grava'];
$tela=$_GET['tela'];

function maquina_estados($gerar_tb, $nro_canal,$grava,$tela){

$k=$nro_canal-1;

$conteudo="
library␣IEEE;
use␣IEEE.STD_LOGIC_1164.all;
USE␣ieee.std_logic_unsigned.all;

-------------------------------------------------------------------------------------------



entity␣STATE_MACHINE␣is
␣␣␣␣port(
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣sinais␣externos␣ao␣DMA
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣clk␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣in␣␣␣␣
    std_logic;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣reset␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣in␣␣␣␣
    STD_LOGIC;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣hlda␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣in␣␣␣␣
    STD_LOGIC;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣dreq␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣in␣␣␣␣
    std_logic_vector␣($k␣downto␣0);

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣db_f␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣out␣␣␣
    std_logic_vector␣(31␣downto␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣hold␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣out␣␣␣
    STD_LOGIC;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣eop␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣out␣␣␣
    STD_LOGIC;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣io_r␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣out␣␣␣
    STD_LOGIC;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣io_w␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣out␣␣␣
    STD_LOGIC;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣mem_r␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣out␣␣␣
    STD_LOGIC;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣mem_w␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣out␣␣␣
    STD_LOGIC;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣dack␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣out␣␣␣
    std_logic_vector␣($k␣downto␣0);

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣ready␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣in␣std_logic;
    ␣--␣usado␣por␣dispositivos␣lentos␣pra␣prolongar␣os␣ciclos␣de
    ␣leitura/gravacao

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣sinais␣internos␣do␣dma
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--decoder
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_decoder_de␣␣␣:␣␣␣␣␣␣␣in␣STD_LOGIC;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣r_status_de␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣in␣
    STD_LOGIC;
```

```
                        step_de                  :       in
    std_logic_vector (1 downto 0);
                        sl_w_ch_out_de  :       in
    std_logic_vector (4 downto 0);

                        en_decoder_sm   :       out STD_LOGIC;

                        -- channel data
                        en_w_ch_cd              :       out
    STD_LOGIC;
                        sl_r_ch_cd              :       out
    std_logic_vector (4 downto 0);

                        en_channel_data_sm : out std_logic;
                        --main operation

                        flags_out_co_mo                 :
    :       in std_logic_vector (15 downto 0);
                        counter_is_empty_mo             :
    :       in STD_LOGIC;
                        offset_is_full_mo               :
    :       in STD_LOGIC;
                        offset_is_empty_mo              :
    :       in STD_LOGIC;

                        sl_op_mode_mo                   :
    :       out STD_LOGIC;
                        en_w_flags_co_mo                :
    :       out STD_LOGIC;
                        en_w_offset_co_mo               :
    :       out STD_LOGIC;
                        en_w_counter_co_mo              :
    :       out STD_LOGIC;
                        sl_w_counter_co_mo              :
    :       out STD_LOGIC;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_w_addr_f_co_mo␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣␣out␣STD_LOGIC;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_addr_f_co_mo␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣␣out␣STD_LOGIC;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_w_addr_b_co_mo␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣␣out␣STD_LOGIC;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_addr_b_co_mo␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣␣out␣STD_LOGIC;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_w_addr_f_f_co_mo␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣␣out␣STD_LOGIC;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_w_addr_f_b_co_mo␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣␣out␣STD_LOGIC;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_w_addr_b_f_co_mo␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣␣out␣STD_LOGIC;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_w_addr_b_b_co_mo␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣␣out␣STD_LOGIC;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_w_db_f_co_mo␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣␣out␣STD_LOGIC;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_w_db_b_co_mo␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣␣out␣STD_LOGIC;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_r_db_f_co_mo␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣␣out␣STD_LOGIC;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_r_db_b_co_mo␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣␣out␣STD_LOGIC;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_main_operation_sm␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣␣out␣std_logic

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣);
end␣STATE_MACHINE;

ARCHITECTURE␣BHV_STATE_MACHINE␣OF␣STATE_MACHINE␣IS
```

```
                COMPONENT reg is Generic (      N : Integer
    :=32 );
                port(
                        reset , clk , carga : IN   STD_LOGIC;
                        D :   IN      std_logic_vector (N-1
    DOWNTO 0);
                        Q :   OUT     std_logic_vector (N-1
    DOWNTO 0)
                );
                end component;

                type STATE_TYPE is (
                        ESPERA,
                        PASSO_0,
                        PASSO_1,
                        PASSO_2,
                        ERRO,
                        STATUS,
                        TRANSACAO_SOLICITADA,
    BARRAMENTO_SOLICITADO,
                        BARRAMENTO_GARANTIDO,
    TRANSACAO_CONFIRMADA,   OP_LEITURA_A,
                        OP_LEITURA_B,
                        OP_CICLO_LEITURA,
                        VERIFICA_MAIS_LEITURA,  OP_ESCRITA_A,
                        OP_ESCRITA_B,
                        OP_CICLO_ESCRITA,
                        FAZER_MAIS_ESCRITA,
    TESTA_CONTADOR,                 FINALIZACAO
                );
                signal estado_atual    : STATE_TYPE;

                -- constantes de operacao interna
                constant HIGH                                   :       std_logic := '1';
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣constant␣ALL_0␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic_vector($k␣downto␣0):=(others=>'0')
    ;--9

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣constant␣MAX_CHANNEL␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:
    ␣␣␣␣␣␣␣std_logic_vector(4␣downto␣0):=\"".int_para_binario($k
    )."\";

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣constant␣ST_OK␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic_vector(1␣downto␣0):=\"00\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣constant␣ST_EM_PROCESSO␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣
    std_logic_vector(1␣downto␣0):=\"01\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣constant␣ST_ERRO␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic_vector(1␣downto␣0):=\"10\";

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣constant␣ST_CANAL_INVALIDO␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣
    std_logic_vector(24␣downto␣0)
    :=\"0000000000000000000000000\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣constant␣ST_CANAL_PASSO_INVALIDO␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣
    std_logic_vector(24␣downto␣0)
    :=\"0000000000000000000000001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣constant␣ST_CONFIGURANDO_CANAL_PASSO_0␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic_vector(24␣downto␣
    0):=\"0000000000000000000000010\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣constant␣ST_CONFIGURANDO_CANAL_PASSO_1␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic_vector(24␣downto␣
    0):=\"0000000000000000000000011\";
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣constant␣ST_CONFIGURANDO_CANAL_PASSO_2␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic_vector(24␣downto␣
    0):=\"000000000000000000000100\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣constant␣ST_CANAL_DIFERENTE_PASSO_0␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic_vector(24␣
    downto␣0):=\"000000000000000000000101\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣constant␣ST_CONFIGURACAO_ABANDONADA␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic_vector(24␣
    downto␣0):=\"000000000000000000000110\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣constant␣ST_CONFIGURACAO_COMPLETA␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic_vector(24␣
    downto␣0):=\"000000000000000000000111\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣constant␣ST_TRANSACAO_NAO_CONFIGURADA␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic_vector(24␣downto␣
    0):=\"000000000000000000001000\";

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣sinais␣de␣apoio␣aos␣registradores
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_in_data_in_sm␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic_vector(28␣downto␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_in_data_out_sm␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic_vector(28␣downto␣0);

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_out_data_in_sm␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic_vector(31␣downto␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_out_data_out_sm␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic_vector(31␣downto␣0);

--␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_internal_data_in_sm␣␣␣␣␣␣␣␣␣␣:
    ␣␣␣␣␣␣␣std_logic_vector(31␣downto␣0);
--␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_internal_data_out_sm␣␣␣␣␣␣␣␣␣:
    ␣␣␣␣␣␣␣std_logic_vector(31␣downto␣0);

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_status_in_sm␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic_vector(31␣downto␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_status_out_sm␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic_vector(31␣downto␣0);
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_en_w_status_sm␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_en_w_status_db_f_sm␣␣:␣␣␣␣␣␣␣
    std_logic;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_dreq_in_sm␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic_vector($k␣downto␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_dreq_out_sm␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic_vector($k␣downto␣0);
——␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_en_w_dreq_sm␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_dack_in_sm␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic_vector($k␣downto␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_dack_out_sm␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic_vector($k␣downto␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_en_w_dack_sm␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_dcfg_in_sm␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic_vector($k␣downto␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_dcfg_out_sm␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic_vector($k␣downto␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_en_w_dcfg_sm␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_en_w_ch_sm␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_ch_session_in_sm␣␣␣␣␣␣␣␣␣␣␣:
    ␣␣␣␣␣␣␣std_logic_vector(4␣downto␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_ch_session_out_sm␣␣␣␣␣␣␣␣␣␣:
    ␣␣␣␣␣␣␣std_logic_vector(4␣downto␣0);
```

```
␣␣␣␣␣␣␣␣----␣sinais␣de␣operacao␣da␣maquina␣de␣estados␣de␣apoio␣
    a␣entrada␣e␣saida␣de␣sinais
--␣␣␣␣␣␣--␣out
--␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣externo
--␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_hold_sm␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic;
--␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_eop_sm␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣
    std_logic;
--␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_io_r_sm␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic;
--␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_io_w_sm␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic;
--␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_mem_r_sm␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic;
--␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_mem_w_sm␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic;
--␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣decoder
--␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_en_decoder_sm␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic;
--␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣ch␣data
--␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_en_w_ch_cd_sm␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic;
--␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_sl_r_ch_cd_sm␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic_vector(4␣
    downto␣0);
--␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_en_channel_data_sm␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic;
--␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣main␣operation
--␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_sl_op_mode_mo_sm␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic;
--␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_en_w_flags_co_mo_sm␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic;
--␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_en_w_offset_co_mo_sm␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic;
```

```
--                    signal sig_en_w_counter_co_mo_sm
                      : std_logic;
--                    signal sig_sl_w_counter_co_mo_sm
                      : std_logic;
--                    signal sig_en_w_addr_f_co_mo_sm
                  : std_logic;
--                    signal sig_sl_w_addr_f_co_mo_sm
                  : std_logic;
--                    signal sig_en_w_addr_b_co_mo_sm
                  : std_logic;
--                    signal sig_sl_w_addr_b_co_mo_sm
                  : std_logic;
--                    signal sig_en_w_addr_f_f_co_mo_sm
                : std_logic;
--                    signal sig_en_w_addr_f_b_co_mo_sm
                : std_logic;
--                    signal sig_en_w_addr_b_f_co_mo_sm
                : std_logic;
--                    signal sig_en_w_addr_b_b_co_mo_sm
                : std_logic;
--                    signal sig_en_w_db_f_co_mo_sm
                    : std_logic;
--                    signal sig_en_w_db_b_co_mo_sm
                    : std_logic;
--                    signal sig_en_r_db_f_co_mo_sm
                    : std_logic;
--                    signal sig_en_r_db_b_co_mo_sm
                    : std_logic;
--                    signal sig_en_main_operation_sm
                : std_logic;
--            ---- sinais de operacao internos.
--                    signal sig_ch_temp
                                        : 
    std_logic_vector(4 downto 0);

BEGIN
```

```
␣␣␣␣␣␣␣␣−−␣logica␣de␣proximo␣estado
␣␣␣␣␣␣␣␣out_data_sm␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣reg␣generic␣map
   ␣(␣N␣=>␣32)␣port␣map␣(␣␣␣reset ,␣clk ,␣HIGH,␣
   sig_out_data_in_sm ,␣sig_out_data_out_sm);
␣␣␣␣␣␣␣␣status_sm␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣reg␣generic␣map␣(␣N␣=>␣
   32)␣port␣map␣(␣␣␣reset ,␣clk ,␣sig_en_w_status_sm ,␣
   sig_status_in_sm ,␣sig_status_out_sm);
␣␣␣␣␣␣␣␣ch_session_sm␣␣␣:␣␣␣␣␣␣␣reg␣generic␣map␣(␣N␣=>␣5␣)␣port
   ␣map␣(␣␣␣reset ,␣clk ,␣sig_en_w_ch_sm ,␣sig_ch_session_in_sm ,
   sig_ch_session_out_sm);
␣␣␣␣␣␣␣␣dreq_sm␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣reg␣generic␣map␣(␣N␣=>␣
   $nro_canal␣)␣port␣map␣(␣␣reset ,␣clk ,␣HIGH,␣sig_dreq_in_sm ,␣
   sig_dreq_out_sm);
␣␣␣␣␣␣␣␣dack_sm␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣reg␣generic␣map␣(␣N␣=>␣
   $nro_canal␣)␣port␣map␣(␣␣reset ,␣clk ,␣sig_en_w_dack_sm ,␣
   sig_dack_in_sm ,␣sig_dack_out_sm);
␣␣␣␣␣␣␣␣dcfg_sm␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣reg␣generic␣map␣(␣N␣=>␣
   $nro_canal␣)␣port␣map␣(␣␣reset ,␣clk ,␣sig_en_w_dcfg_sm ,␣
   sig_dcfg_in_sm ,␣sig_dcfg_out_sm);

␣␣␣␣␣␣␣␣sl_r_ch_cd<=sig_ch_session_out_sm;

␣␣␣␣␣␣␣␣process␣(sig_en_w_status_db_f_sm ,␣sig_status_out_sm)
␣␣␣␣␣␣␣␣begin
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_en_w_status_db_f_sm='1')␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣db_f␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
   ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣
   sig_status_out_sm;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣db_f␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
   ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣(
   others=>'Z');
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣end␣process;

␣␣␣␣␣␣␣␣−−␣in
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--externo
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_dreq_in_sm␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣dreq;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣dack␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_dack_out_sm;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_in_data_in_sm(0)␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣hlda;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣decoder
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_in_data_in_sm(1)␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_decoder_de;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_in_data_in_sm(2)␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣r_status_de;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_in_data_in_sm(04␣downto␣03)␣<=
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣step_de;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_in_data_in_sm(09␣downto␣05)␣<=
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_ch_out_de;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣main␣operation
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_in_data_in_sm(25␣downto␣10)␣<=
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣flags_out_co_mo;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_in_data_in_sm(26)␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣counter_is_empty_mo;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_in_data_in_sm(27)␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣offset_is_full_mo;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_in_data_in_sm(28)␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣offset_is_empty_mo;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_in_data_out_sm␣␣␣␣␣␣<=␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_in_data_in_sm;

␣␣␣␣␣␣␣␣--␣out
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣externo
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣hold␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_out_sm(00);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣eop␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_out_sm(01);
```

```
                        io_r
                <=              sig_out_data_out_sm(02);
                        io_w
                <=              sig_out_data_out_sm(03);
                        mem_r
                <=              sig_out_data_out_sm(04);
                        mem_w
                <=              sig_out_data_out_sm(05);
                -- decoder
                        en_decoder_sm                    <=
              sig_out_data_out_sm(06);
                -- ch data
                        en_w_ch_cd
        <=              sig_out_data_out_sm(07);
                --      sl_r_ch_cd
        <=              sig_out_data_out_sm(12 downto 08);
                        en_channel_data_sm      <=
        sig_out_data_out_sm(13);
                -- main operation
                        sl_op_mode_mo                    <=
              sig_out_data_out_sm(14);
                        en_w_flags_co_mo                 <=
              sig_out_data_out_sm(15);
                        en_w_offset_co_mo       <=
        sig_out_data_out_sm(16);
                        en_w_counter_co_mo      <=
        sig_out_data_out_sm(17);
                        sl_w_counter_co_mo      <=
        sig_out_data_out_sm(18);
                        en_w_addr_f_co_mo       <=
        sig_out_data_out_sm(19);
                        sl_w_addr_f_co_mo       <=
        sig_out_data_out_sm(20);
                        en_w_addr_b_co_mo       <=
        sig_out_data_out_sm(21);
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_addr_b_co_mo␣␣␣␣␣␣␣␣␣␣␣␣␣<=
    ␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_out_sm(22);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_w_addr_f_f_co_mo␣␣␣␣␣<=␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_out_data_out_sm(23);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_w_addr_f_b_co_mo␣␣␣␣␣<=␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_out_data_out_sm(24);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_w_addr_b_f_co_mo␣␣␣␣␣<=␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_out_data_out_sm(25);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_w_addr_b_b_co_mo␣␣␣␣␣<=␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_out_data_out_sm(26);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_w_db_f_co_mo␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=
    ␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_out_sm(27);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_w_db_b_co_mo␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=
    ␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_out_sm(28);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_r_db_f_co_mo␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=
    ␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_out_sm(29);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_r_db_b_co_mo␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=
    ␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_out_sm(30);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_main_operation_sm␣<=␣␣␣␣␣␣␣␣
    sig_out_data_out_sm(31);

--␣logica␣de␣proximo␣estado
␣␣␣␣␣␣␣␣process␣(
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣clk,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣reset,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_in_data_out_sm,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_dreq_out_sm,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_out_sm
␣␣␣␣␣␣␣␣)
␣␣␣␣␣␣␣␣begin
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(reset='1')␣then

--␣estado␣\"000\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣ESPERA;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(clk'EVENT␣and␣clk='1')␣then
```

```
                        case (estado_atual) is
                                when ESPERA =>
                                        if (sig_in_data_out_sm
    (1)='1') then --en_decoder = 1
                                                if (
    sig_in_data_out_sm(2)='1') then -- r_status_de = '1'
-- estado \"011\";
                                                        
    estado_atual                            <=              
    STATUS;
                                                else -- com 
    decoder ativo se nao for status so pode ser configuracao
                                                        if (
    sig_in_data_out_sm(09 downto 05)>max_channel) then -- 
    verifica existencia do canal
-- estado \"012\";
                                                        
            estado_atual                            <=      
            ERRO;
                                                        else
                                                        
            if (sig_in_data_out_sm(4 downto 3)=\"00\") then -- 
    step = 00
-- estado \"013\";
                                                        
                    estado_atual                            <=
                PASSO_0;
                                                        
            else
-- estado \"014\";
                                                        
                    estado_atual                            <=
                ERRO;
                                                        
            end if;
                                                        end if;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(sig_dreq_out_sm
    /=ALL_0)␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(
    sig_dcfg_out_sm(0)='1'␣AND␣sig_dreq_out_sm(0)='1')␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    TRANSACAO_SOLICITADA;
";
for ($i=1;$i<$nro_canal;$i++){
$conteudo.="
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(
    sig_dcfg_out_sm($i)='1'␣AND␣sig_dreq_out_sm($i)='1')␣then
--␣estado␣\"015\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    TRANSACAO_SOLICITADA;
";
}
$conteudo.="
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
--␣estado␣\"016\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ESPERA;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
--␣estado␣\"017\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣ESPERA;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣PASSO_0␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_in_data_out_sm
    (1)='1')␣then␣--en_decoder␣=␣1
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(
    sig_in_data_out_sm(4␣downto␣3)=\"01\")␣then␣--␣step␣=␣01
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(
    sig_in_data_out_sm(09␣downto␣05)>MAX_CHANNEL)␣then␣--␣
    verifica␣existencia␣do␣canal
--␣estado␣\"021\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣ERRO;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else␣--
    ␣canal␣existe
--␣estado␣\"022\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣PASSO_1;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(
    sig_in_data_out_sm(4␣downto␣3)=\"10\")␣then␣--␣step␣=␣10
--␣estado␣\"024\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣ERRO
    ;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(
    sig_in_data_out_sm(4␣downto␣3)=\"00\")␣then␣--␣step␣=␣00
--␣estado␣\"025\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    PASSO_0;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
--␣estado␣\"026\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣ERRO
    ;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else␣--␣en_decoder␣=␣0
--␣estado␣\"027\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣ERRO;
```

```
                                        end if;
                                when PASSO_1 =>
                                        if (sig_in_data_out_sm
    (1)='1') then --en_decoder = 1
                                                if (
    sig_in_data_out_sm(4 downto 3)="10") then -- step = 10
                                                        if (
    sig_in_data_out_sm(09 downto 05)>MAX_CHANNEL) then --
    verifica existencia do canal
-- estado "031";

        estado_atual                            <=
        ERRO;
                                                        else

        if (sig_in_data_out_sm(09 downto 05)=
    sig_ch_session_out_sm) then
-- estado "032";

                estado_atual                            <=
                PASSO_2;

        else
-- estado "033";

                estado_atual                            <=
    ERRO;

        end if;
                                                        end if;
                                                elsif (
    sig_in_data_out_sm(4 downto 3)="00") then -- step = 00
-- estado "034";

    estado_atual                            <=              ERRO
    ;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
--␣estado␣\"035\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    PASSO_1;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else␣--␣en_decoder␣=␣0
--␣estado␣\"036\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣ERRO;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣PASSO_2␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_in_data_out_sm
    (1)='1')␣then␣--en_decoder␣=␣1
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(
    sig_in_data_out_sm(2)='1')␣then␣--␣r_status_de␣=␣'1'
--␣estado␣\"041\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    STATUS;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(
    sig_in_data_out_sm(4␣downto␣3)=\"00\")␣then␣--␣step␣=␣00
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(
    sig_in_data_out_sm(09␣downto␣05)>MAX_CHANNEL)␣then␣--␣
    verifica␣existencia␣do␣canal
--␣estado␣\"042\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣ERRO;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else␣--
    ␣canal␣existe
--␣estado␣\"043\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣PASSO_0;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(
    sig_in_data_out_sm(4␣downto␣3)=\"01\")␣then␣−−␣step␣=␣01
−−␣estado␣\"044\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣ERRO
    ;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(
    sig_in_data_out_sm(09␣downto␣05)=sig_ch_session_out_sm)␣then
    ␣−−␣verifica␣se␣sao␣os␣mesmos␣canais
−−␣estado␣\"045\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣PASSO_2;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else␣−−
    ␣sao␣canais␣diferentes
−−␣estado␣\"046\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣ERRO;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
−−␣estado␣\"047\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣ESPERA;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣ERRO␣=>
−−␣estado␣\"051\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣ESPERA;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣STATUS␣=>
−−␣estado␣\"061\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_in_data_out_sm
    (1)='1'␣and␣sig_in_data_out_sm(2)='1')␣then␣−−␣en_decoder␣=␣
```

```
    1␣E␣r_status_de␣=␣'1'
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣STATUS;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣ESPERA;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣TRANSACAO_SOLICITADA␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣case␣(
    sig_ch_session_out_sm)␣is

";
$lista=lista_binario_dado_inteiro($nro_canal);
for ($i=0;$i<$nro_canal;$i++){
$conteudo.="
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"$lista[
    $i]\"=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(
    sig_dreq_out_sm($i)='1')␣then␣--␣garante␣que␣se␣o␣canal␣$i␣
    foi␣selecionado␣o␣dreq␣esta␣ativo
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣if␣(sig_out_data_out_sm(0)='1')␣then␣--␣hold␣='1'
--␣estado␣\"071\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣BARRAMENTO_SOLICITADO;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣else
--␣estado␣\"072\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣TRANSACAO_SOLICITADA;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
--␣estado␣\"073\";
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=
    ␣␣␣␣␣␣␣␣␣␣␣␣␣TRANSACAO_SOLICITADA;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
";
}
$conteudo.="
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣others=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣
    ␣␣␣␣␣␣␣TRANSACAO_SOLICITADA;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣case;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣BARRAMENTO_SOLICITADO␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_in_data_out_sm
    (0)='1')␣then␣−−␣hlda␣='1'
−−␣estado␣\"081\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣
    BARRAMENTO_GARANTIDO;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
−−␣estado␣\"082\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣
    BARRAMENTO_SOLICITADO;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣BARRAMENTO_GARANTIDO␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_dack_out_sm/=
    ALL_0)␣then
−−␣estado␣\"091\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣
    TRANSACAO_CONFIRMADA;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
−−␣estado␣\"092\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣
```

```
BARRAMENTO_GARANTIDO;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣TRANSACAO_CONFIRMADA␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_in_data_out_sm
    (13␣downto␣12)=\"00\")␣then␣−−␣memoria␣−␣memoria
−−␣estado␣\"0A1\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣OP_LEITURA_A;␣−−
    ␣leitura␣na␣memoria
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(
    sig_in_data_out_sm(13␣downto␣12)=\"01\")␣then␣−−␣memoria␣−␣
    dispositivo
−−␣estado␣\"0A2\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣OP_LEITURA_A;␣−−
    ␣leitura␣na␣memoria
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(
    sig_in_data_out_sm(13␣downto␣12)=\"10\")␣then␣−−␣dispositivo
    ␣−␣memoria
−−␣estado␣\"0A3\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣OP_LEITURA_B;␣−−
    ␣leitura␣no␣dispositivo
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else␣−−␣dispositivo␣−␣
    dispositivo
−−␣estado␣\"0A4\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣OP_LEITURA_B;␣−−
    ␣leitura␣no␣dispositivo
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣OP_LEITURA_A␣=>
−−␣estado␣\"0B1\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣OP_CICLO_LEITURA
    ;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣OP_LEITURA_B␣=>
```

```
-- estado \"0C1\";
                                                    estado_atual
                                <=              OP_CICLO_LEITURA
    ;
                                        when OP_CICLO_LEITURA
    =>
                                    if (sig_in_data_out_sm
    (15 downto 14)=\"10\") then
                                        if (ready='1')
    then
-- estado \"0D1\";
                                        estado_atual                <=
    VERIFICA_MAIS_LEITURA;
                                        else
-- estado \"0D2\";
                                        estado_atual                <=
    OP_CICLO_LEITURA;
                                        end if;
                                    else
-- estado \"0D3\";
                                        estado_atual                <=
    VERIFICA_MAIS_LEITURA;
                                    end if;
                                when VERIFICA_MAIS_LEITURA =>
                                    if (sig_in_data_out_sm
    (15 downto 14)=\"01\") then -- bits (15,14) - flags ->tipo
    transacao 01=bloco
                                        if (
    sig_in_data_out_sm(27)='0') then -- offset is full='0' |
    buffer nao esta cheio ainda
                                            if (
    sig_in_data_out_sm(26)='0') then -- counter_is_empty='0'|
    counter nao eh zero
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣if␣((sig_dreq_out_sm␣AND␣sig_dack_out_sm)/=ALL_0)␣
    then␣−−␣dreq␣continua␣ativo␣(dack␣contem␣o␣canal␣atual␣em␣
    degray)
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_in_data_out_sm(13␣downto␣12)=\"00\")
    ␣then−−␣memoria␣−␣memoria␣−−␣bits␣(13,12)␣−␣flags−>
    modo_operacao
−−␣estado␣\"E01\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣OP_LEITURA_A;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(sig_in_data_out_sm(13␣downto␣12)
    =\"01\")␣then−−␣memoria␣−␣dispositivo␣−−dispositivo
−−␣estado␣\"E02\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣OP_LEITURA_A;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(sig_in_data_out_sm(13␣downto␣12)
    =\"10\")␣then−−␣dispositivo␣−␣memoria
−−␣estado␣\"E03\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣OP_LEITURA_B;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else␣−−␣dispositivo␣−␣dispositivo
−−␣estado␣\"E04\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣OP_LEITURA_B;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣else−−dreq␣nao␣ativo
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_in_data_out_sm(13␣downto␣12)=\"00\")
    ␣then␣−−␣memoria␣para␣memoria
−−␣estado␣\"E05\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣OP_ESCRITA_A;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(sig_in_data_out_sm(13␣downto␣12)
    =\"01\")␣then␣−−␣bits␣(13,12)␣−␣flags−>modo_operacao␣−−␣
    memoria␣−␣dispositivo
−−␣estado␣\"E06\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣OP_ESCRITA_B;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(sig_in_data_out_sm(13␣downto␣12)
    =\"10\")␣then␣−−␣bits␣(13,12)␣−␣flags−>modo_operacao␣−−␣
    dispositivo␣−␣memria
−−␣estado␣\"E07\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣OP_ESCRITA_A;␣−−␣dispositivo
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else␣−−␣dispositivo␣−␣dispositivo
−−␣estado␣\"E08\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣OP_ESCRITA_B;␣−−␣dispositivo
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else−−␣
    counter␣zerado
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_in_data_out_sm(13␣downto␣12)=\"00\")
    ␣then␣--␣memoria␣para␣memoria
--␣estado␣\"E09\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣OP_ESCRITA_A;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(sig_in_data_out_sm(13␣downto␣12)
    =\"01\")␣then␣--␣bits␣(13,12)␣-␣flags->modo_operacao␣--␣
    memoria
--␣estado␣\"E0A\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣OP_ESCRITA_B;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣elsif␣(sig_in_data_out_sm(13␣downto␣12)=\"10\")␣then
    ␣--␣dispositivo␣-␣memoria
--␣estado␣\"E0B\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣OP_ESCRITA_A;␣--␣dispositivo
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣else␣--␣dispositivo␣-␣dispositivo
--␣estado␣\"E0C\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣OP_ESCRITA_B;␣--␣dispositivo
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else--␣buffer␣
    cheio
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(
    sig_in_data_out_sm(13␣downto␣12)=\"00\")␣then␣--␣memoria␣
    para␣memoria
```

```
-- estado \"E0D\";
                                                                                
        estado_atual                                <=
        OP_ESCRITA_A;
                                                                                elsif (
    sig_in_data_out_sm(13 downto 12)=\"01\") then -- bits 
    (13,12) - flags ->modo_operacao -- memoria
-- estado \"E0E\";
                                                                                
        estado_atual                                <=
        OP_ESCRITA_B;
                                                                                elsif (
    sig_in_data_out_sm(13 downto 12)=\"10\") then -- dispositivo
    - memoria
-- estado \"E0F\";
                                                                                
        estado_atual                                <=
        OP_ESCRITA_A; -- dispositivo
                                                                                else --
    dispositivo - dispositivo
-- estado \"E10\";
                                                                                
        estado_atual                                <=
        OP_ESCRITA_B; -- dispositivo
                                                                                end if;
                                                                                end if;
                                                                                else -- demanda/simples
                                                                                if (
    sig_in_data_out_sm(13 downto 12)=\"00\") then -- memoria 
    para memoria
-- estado \"E11\";
                                                                                
    estado_atual                                <=
    OP_ESCRITA_A;
                                                                                elsif (
    sig_in_data_out_sm(13 downto 12)=\"01\") then -- bits 
```

```
    (13,12)␣–␣flags –>modo_operacao␣––␣memoria
––␣estado␣\"E12\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣
    OP_ESCRITA_B;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(
    sig_in_data_out_sm(13␣downto␣12)=\"10\")␣then␣––dispositivo–
    memoria␣––␣escrita␣no␣disp0sitivo
––␣estado␣\"E13\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣
    OP_ESCRITA_A;␣––␣dispositivo
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else␣––␣
    dispositivo␣/␣dispositivo
––␣estado␣\"E14\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣
    OP_ESCRITA_B;␣––␣dispositivo

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣OP_ESCRITA_A␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if(sig_in_data_out_sm
    (28)='1')then––␣offset_is_empty
––␣estado␣\"0F1\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣
    FAZER_MAIS_ESCRITA;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(
    sig_in_data_out_sm(13␣downto␣12)=\"00\")␣then␣––
––␣estado␣\"0F2\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣
    OP_CICLO_ESCRITA;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
--␣estado␣\"0F3\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣
    OP_CICLO_ESCRITA;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣OP_ESCRITA_B␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if(sig_in_data_out_sm
    (28)='1')then--␣offset_is_empty
--␣estado␣\"101\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣
    FAZER_MAIS_ESCRITA;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(
    sig_in_data_out_sm(13␣downto␣12)=\"10\")␣then␣--␣dispositivo
    ␣-␣memoria
--␣estado␣\"102\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣OP_CICLO_ESCRITA
    ;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else␣--␣memoria
    ␣-␣memoria
--␣estado␣\"103\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣
    OP_CICLO_ESCRITA;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣OP_CICLO_ESCRITA␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_in_data_out_sm
    (15␣downto␣14)=\"10\")␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(ready='1')␣
    then
--␣estado␣\"111\";
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣
    FAZER_MAIS_ESCRITA;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
--␣estado␣\"112\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣
    OP_CICLO_ESCRITA;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
--␣estado␣\"113\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣
    FAZER_MAIS_ESCRITA;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣FAZER_MAIS_ESCRITA␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_in_data_out_sm
    (15␣downto␣14)=\"01\")␣then␣--␣bits␣(15,14)␣-␣flags->tipo␣
    transacao␣01=bloco
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if(
    sig_in_data_out_sm(28)='1')then--␣offset_is_empty
--␣estado␣\"121\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣
    TESTA_CONTADOR;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else␣--␣offset␣
    nao␣esta␣vazio
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(
    sig_in_data_out_sm(13␣downto␣12)=\"00\")␣then␣--␣memoria␣
    para␣memoria
--␣estado␣\"122\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣OP_ESCRITA_A;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(
    sig_in_data_out_sm(13␣downto␣12)=\"01\")␣then␣--␣bits␣
    (13,12)␣-␣flags->modo_operacao␣--␣memoria␣-␣dispositivo
--␣estado␣\"123\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣OP_ESCRITA_B;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(
    sig_in_data_out_sm(13␣downto␣12)=\"10\")␣then␣--dispositivo␣
    -␣memoria
--␣estado␣\"124\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣OP_ESCRITA_A;␣--␣dispositivo
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else␣--
    ␣dispositivo␣-␣dispositivo
--␣estado␣\"125\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣OP_ESCRITA_B;␣--␣dispositivo
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else␣--␣simples/demanda




--␣estado␣\"125\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣
    TESTA_CONTADOR;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣TESTA_CONTADOR␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_in_data_out_sm
    (26)='0')␣then␣--␣counter_is_empty_mo␣--␣0␣nao␣esta␣vazio
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(
    sig_in_data_out_sm(15␣downto␣14)=\"01\")␣then␣−−␣bits␣
    (15,14)␣−␣flags−>tipo␣transacao␣01=bloco
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(
    sig_in_data_out_sm(13␣downto␣12)=\"00\")␣then␣−−␣bits␣
    (13,12)␣−␣flags−>modo_operacao␣−−␣memoria␣−␣memoria
−−␣estado␣\"131\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣OP_LEITURA_A;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(
    sig_in_data_out_sm(13␣downto␣12)=\"01\")␣then␣−−memoria␣−␣
    dispositivo
−−␣estado␣\"132\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣OP_LEITURA_A;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(
    sig_in_data_out_sm(13␣downto␣12)=\"10\")␣then␣−−dispositivo␣
    −␣memoria
−−␣estado␣\"133\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣OP_LEITURA_B;␣−−␣dispositivo
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else−−␣
    dispositivo␣−␣dispositivo
−−␣estado␣\"134\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣OP_LEITURA_B;␣−−␣dispositivo
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
−−␣estado␣\"135\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    estado_atual␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    TRANSACAO_SOLICITADA;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else␣−−␣esta␣vazio
−−␣estado␣\"136\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual
   ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣FINALIZACAO;␣−−␣
   dispositivo
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣FINALIZACAO␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_in_data_out_sm
   (0)='0')␣then␣−−␣finaliza␣quando␣hlda␣descer
−−␣estado␣\"141\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual
   ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣ESPERA;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else␣−−␣hlda␣em␣1␣ainda



−−␣estado␣\"142\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual
   ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣␣␣␣␣␣␣␣FINALIZACAO;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣case;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣end␣process;

−−␣logica␣de␣saida␣de␣sinais␣por␣estado
␣␣␣␣␣␣␣␣process␣(
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣clk,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣reset,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣estado_atual,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_in_data_out_sm,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_out_sm,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_dreq_out_sm,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_ch_session_out_sm
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣)
```

```
␣␣␣␣␣␣␣␣begin
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(reset='1')␣then

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_dcfg_sm␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=
    ␣␣␣␣␣␣'0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_status_sm␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=
    ␣␣␣␣␣␣'0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_ch_sm␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=
    ␣␣␣␣␣␣'0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_dack_sm␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=
    ␣␣␣␣␣␣'0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_status_db_f_sm␣␣<='0';

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_dcfg_in_sm␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=
    ␣␣␣␣␣␣ALL_0;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=
    ␣␣␣␣␣␣(others=>'0');
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_ch_session_in_sm␣␣␣␣<=␣␣␣␣␣␣(others
    =>'0');
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_dack_in_sm␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=
    ␣␣␣␣␣␣(others=>'0');
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_status_in_sm␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=
    ␣␣␣␣␣␣(others=>'0');
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(clk='1'␣and␣clk'EVENT)␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣case(estado_atual)␣is
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣␣␣␣ESPERA␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_ch_sm␣␣<='1';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_in_data_out_sm
    (1)='1')␣then␣--en_decoder␣=␣1
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(7␣downto␣0)␣␣<=\"01000000\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(12␣downto␣8)␣<=␣␣␣␣␣␣sig_in_data_out_sm
    (09␣downto␣05);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(31␣downto␣13)␣␣␣␣␣␣␣␣
```

```
<=\"00000000000000000001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_ch_session_in_sm␣␣<=sig_in_data_in_sm(09␣downto␣05);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(sig_dreq_out_sm
    /=ALL_0)␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(
    sig_dreq_out_sm(0)='1'␣AND␣sig_dcfg_out_sm(0)='1')␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_ch_session_in_sm␣␣␣␣␣␣␣␣␣␣␣␣<=\"".
    int_para_binario(0)."\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_out_data_in_sm(7␣downto␣0)␣␣<=\"00000000\";␣␣--␣
    seleciona␣canal␣$0
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_out_data_in_sm(12␣downto␣8)␣<=␣␣␣␣␣␣\"".
    int_para_binario(0)."\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_out_data_in_sm(31␣downto␣13)␣␣␣␣␣␣␣␣
    <=\"1000000001111111101\";



";
for($i=1;$i<$nro_canal;$i++){
$conteudo.="
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(
    sig_dreq_out_sm($i)='1'␣AND␣sig_dcfg_out_sm($i)='1')␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_ch_session_in_sm␣␣␣␣␣␣␣␣␣␣␣␣<=\"".
    int_para_binario($i)."\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_out_data_in_sm(7␣downto␣0)␣␣<=\"00000000\";␣␣--␣
    seleciona␣canal␣$i
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_out_data_in_sm(12␣downto␣8)␣<=␣␣␣␣␣␣\"".
    int_para_binario($i)."\";
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_out_data_in_sm(31␣downto␣13)␣␣␣␣␣␣␣␣
    <=\"1000000001111111101\";
";
}
$conteudo.="
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_status_in_sm(31␣downto␣30)<=ST_EM_PROCESSO;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_status_in_sm(29␣downto␣05)<=ST_TRANSACAO_NAO_CONFIGURADA
    ;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_status_in_sm(04␣downto␣00)<=sig_ch_session_out_sm;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm␣<=x\"00002040\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_en_w_status_db_f_sm␣␣␣␣␣␣␣␣␣<='0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm␣<=x\"00002040\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_en_w_status_db_f_sm␣␣␣␣␣␣␣␣␣<='0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣␣␣␣PASSO_0␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--sig_out_data_in_sm␣<=
    x\"00002040\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm<=
    sig_out_data_out_sm;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_status_sm
    ␣␣␣␣␣␣<='1';␣--␣ativa␣escrita␣no␣registrador␣status
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_ch_sm␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣'0';
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_in_data_out_sm
    (1)='1')␣then␣−−en_decoder␣=␣1
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(
    sig_in_data_out_sm(4␣downto␣3)=\"01\")␣then␣−−␣step␣=␣01
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(
    sig_in_data_out_sm(09␣downto␣05)>MAX_CHANNEL)␣then␣−−␣
    verifica␣existencia␣do␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣−−␣nao␣existe
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣sig_status_in_sm(31␣downto␣30)<=ST_ERRO;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣sig_status_in_sm(29␣downto␣05)<=ST_CANAL_INVALIDO;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣sig_status_in_sm(04␣downto␣00)<=
    sig_ch_session_out_sm;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣sig_en_w_ch_sm␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=
    ␣␣␣␣␣␣␣'0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else␣−−
    ␣canal␣existe
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_status_in_sm(31␣downto␣30)<=
    ST_EM_PROCESSO;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_status_in_sm(29␣downto␣05)<=
    ST_CONFIGURANDO_CANAL_PASSO_1;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_status_in_sm(04␣downto␣00)<=
    sig_ch_session_out_sm;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_ch_sm␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣␣'0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
```

```
                elsif (
    sig_in_data_out_sm(4 downto 3)="10") then -- step = 10
                    sig_status_in_sm(31 downto 30)<=ST_ERRO;
                    sig_status_in_sm(29 downto 05)<=ST_CANAL_PASSO_INVALIDO;
                    sig_status_in_sm(04 downto 00)<=sig_ch_session_out_sm;

                    sig_en_w_ch_sm                                  <=      '0';
                elsif (
    sig_in_data_out_sm(4 downto 3)="00") then -- step = 00
                    sig_status_in_sm(31 downto 30)<=ST_EM_PROCESSO;
                    sig_status_in_sm(29 downto 05)<=
    ST_CONFIGURANDO_CANAL_PASSO_0;
                    sig_status_in_sm(04 downto 00)<=sig_ch_session_out_sm;

                    sig_en_w_ch_sm                                  <=      '1';
                    sig_ch_session_in_sm                    <=      
    sig_in_data_out_sm(09 downto 05);
                else
                    sig_status_in_sm(31 downto 30)<=ST_ERRO;
                    sig_status_in_sm(29 downto 05)<=ST_CANAL_PASSO_INVALIDO;
                    sig_status_in_sm(04 downto 00)<=sig_ch_session_out_sm;

                    sig_en_w_ch_sm                                  <=      '0';
```

```
                                            end if;

                                    else -- en_decoder = 0
                                            sig_status_in_sm(31 downto 30)<=ST_ERRO;
                                            sig_status_in_sm(29 downto 05)<=ST_CONFIGURACAO_ABANDONADA;
                                            sig_status_in_sm(04 downto 00)<=sig_ch_session_out_sm;

                                            sig_en_w_ch_sm                              <=      '0';
                                    end if;


                            when    PASSO_1 =>
                                    sig_en_w_status_sm      <='1'; -- ativa escrita no registrador status
                                    if (sig_in_data_out_sm(1)='1') then --en_decoder = 1
                                            if (sig_in_data_out_sm(09 downto 05)=sig_ch_session_out_sm) then -- verifica se eh o mesmo canal
                                                    if (sig_in_data_out_sm(4 downto 3)=\"10\") then -- step = 10
                                                            if (sig_in_data_out_sm(09 downto 05)>MAX_CHANNEL) then -- verifica existencia do canal
                                                                    sig_status_in_sm(31 downto 30)<=ST_ERRO;
                                                                    sig_status_in_sm(29 downto 05)<=ST_CANAL_INVALIDO;
                                                                    sig_status_in_sm(04 downto 00)<=
```

```
    sig_ch_session_out_sm;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(31␣downto␣13)␣
    <=\"0000000000000000001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(12␣downto␣08)␣<=
    sig_ch_session_out_sm;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(07␣downto␣00)␣
    <=\"11000000\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_status_in_sm(31␣downto␣30)<=
    ST_EM_PROCESSO;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_status_in_sm(29␣downto␣05)<=
    ST_CONFIGURANDO_CANAL_PASSO_2;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_status_in_sm(04␣downto␣00)<=
    sig_ch_session_out_sm;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣−−␣escrita␣no␣registrador␣de␣configuracao
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_dcfg_sm␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    <='1';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣case␣(sig_ch_session_out_sm)␣is
";
for ($i=0;$i<$nro_canal;$i++){
        $conteudo.="
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"".int_para_binario($i)."\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_dcfg_in_sm␣␣<=␣
    sig_dcfg_out_sm␣OR␣\"".int_para_degray($i,$nro_canal)."\";
```

```
␣␣␣␣␣␣␣␣";
}
$conteudo.="
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣others=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_dcfg_in_sm␣␣<=
    sig_dcfg_out_sm;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣case;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(
    sig_in_data_out_sm(4␣downto␣3)=\"00\")␣then␣--␣step␣=␣00
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_status_in_sm(31␣downto␣30)<=ST_ERRO;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_status_in_sm(29␣downto␣05)<=
    ST_CANAL_PASSO_INVALIDO;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_status_in_sm(04␣downto␣00)<=
    sig_ch_session_out_sm;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_status_in_sm(31␣downto␣30)<=ST_EM_PROCESSO;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_status_in_sm(29␣downto␣05)<=
    ST_CONFIGURANDO_CANAL_PASSO_1;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_status_in_sm(04␣downto␣00)<=
    sig_ch_session_out_sm;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else␣--␣canais␣
    diferentes
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_status_in_sm(31␣downto␣30)<=ST_ERRO;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_status_in_sm(29␣downto␣05)<=ST_CANAL_DIFERENTE_PASSO_0
    ;———————————————————————————————————————————————
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_status_in_sm(04␣downto␣00)<=sig_ch_session_out_sm;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else␣——␣en_decoder␣=␣0
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_status_in_sm(31␣downto␣30)<=ST_ERRO;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_status_in_sm(29␣downto␣05)<=ST_CONFIGURACAO_ABANDONADA;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_status_in_sm(04␣downto␣00)<=sig_ch_session_out_sm;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣␣␣␣PASSO_2␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_dcfg_sm
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_status_sm
    ␣␣␣␣␣␣<='1';␣——␣ativa␣escrita␣no␣registrador␣status
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(
    sig_in_data_out_sm(1)='1')␣then␣——en_decoder␣=␣1
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(
    sig_in_data_out_sm(2)='1')␣then␣——␣r_status_de␣=␣'1'
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_out_data_in_sm␣<=x\"00002040\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_en_w_status_db_f_sm␣␣␣␣␣␣␣␣<='1';␣——␣habilita␣a
    ␣escrita␣do␣conteudo␣do␣registrador␣status␣no␣barramento␣de␣
    dados␣superior
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(
    sig_in_data_out_sm(4␣downto␣3)=\"00\")␣then␣——␣step␣=␣00
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_out_data_in_sm␣<=x\"00002040\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣if␣(sig_in_data_out_sm(09␣downto␣05)>MAX_CHANNEL)␣
```

```
then -- verifica existencia do canal
                                                        
                sig_status_in_sm(31 downto 30)<=ST_ERRO;
                                                        
                sig_status_in_sm(29 downto 05)<=
ST_CANAL_INVALIDO;
                                                        
                sig_status_in_sm(04 downto 00)<=
sig_ch_session_out_sm;
                                                        
        else -- canal existe
                                                        
                sig_status_in_sm(31 downto 30)<=
ST_EM_PROCESSO;
                                                        
                sig_status_in_sm(29 downto 05)<=
ST_CONFIGURANDO_CANAL_PASSO_0;
                                                        
                sig_status_in_sm(04 downto 00)<=
sig_in_data_out_sm(09 downto 05);


                                                        
                sig_en_w_ch_sm                          
        <=      '1';
                                                        
                sig_ch_session_in_sm                    <=
      sig_in_data_out_sm(09 downto 05);
                                                        
        end if;
                                                        elsif (
sig_in_data_out_sm(4 downto 3)=\"01\") then -- step = 01
                                                        
        sig_out_data_in_sm <=x\"00002040\";
                                                        
        sig_status_in_sm(31 downto 30)<=ST_ERRO;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_status_in_sm(29␣downto␣05)<=
    ST_CANAL_PASSO_INVALIDO;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_status_in_sm(04␣downto␣00)<=sig_in_data_out_sm
    (09␣downto␣05);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣if␣(sig_in_data_out_sm(09␣downto␣05)=
    sig_ch_session_out_sm)␣then␣--␣verifica␣se␣sao␣os␣mesmos␣
    canais

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(31␣downto␣13)␣
    <=\"0000000000000000001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(12␣downto␣08)␣<=
    sig_ch_session_out_sm;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(07␣downto␣00)␣
    <=\"11000000\";

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_status_in_sm(31␣downto␣30)<=
    ST_EM_PROCESSO;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_status_in_sm(29␣downto␣05)<=
    ST_CONFIGURANDO_CANAL_PASSO_2;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_status_in_sm(04␣downto␣00)<=
    sig_ch_session_out_sm;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣else␣--␣sao␣canais␣diferentes
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm␣<=x\"00002040\";
```

```
                                                              
                  sig_status_in_sm(31␣downto␣30)<=ST_ERRO;
                                                              
                  sig_status_in_sm(29␣downto␣05)<=
    ST_CANAL_DIFERENTE_PASSO_0
    ;—————————————————————————————————————————
                                                              
                  sig_status_in_sm(04␣downto␣00)<=
    sig_ch_session_out_sm;
                                                              
          end␣if;
                                                              end␣if;
                                                     else
                                                              
    sig_out_data_in_sm␣<=x\"00002040\";
                                                              
    sig_status_in_sm(31␣downto␣30)<=ST_OK;
                                                              
    sig_status_in_sm(29␣downto␣05)<=ST_CONFIGURACAO_COMPLETA;
                                                              
    sig_status_in_sm(04␣downto␣00)<=sig_ch_session_out_sm;
                                                     end␣if;

                                    when␣␣␣␣ERRO␣=>         
                            ——␣somente␣ativo␣em␣modo␣debug
                                            sig_en_w_status_db_f_sm
    ␣<='1';
                                            sig_en_w_ch_sm␣␣
                                    <=␣␣␣␣␣␣'0';
                                            sig_out_data_in_sm␣<=x
    \"00002040\";
                                    when␣STATUS␣=>
                                            sig_out_data_in_sm␣<=x
    \"00002040\";
                                            if␣(sig_in_data_out_sm
    (1)='1'␣and␣sig_in_data_out_sm(2)='1')␣then␣——␣en_decoder␣=␣
```

```
    1 E r_status_de = '1'
                                                    sig_en_w_status_db_f_sm <='1';
                                                    else
                                                    sig_en_w_status_db_f_sm <='0';
                                                    end if;
                                                    when TRANSACAO_SOLICITADA =>
                                                    sig_en_w_ch_sm
                                                    <=      '0';
                                                    sig_en_w_dack_sm
                                                    <= '0';
                                                    sig_out_data_in_sm(7 
    downto 0)  <=\"00000001\"; -- levanta hold
                                                    sig_out_data_in_sm(12 
    downto 8) <=      sig_out_data_out_sm(12 downto 8);--mantem 
    o canal
                                                    sig_out_data_in_sm(31 
    downto 13)        <=\"1000000000000000000\";
                                                    when BARRAMENTO_SOLICITADO =>
                                                    --081
                                                    sig_out_data_in_sm(7 
    downto 0)  <=\"00000001\";
                                                    sig_out_data_in_sm(12 
    downto 8) <=      sig_out_data_out_sm(12 downto 8);--mantem 
    o canal
                                                    sig_out_data_in_sm(31 
    downto 13)        <=\"1000000000000000000\";
                                                    -- faz nada, aguarda o 
    hlda
                                                    when BARRAMENTO_GARANTIDO =>
                                                    --091
                                                    sig_out_data_in_sm(7 
    downto 0)  <=\"00000001\";
                                                    sig_out_data_in_sm(12 
    downto 8) <=      sig_out_data_out_sm(12 downto 8);--mantem 
```

```
    o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(31␣
    downto␣13)␣␣␣␣␣␣␣␣<=\"1000000000000000000\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--sig_out_data_in_sm(31
    ␣downto␣13)␣␣␣␣␣␣<=\"1000000000001011000\";

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_dack_sm
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣case␣(
    sig_ch_session_out_sm)␣is
";
for ($i=0;$i<$nro_canal;$i++){
$conteudo.="
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"".$lista
    [$i]."\"=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_dack_in_sm␣␣<=\"".binario_para_degray($lista[$i],
    $nro_canal)."\";␣--␣ack␣do␣canal␣$i
";
}
$conteudo.="
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣others=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣null;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣case;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣TRANSACAO_CONFIRMADA␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_dack_sm
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_in_data_out_sm
    (13␣downto␣12)=\"00\")␣then␣--memoria␣-␣memoria␣-␣␣bits␣
    (13,12)␣-␣flags->modo_operacao
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--0A1
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(7␣downto␣0)␣␣<=\"00010001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(12␣downto␣8)␣<=␣␣␣␣␣␣sig_out_data_out_sm
    (12␣downto␣8);--mantem␣o␣canal
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(31␣downto␣13)␣␣␣␣␣␣␣␣
    <=\"1010000010000000000\";␣␣␣␣␣␣−−␣ok
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(
    sig_in_data_out_sm(13␣downto␣12)=\"01\")␣then␣−−␣memoria−
    dispositivo
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣−−0A2
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(7␣downto␣0)␣␣<=\"00010001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(12␣downto␣8)␣<=␣␣␣␣␣␣sig_out_data_out_sm
    (12␣downto␣8);−−mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(31␣downto␣13)␣␣␣␣␣␣␣␣
    <=\"1010000010000000000\";␣␣␣␣␣␣−−ok
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(
    sig_in_data_out_sm(13␣downto␣12)=\"10\")␣then␣−−dispositivo−
    memoria
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣−−0A3
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(7␣downto␣0)␣␣<=\"00000101\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(12␣downto␣8)␣<=␣␣␣␣␣␣sig_out_data_out_sm
    (12␣downto␣8);−−mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(31␣downto␣13)␣␣␣␣␣␣␣␣
    <=\"1100010000000000000\";␣␣␣␣␣␣−−ok
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else␣−−dispositivo␣−␣
    dispositivo
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣−−0A4
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(7␣downto␣0)␣␣<=\"00000101\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(12␣downto␣8)␣<=␣␣␣␣␣␣sig_out_data_out_sm
    (12␣downto␣8);−−mantem␣o␣canal
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
   sig_out_data_in_sm(31␣downto␣13)␣␣␣␣␣␣␣␣
   <=\"1100010000000000000\";␣␣␣␣␣␣--ok
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣OP_LEITURA_A␣=>␣--␣LEITURA
   ␣NA␣MEMORIA
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--0B1
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
   sig_out_data_in_sm(7␣downto␣0)␣␣<=\"00000001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
   sig_out_data_in_sm(12␣downto␣8)␣<=␣␣␣␣␣␣sig_out_data_out_sm
   (12␣downto␣8);--mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
   sig_out_data_in_sm(31␣downto␣13)␣␣␣␣␣␣␣␣
   <=\"1000000000001010000\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣OP_LEITURA_B␣=>--␣LEITURA␣
   NO␣PERIFERICO
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--0C1
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
   sig_out_data_in_sm(7␣downto␣0)␣␣<=\"00000001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
   sig_out_data_in_sm(12␣downto␣8)␣<=␣␣␣␣␣␣sig_out_data_out_sm
   (12␣downto␣8);--mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
   sig_out_data_in_sm(31␣downto␣13)␣␣␣␣␣␣␣␣
   <=\"1000000000100010000\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣OP_CICLO_LEITURA␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_in_data_out_sm
   (15␣downto␣14)=\"10\")␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(ready='1')␣
   then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--0D1
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
   sig_out_data_in_sm(7␣downto␣0)␣␣<=\"00000001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
   sig_out_data_in_sm(12␣downto␣8)␣<=␣␣␣␣␣␣sig_out_data_out_sm
```

```
    (12 downto 8);--mantem o canal
                                                                    
    sig_out_data_in_sm(31 downto 13)        
    <=\"1000000000000000000\";
                                                        else
                                                                    --0D2
                                                                    
    sig_out_data_in_sm(7 downto 0)   <=\"00000001\";
                                                                    
    sig_out_data_in_sm(12 downto 8) <=      sig_out_data_out_sm
    (12 downto 8);--mantem o canal
                                                                    
    sig_out_data_in_sm(31 downto 13)        
    <=\"1000000000000000000\";
                                                        end if;
                                                        else
                                                                    --0D3
                                                                    
    sig_out_data_in_sm(7 downto 0)   <=\"00000001\";
                                                                    
    sig_out_data_in_sm(12 downto 8) <=      sig_out_data_out_sm
    (12 downto 8);--mantem o canal
                                                                    
    sig_out_data_in_sm(31 downto 13)        
    <=\"1000000000000000000\";
                                                        end if;
                                                when VERIFICA_MAIS_LEITURA =>
                                                        if (sig_in_data_out_sm
    (15 downto 14)=\"01\") then -- bits (15,14) - flags ->tipo
    transacao 01=bloco
                                                        if (
    sig_in_data_out_sm(27)='0') then -- offset is full='0' |
    buffer nao esta cheio ainda
                                                        if (
    sig_in_data_out_sm(26)='0') then -- counter_is_empty='0'|
    counter nao eh zero
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣if␣((sig_dreq_out_sm␣AND␣sig_dack_out_sm)/=ALL_0)␣
    then␣--␣dreq␣continua␣ativo␣(dack␣contem␣o␣canal␣atual␣em␣
    degray)
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_in_data_out_sm(13␣downto␣12)=\"00\")
    ␣then␣--␣bits␣(13,12)␣-␣flags->modo_operacao␣-␣memoria␣-␣
    memoria
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--E1
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(7␣downto␣0)␣␣
    <=\"00010001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(12␣downto␣8)␣<=
    ␣␣␣␣␣␣sig_out_data_out_sm(12␣downto␣8);--mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(31␣downto␣13)
    ␣␣␣␣␣␣␣␣<=\"1010000010000001000\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(sig_in_data_out_sm(13␣downto␣12)
    =\"01\")␣then␣--␣memoria␣-␣dispositivo
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--E2
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(7␣downto␣0)␣␣
    <=\"00010001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(12␣downto␣8)␣<=
    ␣␣␣␣␣␣sig_out_data_out_sm(12␣downto␣8);--mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(31␣downto␣13)
    ␣␣␣␣␣␣␣␣<=\"1010000010000001000\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(sig_in_data_out_sm(13␣downto␣12)
    =\"10\")␣then␣--␣dispositivo␣-␣memoria
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣——E3
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(7␣downto␣0)␣␣
    <=\"00000101\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(12␣downto␣8)␣<=
    ␣␣␣␣␣␣sig_out_data_out_sm(12␣downto␣8);−−mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(31␣downto␣13)
    ␣␣␣␣␣␣␣␣<=\"1100010000000001000\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else —dispositivo␣−␣dispositivo
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣——E4
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(7␣downto␣0)␣␣
    <=\"00000101\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(12␣downto␣8)␣<=
    ␣␣␣␣␣␣sig_out_data_out_sm(12␣downto␣8);−−mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(31␣downto␣13)
    ␣␣␣␣␣␣␣␣<=\"1100010000000001000\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣else —dreq␣nao␣ativo
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_in_data_out_sm(13␣downto␣12)=\"00\")
    ␣then␣——␣memoria␣para␣memoria
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣——E5
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(7␣downto␣0)␣␣
    <=\"00100001\";
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(12␣downto␣8)␣<=
    ␣␣␣␣␣␣sig_out_data_out_sm(12␣downto␣8);−−mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(31␣downto␣13)
    ␣␣␣␣␣␣␣␣<=\"1000101000000000010\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(sig_in_data_out_sm(13␣downto␣12)
    =\"01\")␣then␣−−␣bits␣(13,12)␣−␣flags−>modo_operacao␣−−␣
    memoria
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣−−E6
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(7␣downto␣0)␣␣
    <=\"00100001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(12␣downto␣8)␣<=
    ␣␣␣␣␣␣sig_out_data_out_sm(12␣downto␣8);−−mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(31␣downto␣13)
    ␣␣␣␣␣␣␣␣<=\"1001010000000000010\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(sig_in_data_out_sm(13␣downto␣12)
    =\"10\")␣then␣−−␣dispositivo␣−␣memoria
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣−−E7
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(7␣downto␣0)␣␣
    <=\"00001001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(12␣downto␣8)␣<=
    ␣␣␣␣␣␣sig_out_data_out_sm(12␣downto␣8);−−mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(31␣downto␣13)
    ␣␣␣␣␣␣␣␣<=\"1000100010000000010\";
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--E8
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(7␣downto␣0)␣␣
    <=\"00001001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(12␣downto␣8)␣<=
    ␣␣␣␣␣␣sig_out_data_out_sm(12␣downto␣8);--mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(31␣downto␣13)
    ␣␣␣␣␣␣␣␣<=\"1001000100000000010\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else--␣
    counter␣zerado
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣if␣(sig_in_data_out_sm(13␣downto␣12)=\"00\")␣then␣--
    ␣memoria␣para␣memoria
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--E9
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(7␣downto␣0)␣␣
    <=\"00100001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(12␣downto␣8)␣<=␣␣␣␣␣␣
    sig_out_data_out_sm(12␣downto␣8);--mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(31␣downto␣13)␣␣␣␣␣␣␣␣
    <=\"1000101000000001010\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣elsif␣(sig_in_data_out_sm(13␣downto␣12)=\"01\")␣then
    ␣--␣bits␣(13,12)␣-␣flags->modo_operacao␣--␣memoria
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣−−EA
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(7␣downto␣0)␣␣
    <=\"00100001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(12␣downto␣8)␣<=␣␣␣␣␣␣
    sig_out_data_out_sm(12␣downto␣8);−−mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(31␣downto␣13)␣␣␣␣␣␣␣␣
    <=\"1001010000000001010\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣elsif␣(sig_in_data_out_sm(13␣downto␣12)=\"10\")␣then
    ␣−−␣dispositivo␣−␣memoria
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣−−EB
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(7␣downto␣0)␣␣
    <=\"00001001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(12␣downto␣8)␣<=␣␣␣␣␣␣
    sig_out_data_out_sm(12␣downto␣8);−−mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(31␣downto␣13)␣␣␣␣␣␣␣␣
    <=\"1000100010000001010\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣−−EC
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(7␣downto␣0)␣␣
    <=\"00001001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(12␣downto␣8)␣<=␣␣␣␣␣␣
    sig_out_data_out_sm(12␣downto␣8);−−mantem␣o␣canal
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_out_data_in_sm(31␣downto␣13)␣␣␣␣␣␣␣␣
    <=\"1001000100000001010\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else—␣buffer␣
    cheio
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(
    sig_in_data_out_sm(13␣downto␣12)=\"00\")␣then␣−−␣memoria␣
    para␣memoria
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣−−ED
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_out_data_in_sm(7␣downto␣0)␣␣<=\"00100001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_out_data_in_sm(12␣downto␣8)␣<=␣␣␣␣␣␣
    sig_out_data_out_sm(12␣downto␣8);−−mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_out_data_in_sm(31␣downto␣13)␣␣␣␣␣␣␣␣
    <=\"1000101000000001010\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(
    sig_in_data_out_sm(13␣downto␣12)=\"01\")␣then␣−−␣bits␣
    (13,12)␣−␣flags−>modo_operacao␣−−␣memoria
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣−−EE
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_out_data_in_sm(7␣downto␣0)␣␣<=\"00100001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_out_data_in_sm(12␣downto␣8)␣<=␣␣␣␣␣␣
    sig_out_data_out_sm(12␣downto␣8);−−mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_out_data_in_sm(31␣downto␣13)␣␣␣␣␣␣␣␣
    <=\"1001010000000001010\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(
    sig_in_data_out_sm(13␣downto␣12)=\"10\")␣then␣−−␣dispositivo
```

```
    ␣–␣memoria
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣––EF
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_out_data_in_sm(7␣downto␣0)␣␣<=\"00001001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_out_data_in_sm(12␣downto␣8)␣<=␣␣␣␣␣␣
    sig_out_data_out_sm(12␣downto␣8);−−mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_out_data_in_sm(31␣downto␣13)␣␣␣␣␣␣␣␣
    <=\"1000100010000001010\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣––E10
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_out_data_in_sm(7␣downto␣0)␣␣<=\"00001001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_out_data_in_sm(12␣downto␣8)␣<=␣␣␣␣␣␣
    sig_out_data_out_sm(12␣downto␣8);−−mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_out_data_in_sm(31␣downto␣13)␣␣␣␣␣␣␣␣
    <=\"1001000100000001010\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else␣––␣demanda/simples
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(
    sig_in_data_out_sm(13␣downto␣12)=\"00\")␣then␣––␣memoria␣
    para␣memoria
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣––E11
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(7␣downto␣0)␣␣<=\"00100001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(12␣downto␣8)␣<=␣␣␣␣␣␣sig_out_data_out_sm
    (12␣downto␣8);−−mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(31␣downto␣13)␣␣␣␣␣␣␣␣
```

```
    <=\"1000101000000000010\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(
    sig_in_data_out_sm(13␣downto␣12)=\"01\")␣then␣--␣bits␣
    (13,12)␣-␣flags->modo_operacao␣--␣memoria
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--E12
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(7␣downto␣0)␣␣<=\"00100001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(12␣downto␣8)␣<=␣␣␣␣␣␣sig_out_data_out_sm
    (12␣downto␣8);--mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(31␣downto␣13)␣␣␣␣␣␣␣␣
    <=\"1001010000000000010\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(
    sig_in_data_out_sm(13␣downto␣12)=\"10\")␣then␣--␣dispositivo
    ␣-␣memoria
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--E13
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(7␣downto␣0)␣␣<=\"00001001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(12␣downto␣8)␣<=␣␣␣␣␣␣sig_out_data_out_sm
    (12␣downto␣8);--mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(31␣downto␣13)␣␣␣␣␣␣␣␣
    <=\"1000110000000000010\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--E14
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(7␣downto␣0)␣␣<=\"00001001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(12␣downto␣8)␣<=␣␣␣␣␣␣sig_out_data_out_sm
    (12␣downto␣8);--mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(31␣downto␣13)␣␣␣␣␣␣␣␣
    <=\"1001000100000000010\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣OP_ESCRITA_A␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if(sig_in_data_out_sm
    (28)='1')then--␣offset_is_empty
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--0F1
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(7␣downto␣0)␣␣<=\"00000001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(12␣downto␣8)␣<=␣␣␣␣␣␣sig_out_data_out_sm
    (12␣downto␣8);--mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(31␣downto␣13)␣␣␣␣␣␣␣␣
    <=\"1000000000000000010\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(
    sig_in_data_out_sm(13␣downto␣12)=\"00\")␣then␣--␣memoria␣
    para␣memoria
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--0F2
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(7␣downto␣0)␣␣<=\"00000001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(12␣downto␣8)␣<=␣␣␣␣␣␣sig_out_data_out_sm
    (12␣downto␣8);--mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(31␣downto␣13)␣␣␣␣␣␣␣␣
    <=\"1000000000100000010\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--0F3
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(7␣downto␣0)␣␣<=\"00000001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(12␣downto␣8)␣<=␣␣␣␣␣␣sig_out_data_out_sm
    (12␣downto␣8);--mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(31␣downto␣13)␣␣␣␣␣␣␣␣
    <=\"1000000000001000010\";
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣OP_ESCRITA_B␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if(sig_in_data_out_sm
    (28)='1')then—␣offset_is_empty
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--101
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(7␣downto␣0)␣␣<=\"00000001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(12␣downto␣8)␣<=␣␣␣␣␣␣sig_out_data_out_sm
    (12␣downto␣8);--mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(31␣downto␣13)␣␣␣␣␣␣␣␣
    <=\"1000000000000000010\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(
    sig_in_data_out_sm(13␣downto␣12)=\"10\")␣then␣—␣dispositivo
    ␣para␣memoria
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--102
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(7␣downto␣0)␣␣<=\"00000001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(12␣downto␣8)␣<=␣␣␣␣␣␣sig_out_data_out_sm
    (12␣downto␣8);--mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(31␣downto␣13)␣␣␣␣␣␣␣␣
    <=\"1000000000100000010\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--103
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(7␣downto␣0)␣␣<=\"00000001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(12␣downto␣8)␣<=␣␣␣␣␣␣sig_out_data_out_sm
    (12␣downto␣8);--mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(31␣downto␣13)␣␣␣␣␣␣␣␣
```

```
<=\"1000000000100000010\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣OP_CICLO_ESCRITA␣
=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_in_data_out_sm
(15␣downto␣14)=\"10\")␣then␣−−␣demanda
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if(ready='1')␣
then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣−−111
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
sig_out_data_in_sm(7␣downto␣0)␣␣<=\"00000001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
sig_out_data_in_sm(12␣downto␣8)␣<=␣␣␣␣␣␣sig_out_data_out_sm
(12␣downto␣8);−−mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
sig_out_data_in_sm(31␣downto␣13)␣␣␣␣␣␣␣␣
<=\"1000000000000000010\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣−−112
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
sig_out_data_in_sm(7␣downto␣0)␣␣<=\"00000001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
sig_out_data_in_sm(12␣downto␣8)␣<=␣␣␣␣␣␣sig_out_data_out_sm
(12␣downto␣8);−−mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
sig_out_data_in_sm(31␣downto␣13)␣␣␣␣␣␣␣␣
<=\"1000000000000000010\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣−−113
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
sig_out_data_in_sm(7␣downto␣0)␣␣<=\"00000001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
sig_out_data_in_sm(12␣downto␣8)␣<=␣␣␣␣␣␣sig_out_data_out_sm
```

```
    (12␣downto␣8);−−mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(31␣downto␣13)␣␣␣␣␣␣␣␣
    <=\"1000000000000000010\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣FAZER_MAIS_ESCRITA␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_in_data_out_sm(15␣
    downto␣14)=\"01\")␣then␣−−␣bits␣(15,14)␣−␣flags−>tipo␣
    transacao␣01=bloco
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if(sig_in_data_out_sm
    (28)='1')then−−␣offset_is_empty
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣−−121
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(7␣downto␣0)␣␣<=\"00000001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(12␣downto␣8)␣<=␣␣␣␣␣␣sig_out_data_out_sm
    (12␣downto␣8);−−mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(31␣downto␣13)␣␣␣␣␣␣␣␣
    <=\"1000000000000000010\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(
    sig_in_data_out_sm(13␣downto␣12)=\"00\")␣then␣−−␣memoria␣
    para␣memoria
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣−−122
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(7␣downto␣0)␣␣<=\"00100001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(12␣downto␣8)␣<=␣␣␣␣␣␣sig_out_data_out_sm
    (12␣downto␣8);−−mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(31␣downto␣13)␣␣␣␣␣␣␣␣
    <=\"1000101000000001010\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(
    sig_in_data_out_sm(13␣downto␣12)=\"01\")␣then␣−−␣bits␣
    (13,12)␣−␣flags−>modo_operacao␣−−␣memoria
```

```
                                                                --123
                                                                sig_out_data_in_sm(7 downto 0)  <=\"00100001\";
                                                                sig_out_data_in_sm(12 downto 8) <=      sig_out_data_out_sm(12 downto 8);--mantem o canal
                                                                sig_out_data_in_sm(31 downto 13)        <=\"1001010000000001010\";
                                                        elsif (sig_in_data_out_sm(13 downto 12)=\"10\") then -- dispositivo - memoria
                                                                --124
                                                                sig_out_data_in_sm(7 downto 0)  <=\"00001001\";
                                                                sig_out_data_in_sm(12 downto 8) <=      sig_out_data_out_sm(12 downto 8);--mantem o canal
                                                                sig_out_data_in_sm(31 downto 13)        <=\"1000100010000001010\";
                                                        else
                                                                --125
                                                                sig_out_data_in_sm(7 downto 0)  <=\"00001001\";
                                                                sig_out_data_in_sm(12 downto 8) <=      sig_out_data_out_sm(12 downto 8);--mantem o canal
                                                                sig_out_data_in_sm(31 downto 13)        <=\"1001000100000001010\";
                                                        end if;
                                                end if;
                                        else -- simples/demanda
                                                        --126
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(7␣downto␣0)␣␣<=\"00000001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(12␣downto␣8)␣<=␣␣␣␣␣␣sig_out_data_out_sm
    (12␣downto␣8);--mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm(31␣downto␣13)␣␣␣␣␣␣␣␣
    <=\"1000000000000000010\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣TESTA_CONTADOR␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_in_data_out_sm
    (26)='0')␣then␣--␣counter_is_empty_mo␣--␣0␣nao␣esta␣vazio
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(
    sig_in_data_out_sm(15␣downto␣14)=\"01\")␣then␣--␣bits␣
    (15,14)␣-␣flags->tipo␣transacao␣01=bloco
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(
    sig_in_data_out_sm(13␣downto␣12)=\"00\")␣then␣--␣bits␣
    (13,12)␣-␣flags->modo_operacao
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣--131
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_out_data_in_sm(7␣downto␣0)␣␣<=\"00010001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_out_data_in_sm(12␣downto␣8)␣<=␣␣␣␣␣␣
    sig_out_data_out_sm(12␣downto␣8);--mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_out_data_in_sm(31␣downto␣13)␣␣␣␣␣␣␣␣
    <=\"1010000010000001000\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(
    sig_in_data_out_sm(13␣downto␣12)=\"01\")␣then␣--␣memoria-
    dispositivo
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣--132
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_out_data_in_sm(7␣downto␣0)␣␣<=\"00010001\";
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_out_data_in_sm(12␣downto␣8)␣<=␣␣␣␣␣␣
    sig_out_data_out_sm(12␣downto␣8);--mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_out_data_in_sm(31␣downto␣13)␣␣␣␣␣␣␣␣
    <=\"1010000010000001000\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(
    sig_in_data_out_sm(13␣downto␣12)=\"10\")␣then␣--dispositivo␣
    -␣memoria
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣--133
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_out_data_in_sm(7␣downto␣0)␣␣<=\"00000101\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_out_data_in_sm(12␣downto␣8)␣<=␣␣␣␣␣␣
    sig_out_data_out_sm(12␣downto␣8);--mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_out_data_in_sm(31␣downto␣13)␣␣␣␣␣␣␣␣
    <=\"1100010000000001000\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else␣--
    dispositivo␣-␣dispositivo
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣--134
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_out_data_in_sm(7␣downto␣0)␣␣<=\"00000101\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_out_data_in_sm(12␣downto␣8)␣<=␣␣␣␣␣␣
    sig_out_data_out_sm(12␣downto␣8);--mantem␣o␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_out_data_in_sm(31␣downto␣13)␣␣␣␣␣␣␣␣
    <=\"1100010000000001000\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--135
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_en_w_dack_sm␣␣␣␣␣␣␣␣<='1';
```

```
                                                        sig_dack_in_sm          <=ALL_0;

                                                        sig_out_data_in_sm(7 downto 0)  <=\"00000000\";
                                                        sig_out_data_in_sm(12 downto 8) <=      sig_out_data_out_sm(12 downto 8);--mantem o canal
                                                        sig_out_data_in_sm(31 downto 13)        <=\"1000000000000000000\";
        --                                              estado_atual                            <=              TRANSACAO_SOLICITADA;
                                                end if;
                                        else -- esta vazio
                                                --136
                                                sig_en_w_dack_sm        <='1';
                                                sig_dack_in_sm          <=ALL_0;
                                                sig_out_data_in_sm      <=X\"80000002\";
                                        end if;
                                when FINALIZACAO =>
                                        sig_en_w_dack_sm        <='0';

                                        if (sig_in_data_out_sm(0)='0') then -- finaliza quando hlda descer
                                                --141
                                                sig_out_data_in_sm      <=X\"00002040\";
                                        else
                                                --142
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_out_data_in_sm␣␣␣␣␣␣<=X\"80000002\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣case;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣end␣process;

END␣BHV_STATE_MACHINE;
";
if ($grava){
        gravar("state_machine.vhd", $conteudo);
        }
if ($tela){
        tela("state_machine.vhd", $conteudo);
        }
$conteudo_tb="

";
if ($gerar_tb){
        if ($grava){
        gravar("tb_state_machine.vhd", $conteudo_tb);
        }
        if ($tela){
        tela("tb_state_machine.vhd", $conteudo_tb);
        }
        }
}
?>
```

B.0.4 centro_operacao.php (*Datapath*)

```
<?
//include ("funcoes.php");
$gerar_tb=$_GET['gerar_tb'];
$nro_canal=$_GET['nro_canal'];
$grava=$_GET['grava'];
$tela=$_GET['tela'];
```

```
function centro_operacao($gerar_tb, $nro_canal,$grava,$tela){

$conteudo="
——Arquivo␣:␣main_operation.vhd
——————————␣INICIO␣DO␣ARQUIVO————————————————

library␣IEEE;
use␣IEEE.STD_LOGIC_1164.all;
USE␣ieee.std_logic_signed.all;

————————————————————————————————————————————————————————————————————————————————————————


entity␣MAIN_OPERATION␣is
␣␣␣␣port(
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣——␣sinais␣basicos␣da␣entrada␣——
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣clk␣:␣in␣std_logic;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣reset␣:␣in␣std_logic;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣——entradas
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣in_data␣:␣␣␣␣␣␣␣in␣std_logic_vector␣(95
    ␣downto␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣db_f_in␣:␣␣␣␣␣␣␣in␣std_logic_vector␣(31
    ␣downto␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣db_b_in␣:␣␣␣␣␣␣␣in␣std_logic_vector␣(31
    ␣downto␣0);

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣——␣saidas
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣flags_out_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣out␣std_logic_vector␣(15␣downto␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣counter_is_empty␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:
    ␣␣␣␣␣␣␣out␣STD_LOGIC;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣offset_is_full␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣out␣STD_LOGIC;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣offset_is_empty␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣out␣STD_LOGIC;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣addr_f_out␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣out␣std_logic_vector␣(31␣downto␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣addr_b_out␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣out␣std_logic_vector␣(31␣downto␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣db_f_out␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣out␣std_logic_vector␣(31␣
    downto␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣db_b_out␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣out␣std_logic_vector␣(31␣
    downto␣0);


␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣sinais␣provindos␣da␣unidade␣de␣
    controle ---
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_op_mode␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣in␣STD_LOGIC;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_w_flags_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣in␣STD_LOGIC;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_w_offset_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣in␣STD_LOGIC;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_w_counter_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:
    ␣␣␣␣␣␣␣in␣STD_LOGIC;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_counter_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:
    ␣␣␣␣␣␣␣in␣STD_LOGIC;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_w_addr_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣in␣STD_LOGIC;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_addr_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣in␣STD_LOGIC;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_w_addr_b_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣in␣STD_LOGIC;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_addr_b_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣in␣STD_LOGIC;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_w_addr_f_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣in␣STD_LOGIC;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_w_addr_f_b_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣in␣STD_LOGIC;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_w_addr_b_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣in␣STD_LOGIC;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_w_addr_b_b_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣in␣STD_LOGIC;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_w_db_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣in␣STD_LOGIC;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_w_db_b_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣in␣STD_LOGIC;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_r_db_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣in␣STD_LOGIC;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_r_db_b_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣in␣STD_LOGIC
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣);
end␣MAIN_OPERATION;

ARCHITECTURE␣BHV_MAIN_OPERATION␣OF␣MAIN_OPERATION␣IS

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣COMPONENT␣mem␣IS␣PORT␣(
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣clock␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣IN␣
    STD_LOGIC␣␣:=␣'1';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣address␣␣␣␣␣␣␣␣␣:␣IN␣
    std_logic_vector␣(7␣DOWNTO␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣IN␣
    std_logic_vector␣(31␣DOWNTO␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wren␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣IN␣
    STD_LOGIC␣;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣q␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣OUT␣std_logic_vector␣(31␣DOWNTO␣0)
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣END␣COMPONENT;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣COMPONENT␣reg␣is␣Generic␣(␣␣␣␣␣N␣:␣
    Integer␣:=32␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣port(
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣reset,␣clk,␣carga␣:␣IN␣␣␣
    STD_LOGIC;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣D␣:␣␣␣IN␣␣␣␣␣␣std_logic_vector␣
    (N-1␣DOWNTO␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣Q␣:␣␣␣OUT␣␣␣␣␣std_logic_vector␣
    (N-1␣DOWNTO␣0)
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣component;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣----------␣Current␣Operation␣signals␣
    ----------------

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_addr_f_in_co␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣std_logic_vector␣(31␣downto␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_addr_b_in_co␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣std_logic_vector␣(31␣downto␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_counter_in_co␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣std_logic_vector␣(15␣downto␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_offset_in_co␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣std_logic_vector␣(15␣downto␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_data_in_buffer_co␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣std_logic_vector␣(31␣downto␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_en_w_databuffer_co␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣std_logic;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_addr_f_out_co␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣std_logic_vector␣(31␣downto␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_addr_b_out_co␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣std_logic_vector␣(31␣downto␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_counter_out_co␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣std_logic_vector␣(15␣downto␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_offset_out_co␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣std_logic_vector␣(15␣downto␣0);
```

```
                        signal sig_flags_out_co
                        : std_logic_vector (15 downto 0);
                        signal sig_data_out_buffer_co
                        : std_logic_vector (31 downto 0);

                        ---------- sinais de transicao
                        ----------------------

                        signal sig_addr_f_increment
                        : std_logic_vector (31 downto 0);
                        signal sig_addr_b_increment
                        : std_logic_vector (31 downto 0);
                        signal sig_counter_increment
                        : std_logic_vector (15 downto 0);
--                      signal sig_counter_increment2
                        : std_logic_vector (15 downto 0);

      signal sig_word_size      : std_logic_vector (1 downto 0)
      ;
      signal sig_increment_b    : std_logic;

      signal clk_neg    : std_logic;


BEGIN
              --mapeamentos


              addr_f_co       :       reg generic map( N  =>
              32 ) port map(
                        reset ,
                        clk ,
                        en_w_addr_f_co ,
                        sig_addr_f_in_co ,
                        sig_addr_f_out_co
              );
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣addr_b_co␣␣␣␣␣␣␣:␣␣␣␣␣␣␣reg␣generic␣map(␣N␣␣=>␣
    32␣)␣port␣map(
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣reset,␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣clk,␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    en_w_addr_b_co,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_addr_b_in_co,␣␣␣␣␣␣␣
    sig_addr_b_out_co
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣);

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣counter_co␣␣␣␣␣␣:␣␣␣␣␣␣␣reg␣generic␣map(␣N␣␣=>␣
    16␣)␣port␣map(
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣reset,␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣clk,␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    en_w_counter_co,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_counter_in_co,␣␣␣␣␣␣
    sig_counter_out_co
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣);

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣offset_co␣␣␣␣␣␣␣:␣␣␣␣␣␣␣reg␣generic␣map(␣N␣␣=>␣
    16␣)␣port␣map(
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣reset,␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣clk,␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    en_w_offset_co,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_offset_in_co,␣␣␣␣␣␣␣
    sig_offset_out_co
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣);

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣flags_co␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣reg␣generic␣map(␣N␣␣=>␣
    16␣)␣port␣map(
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣reset,␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣clk,␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    en_w_flags_co,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣in_data(15␣downto␣0),␣␣␣␣␣␣␣␣␣␣␣
    sig_flags_out_co
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣);
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣clk_neg␣<=␣not␣clk;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣buffer_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣mem␣port␣map(
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣clk_neg,␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_offset_out_co(7␣downto␣0),␣sig_data_in_buffer_co,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_databuffer_co,␣sig_data_out_buffer_co
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣);


--␣processos␣referente␣a␣FLAGS
␣␣␣␣␣␣␣␣--␣atualiza␣a␣saida␣do␣registrador␣de␣flags␣para␣maquina␣de␣estados
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣flags_out_co␣␣␣␣␣␣␣␣␣␣␣␣<=␣sig_flags_out_co;
--␣processos␣referente␣a␣ADDR_F
␣␣␣␣␣␣␣␣--␣processo␣que␣atualiza␣o␣proximo␣valor␣de␣addr_f
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣process␣(
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_flags_out_co(7),␣--␣sinal␣que␣diz␣se␣incrementa/decrementa␣no␣barramento␣superior␣[sempre␣32␣bits]
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_addr_f_out_co␣␣␣␣--␣sinal␣da␣saida␣do␣registrador␣addr_f
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣)
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣begin
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_flags_out_co(7)='0')␣then␣--␣incrementa
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_addr_f_increment␣␣␣␣<=␣␣sig_addr_f_out_co+X\"00000004\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else␣--␣decrementa
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_addr_f_increment␣␣␣␣<=␣␣sig_addr_f_out_co+X\"FFFFFFFC\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣process;
```

```
        -- processo que atualiza o valor de addr_f
                process (sl_w_addr_f_co, in_data(95 downto 64),
    sig_addr_f_increment, sig_flags_out_co(9), sig_addr_f_out_co)
                begin
                        if (sl_w_addr_f_co='1') then
                                sig_addr_f_in_co        <=
    in_data(95 downto 64);
                        else
                                if (sig_flags_out_co(9)='0')
    then -- nao bloqueia incremento
                                        sig_addr_f_in_co
    <=      sig_addr_f_increment;
                                else
                                sig_addr_f_in_co        <=
    sig_addr_f_out_co;
                                end if;
                        end if;
                end process;
        -- processo que libera escrita de addr_f no barramento
                process (en_w_addr_f_f_co, sig_addr_f_out_co)
                begin
                        if (en_w_addr_f_f_co='1') then
                                addr_f_out      <=
    sig_addr_f_out_co;
                        else
                                addr_f_out      <=      (others
    =>'Z');
                        end if;
                end process;
        -- processo que libera escrita de addr_f no barramento
    inferior (no caso dispositivo-dispositivo)
                process (en_w_addr_f_b_co, sig_addr_f_out_co)
                begin
                        if (en_w_addr_f_b_co='1') then
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣addr_b_out␣␣␣␣␣␣<=␣␣␣␣␣␣
    sig_addr_f_out_co;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣addr_b_out␣␣␣␣␣␣<=␣␣␣␣␣␣(others
    =>'Z');
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣process;

--␣processos␣referente␣a␣ADDR_B
␣␣␣␣sig_word_size␣<=sig_flags_out_co(1␣downto␣0);
␣␣␣␣sig_increment_b␣<=sig_flags_out_co(6);
␣␣␣␣␣␣␣␣--␣processo␣que␣atualiza␣o␣proximo␣valor␣de␣addr_b
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣process␣(
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_increment_b,␣--␣sinal␣que␣diz␣se␣
    incrementa/decrementa␣no␣barramento␣inferior
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_word_size,␣--␣sinal␣que␣diz␣o␣
    tamanho␣de␣palavra␣do␣dispositivo,␣incrementar/decrementar␣
    de␣acordo␣com␣tamanho␣da␣palavra
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_addr_b_out_co,␣␣␣␣--␣sinal␣da␣saida
    ␣do␣registrador␣addr_f
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_addr_b_increment␣--␣sinal␣com␣
    endereco␣atualizado
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣)
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣begin
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_increment_b='0')␣then␣--␣
    incrementa
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣case␣(sig_word_size)␣is
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00\"␣␣␣␣␣=>␣--␣+
    ␣1␣(sistema␣de␣8␣bits)
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_addr_b_increment␣␣␣␣<=␣␣sig_addr_b_out_co+X\"00000001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01\"␣␣␣␣␣=>␣--␣+
    ␣2␣(sistema␣de␣16␣bits)
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_addr_b_increment␣␣␣␣<=␣␣sig_addr_b_out_co+X\"00000002\";
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10\"␣␣␣␣␣=>␣--␣+
    ␣4␣(sistema␣de␣32␣bits)
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_addr_b_increment␣␣␣␣<=␣␣sig_addr_b_out_co+X\"00000004\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11\"␣␣␣␣␣=>␣--␣+
    ␣8␣(sistema␣de␣64␣bits)
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_addr_b_increment␣␣␣␣<=␣␣sig_addr_b_out_co+X\"00000008\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣others=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_addr_b_increment
    ␣␣<=sig_addr_b_increment;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣case;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else␣--␣decrementa
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣case␣(sig_word_size)␣is
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00\"␣␣␣␣␣=>␣--␣-
    ␣1␣(sistema␣de␣8␣bits)
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_addr_b_increment␣␣␣␣<=␣␣sig_addr_b_out_co+X\"FFFFFFFF\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01\"␣␣␣␣␣=>␣--␣-
    ␣2␣(sistema␣de␣16␣bits)
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_addr_b_increment␣␣␣␣<=␣␣sig_addr_b_out_co+X\"FFFFFFFE\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10\"␣␣␣␣␣=>␣--␣-
    ␣4␣(sistema␣de␣32␣bits)
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_addr_b_increment␣␣␣␣<=␣␣sig_addr_b_out_co+X\"FFFFFFFC\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11\"␣␣␣␣␣=>␣--␣-
    ␣8␣(sistema␣de␣64␣bits)
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_addr_b_increment␣␣␣␣<=␣␣sig_addr_b_out_co+X\"FFFFFFF8\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣others=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_addr_b_increment
    ␣␣<=sig_addr_b_increment;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣case;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣process;
```

```
        -- processo que atualiza o valor de addr_b
                process (sl_w_addr_b_co , in_data(63 downto 32), sig_addr_b_increment,sig_flags_out_co(8),sig_addr_b_out_co)
                begin
                        if (sl_w_addr_b_co='1') then
                                sig_addr_b_in_co        <=      in_data(63 downto 32);
                        else
                                if (sig_flags_out_co(8)='0') then -- nao bloqueia incremento
                                        sig_addr_b_in_co        <=      sig_addr_b_increment;
                                else
                                        sig_addr_b_in_co<=sig_addr_b_out_co;
                                end if;
                        end if;
                end process;
        -- processo que libera escrita de addr_b no barramento inferior
                process (en_w_addr_b_b_co,sig_addr_b_out_co)
                begin
                        if (en_w_addr_b_b_co='1') then
                                addr_b_out      <=      sig_addr_b_out_co;
                        else
                                addr_b_out      <=      (others=>'Z');
                        end if;
                end process;
        -- processo que libera escrita de addr_b no barramento superior (caso de memoria para memoria)
                process (en_w_addr_b_f_co,sig_addr_b_out_co)
                begin
                        if (en_w_addr_b_f_co='1') then
```

```
                                addr_f_out      <=      sig_addr_b_out_co;
                        else
                                addr_f_out      <=      (others =>'Z');
                        end if;
                end process;

-- processos referente a COUNTER
        -- processo que calcula o decremento de counter

-- Update the variable only when updates are enabled
                sig_counter_increment   <=sig_counter_out_co + X\"FFFF\"; -- decrementa 1

        -- processo que calcula o novo valor de counter
                process (sl_w_counter_co, sig_counter_increment, in_data(31 downto 16))
                begin
                        if (sl_w_counter_co='1') then -- seleciona o dado vindo do banco de dados de canal
                                sig_counter_in_co       <= in_data(31 downto 16);
                        else
                                sig_counter_in_co       <= sig_counter_increment;
                        end if;
                end process;
        -- processo que informa a maquina de estados se counter for menor que offset
                process (sig_counter_out_co)
                begin
                        if (sig_counter_out_co=X\"0000\") then
                                counter_is_empty        <='1';
                        else
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣counter_is_empty␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣<='0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣process;
--␣processos␣referente␣a␣OFFSET
␣␣␣␣␣␣␣␣--␣processo␣que␣atualiza␣o␣proximo␣valor␣de␣offset
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣process␣(
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_offset_out_co,␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣sinal␣da␣saida␣do␣registrador␣offset
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_op_mode,␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣sinal␣da␣maquina␣de␣
    estados
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_counter_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣sinal␣que␣seleciona␣a␣origem,␣usa␣o␣mesmo
    ␣sinal␣de␣counter
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣)
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣begin
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sl_w_counter_co='1')␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_offset_in_co␣␣␣␣␣␣␣␣<=X
    \"0001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sl_op_mode␣=␣'0')␣then␣--␣
    leitura.␣somente␣incrementa
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_offset_in_co
    ␣␣␣␣␣␣␣␣<=␣␣sig_offset_out_co+X\"0001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣escrita,␣somente␣
    decrementa
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_offset_in_co
    ␣␣␣␣␣␣␣␣<=␣␣sig_offset_out_co+X\"FFFF\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣process;
␣␣␣␣␣␣␣␣--processo␣que␣informa␣a␣maquina␣de␣estados␣quando␣
    offset␣alcancou␣o␣limite␣(do␣buffer)
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣process␣(sig_offset_out_co)
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣begin
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_offset_out_co(7␣downto␣0)=X\"FF
    \")␣then␣--␣eh␣por␣quantidade␣de␣transacoes
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣offset_is_full␣␣␣␣␣␣␣␣␣<='1';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣offset_is_full␣␣␣␣␣␣␣␣␣<='0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣process;
␣␣␣␣␣␣␣␣--␣processo␣que␣informa␣se␣offset␣esta␣vazio
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣process␣(sig_offset_out_co)
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣begin
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_offset_out_co=X\"0000\")␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣offset_is_empty␣␣␣␣␣␣␣␣<='1';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣offset_is_empty␣␣␣␣␣␣␣␣<='0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣process;
--␣processo␣referente␣a␣BUFFER
␣␣␣␣␣␣␣␣--␣processo␣que␣seleciona␣entrada␣E␣gera␣sinal␣de␣
    habilitacao␣de␣escrita␣no␣buffer
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣process␣(en_r_db_f_co,en_r_db_b_co,db_f_in,
    db_b_in,␣sig_flags_out_co)
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣begin
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_databuffer_co␣␣<=en_r_db_f_co␣
    OR␣en_r_db_b_co;␣--␣gera␣sinal␣de␣escrita
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣ideal␣seria␣fazer␣um␣tristate,␣porem
    ␣como␣existe␣o␣enable,␣nao␣faz␣diferenca.
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣se␣o␣sinal␣de␣habilitacoa␣estiver␣
    ativado␣eh␣poque␣necessita␣ler␣de␣algum␣barramento
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣se␣en_r_db_f_co␣for␣1␣e␣do␣
    barramento␣superior,␣senao␣␣do␣barramento␣inferior
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(en_r_db_f_co='1')␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣do␣barramento␣superior␣
    sempre␣sera␣32␣bits
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_data_in_buffer_co␣␣␣<=
    ␣␣␣␣␣␣db_f_in;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣case␣(sig_flags_out_co(1␣downto
    ␣0))␣is
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00\"␣␣␣␣␣=>␣−−␣
    08␣bits
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_data_in_buffer_co(7␣downto␣0)␣␣␣␣␣␣␣<=␣␣␣␣␣␣db_b_in(7␣
    downto␣0);␣−−␣somento␣os␣8␣menos␣significativos
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_data_in_buffer_co(31␣downto␣8)␣␣␣␣␣␣<=X\"000000\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01\"␣␣␣␣␣=>␣−−␣
    16␣bits
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_data_in_buffer_co(15␣downto␣0)␣␣␣␣␣␣<=␣␣␣␣␣␣db_b_in(15␣
    downto␣0);␣−−␣somento␣os␣16␣menos␣significativos
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_data_in_buffer_co(31␣downto␣16)␣␣␣␣␣<=X\"0000\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10\"␣␣␣␣␣=>␣−−␣
    32␣bits
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_data_in_buffer_co␣␣␣<=␣␣␣␣␣␣db_b_in;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11\"␣␣␣␣␣=>␣−−␣
    64␣bits
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_data_in_buffer_co␣␣␣<=␣␣␣␣␣␣db_b_in;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣others=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣null;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣case;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣process;
␣␣␣␣␣␣␣␣−−␣processo␣que␣permite␣escrita␣em␣db_f
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣process␣(en_w_db_f_co,␣sig_data_out_buffer_co)
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣begin
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(en_w_db_f_co='1')␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣db_f_out␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=
    ␣␣␣␣␣␣sig_data_out_buffer_co;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣db_f_out␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=(
    Others=>'Z');
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣process;
␣␣␣␣␣␣␣␣--␣processo␣que␣permite␣escrita␣em␣db_b
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣process␣(en_w_db_b_co,␣sig_data_out_buffer_co)
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣begin
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(en_w_db_b_co='1')␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣db_b_out␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=
    ␣␣␣␣␣␣sig_data_out_buffer_co;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣db_b_out␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=(
    Others=>'Z');
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣process;
END␣BHV_MAIN_OPERATION;

————————␣TERMINO␣DO␣ARQUIVO␣————————————
";
if ($grava){
        gravar("main_operation.vhd", $conteudo);
        }
if ($tela){
        tela("main_operation.vhd", $conteudo);
        }
$conteudo_tb="

library␣ieee;
use␣ieee.std_logic_1164.all;
use␣ieee.std_logic_arith.all;

entity␣MAIN_OPERATION_TB␣is
end␣MAIN_OPERATION_TB;

architecture␣TB_MAIN_OPERATION␣of␣MAIN_OPERATION_TB␣is
```

```
    component MAIN_OPERATION
        port(
                clk : in std_logic;
                reset : in std_logic;

                --entradas
                in_data :       in std_logic_vector (95 downto 0);
                db_f_in :       in std_logic_vector (31 downto 0);
                db_b_in :       in std_logic_vector (31 downto 0);

                -- saidas
                flags_out_co                           :       out std_logic_vector (15 downto 0);
                counter_less_offset             :       out STD_LOGIC;
                offset_if_full                          :       out STD_LOGIC;
                addr_f_out                                     :       out std_logic_vector (31 downto 0);
                addr_b_out                                     :       out std_logic_vector (31 downto 0);
                db_f_out                                                :       out std_logic_vector (31 downto 0);
                db_b_out                                                :       out std_logic_vector (31 downto 0);

                -- sinais provindos da unidade de controle --
                sl_op_mode                                     :       in STD_LOGIC;
                en_w_flags_co                          :       in STD_LOGIC;
```

```
                en_w_offset_co                          :
        in STD_LOGIC;
                en_w_counter_co                 :       in
    STD_LOGIC;
                sl_w_counter_co                 :       in
    STD_LOGIC;

                en_w_addr_f_co                          :
        in STD_LOGIC;
                sl_w_addr_f_co                          :
        in STD_LOGIC;
                en_w_addr_b_co                          :
        in STD_LOGIC;
                sl_w_addr_b_co                          :
        in STD_LOGIC;
                en_w_addr_f_f_co                        :
        in STD_LOGIC;

                en_w_addr_b_f_co                        :
        in STD_LOGIC;
                en_w_addr_b_b_co                        :
        in STD_LOGIC;
                en_w_db_f_co                            :
        in STD_LOGIC;
                en_w_db_b_co                            :
        in STD_LOGIC;
                en_r_db_f_co                            :
        in STD_LOGIC;

                en_r_db_b_co                            :
        in STD_LOGIC
        );
    end component;

    -- sinais necessarios para realizacao do TB
```

```
        signal  sig_clk
                        :       std_logic;
        signal  sig_reset
                        :       std_logic;

                --entradas
        signal  sig_in_data
                        :       std_logic_vector (95 downto 0);
        signal  sig_db_f_in
                        :       std_logic_vector (31 downto 0);
        signal  sig_db_b_in
                        :       std_logic_vector (31 downto 0);

                -- saidas
        signal  sig_flags_out_co
                        :       std_logic_vector (15 downto 0);
        signal  sig_counter_less_offset
                        :       STD_LOGIC;
        signal  sig_offset_if_full
                        :       STD_LOGIC;
        signal  sig_addr_f_out
                        :       std_logic_vector (31 downto 0);
        signal  sig_addr_b_out
                        :       std_logic_vector (31 downto 0);
        signal  sig_db_f_out
                        :       std_logic_vector (31 downto 0);
        signal  sig_db_b_out
                        :       std_logic_vector (31 downto 0);

                -- sinais provindos da unidade de controle ---
        signal  sig_sl_op_mode
                        :       STD_LOGIC;
```

```
        signal  sig_en_w_flags_co
                : STD_LOGIC;
        signal  sig_en_w_offset_co
                : STD_LOGIC;
        signal  sig_en_w_counter_co
                : STD_LOGIC;
        signal  sig_sl_w_counter_co
                : STD_LOGIC;
        signal  sig_en_w_addr_f_co
                : STD_LOGIC;
        signal  sig_sl_w_addr_f_co
                : STD_LOGIC;
        signal  sig_en_w_addr_b_co
                : STD_LOGIC;
        signal  sig_sl_w_addr_b_co
                : STD_LOGIC;
        signal  sig_en_w_addr_f_f_co
                : STD_LOGIC;
        signal  sig_en_w_addr_b_f_co
                : STD_LOGIC;
        signal  sig_en_w_addr_b_b_co
                : STD_LOGIC;
        signal  sig_en_w_db_f_co
                : STD_LOGIC;
        signal  sig_en_w_db_b_co
                : STD_LOGIC;
        signal  sig_en_r_db_f_co
                : STD_LOGIC;
        signal  sig_en_r_db_b_co
                : STD_LOGIC;

begin
--                    esperado: cada sinal reagir de acordo


-- Instanciamento do decodificador
```

```
␣␣␣␣␣␣␣␣main_op␣:␣MAIN_OPERATION␣port␣map
␣␣␣␣␣␣␣␣(
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_clk ,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_reset ,

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--entradas
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_in_data ,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_db_f_in␣␣␣␣␣,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_db_b_in ,

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣saidas
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_flags_out_co ,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_counter_less_offset ,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_offset_if_full ,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_addr_f_out ,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_addr_b_out ,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_db_f_out ,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_db_b_out ,

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣sinais␣provindos␣da␣unidade␣de␣controle---
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_sl_op_mode ,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_flags_co ,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_offset_co ,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_counter_co ,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_sl_w_counter_co ,

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_addr_f_co ,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_sl_w_addr_f_co ,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_addr_b_co ,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_sl_w_addr_b_co ,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_addr_f_f_co ,

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_addr_b_f_co ,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_addr_b_b_co ,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_db_f_co ,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_db_b_co ,
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_r_db_f_co ,

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_r_db_b_co
␣␣␣␣␣␣␣␣);


--␣geracao␣do␣clock
process
begin
␣␣␣␣␣␣␣␣sig_clk␣<=␣'0';
␣␣␣␣␣␣␣␣wait␣for␣5␣ns;
␣␣␣␣␣␣␣␣sig_clk␣<=␣'1';
␣␣␣␣␣␣␣␣wait␣for␣5␣ns;
end␣process;
process
begin

␣␣␣␣␣␣␣␣sig_reset␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
   ␣␣␣␣␣␣␣␣<='1';

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--entradas
␣␣␣␣␣␣␣␣sig_in_data␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
   ␣␣␣␣␣␣␣␣<=X\"AAAAA000BBBBB000000F0006\";
␣␣␣␣␣␣␣␣--␣addr_f=AAAAA000
␣␣␣␣␣␣␣␣--␣addr_b=BBBBB000
␣␣␣␣␣␣␣␣--␣counter=000F
␣␣␣␣␣␣␣␣--␣flags␣=0006
␣␣␣␣␣␣␣␣␣--(1~0)␣->␣2␣=␣32␣bits
␣␣␣␣␣␣␣␣␣--(3~2)␣->␣1␣=␣m-d
␣␣␣␣␣␣␣␣␣--(5~4)␣->␣0␣=␣simples
␣␣␣␣␣␣␣␣␣--(7~6)␣->␣0␣=␣incrementa␣sup/inf
␣␣␣␣␣␣␣␣␣--(9~8)␣->␣0␣=␣nao␣bloquear␣incremento
␣␣␣␣␣␣␣␣sig_db_f_in␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
   ␣␣␣␣␣␣␣␣<=X\"CCCCCC00\";
␣␣␣␣␣␣␣␣sig_db_b_in␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
   ␣␣␣␣␣␣␣␣<=X\"DDDDDD00\";
```

```
                    -- sinais provindos da unidade de controle --
        sig_sl_op_mode                                    <='0';
        sig_en_w_flags_co                                 <='0';
        sig_en_w_offset_co                                <='0';
        sig_en_w_counter_co                               <='0';
        sig_sl_w_counter_co                               <='0';
        sig_en_w_addr_f_co                                <='0';
        sig_sl_w_addr_f_co                                <='0';
        sig_en_w_addr_b_co                                <='0';
        sig_sl_w_addr_b_co                                <='0';
        sig_en_w_addr_f_f_co                    <='0';
        sig_en_w_addr_b_f_co                    <='0';
        sig_en_w_addr_b_b_co                    <='0';
        sig_en_w_db_f_co                                  <='0';
        sig_en_w_db_b_co                                  <='0';
        sig_en_r_db_f_co                                  <='1';
        sig_en_r_db_b_co                                  <='0';
        wait for 10 ns;
--testar sinais quando reset


        sig_reset                                         
        <='0';

                    --entradas
--      sig_in_data                                       
        <=X\"AAAAAAAABBBBBBBB000F0006\";
--      sig_db_f_in                                       
        <=X\"CCCCCCCC\";
--      sig_db_b_in                                       
        <=X\"DDDDDDDD\";

                    -- sinais provindos da unidade de controle --
        assert false report \"Testbench: Testa 32 bits\" 
    severity note;
```

```
␣␣␣␣␣␣␣␣sig_sl_op_mode␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    <='0';␣--␣diz␣que␣o␣modo␣esta␣em␣leitura␣(leitura)
␣␣␣␣␣␣␣␣sig_en_w_flags_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    <='1';␣--␣habilita␣escrita␣no␣registrador␣flags(habilita)
␣␣␣␣␣␣␣␣sig_en_w_offset_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';␣--␣
    habilita␣escrita␣no␣registrador␣offset(habilita)
␣␣␣␣␣␣␣␣sig_en_w_counter_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';␣
    --␣habilita␣escrita␣no␣registrador␣counter(habilita)
␣␣␣␣␣␣␣␣sig_sl_w_counter_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';␣
    --␣seleciona␣o␣dado␣a␣ser␣escrito␣no␣registrador␣counter(
    vinda␣de␣ch␣data)
␣␣␣␣␣␣␣␣sig_en_w_addr_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';
    ␣--␣habilita␣escrita␣no␣registrador␣addr_f(habilita)
␣␣␣␣␣␣␣␣sig_sl_w_addr_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';
    ␣--␣seleciona␣o␣dado␣a␣ser␣escrito␣no␣registrador␣addr_f(
    vinda␣de␣ch␣data)
␣␣␣␣␣␣␣␣sig_en_w_addr_b_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';
    ␣--␣habilita␣escrita␣no␣registrador␣addr_b(habilita)
␣␣␣␣␣␣␣␣sig_sl_w_addr_b_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';
    ␣--␣seleciona␣o␣dado␣a␣ser␣escrito␣no␣registrador␣addr_b(
    vinda␣de␣ch␣data)
␣␣␣␣␣␣␣␣sig_en_w_addr_f_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';␣--␣
    habilita␣escrita␣no␣barramento␣de␣dados␣superior␣pela␣saida␣
    do␣registrador␣addr_f(habilita)
␣␣␣␣␣␣␣␣sig_en_w_addr_b_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='0';␣--␣
    habilita␣escrita␣no␣barramento␣de␣dados␣superior␣pela␣saida␣
    do␣registrador␣addr_b(n␣habilita)
␣␣␣␣␣␣␣␣sig_en_w_addr_b_b_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';␣--␣
    habilita␣escrita␣no␣barramento␣de␣dados␣inferior␣pela␣saida␣
    do␣registrador␣addr_b(habilita)
␣␣␣␣␣␣␣␣sig_en_w_db_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';
    ␣--␣habilita␣escrita␣no␣barramento␣de␣dados␣superior(
    habilita)
␣␣␣␣␣␣␣␣sig_en_w_db_b_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    <='1';␣--␣habilita␣escrita␣no␣barramento␣de␣dados␣inferior(
    habilita)
```

```
␣␣␣␣␣␣␣␣sig_en_r_db_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    <=’1’;␣--␣habilita␣leitura␣no␣barramento␣de␣dados␣superior(n
    ␣habilita)
␣␣␣␣␣␣␣␣sig_en_r_db_b_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    <=’0’;␣␣␣␣␣␣␣--␣habilita␣leitura␣no␣barramento␣de␣dados␣
    inferior(n␣habilita)
␣␣␣␣␣␣␣␣wait␣for␣10␣ns;
␣␣␣␣␣␣␣␣if␣(sig_flags_out_co=X\"0006\")␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"flags:␣ok\"␣severity␣note
    ;
␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"flags:␣fail\"␣severity␣
    error;
␣␣␣␣␣␣␣␣end␣if;
--␣␣␣␣␣␣if␣(sig_counter_less_offset=X\"\")␣then
--␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"mensagem:␣ok\"␣severity␣
    note;
--␣␣␣␣␣␣else
--␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"mensagem:␣fail\"␣severity
    ␣error;
--␣␣␣␣␣␣end␣if;
--␣␣␣␣␣␣if␣(sig_offset_if_full=X\"\")␣then
--␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"mensagem:␣ok\"␣severity␣
    note;
--␣␣␣␣␣␣else
--␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"mensagem:␣fail\"␣severity
    ␣error;
--␣␣␣␣␣␣end␣if;
--␣␣␣␣␣␣if␣(sig_addr_f_out=X\"\")␣then
--␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"mensagem:␣ok\"␣severity␣
    note;
--␣␣␣␣␣␣else
--␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"mensagem:␣fail\"␣severity
    ␣error;
--␣␣␣␣␣␣end␣if;
--␣␣␣␣␣␣if␣(sig_addr_b_out=X\"\")␣then
```

```
--␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"mensagem:␣ok\"␣severity␣
    note;
--␣␣␣␣␣␣else
--␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"mensagem:␣fail\"␣severity
    ␣error;
--␣␣␣␣␣␣end␣if;
--␣␣␣␣␣␣if␣(sig_db_f_out=X\"\")␣then
--␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"mensagem:␣ok\"␣severity␣
    note;
--␣␣␣␣␣␣else
--␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"mensagem:␣fail\"␣severity
    ␣error;
--␣␣␣␣␣␣end␣if;
--␣␣␣␣␣␣if␣(sig_db_b_out=X\"\")␣then
--␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"mensagem:␣ok\"␣severity␣
    note;
--␣␣␣␣␣␣else
--␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"mensagem:␣fail\"␣severity
    ␣error;
--␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣sig_sl_op_mode␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    <='0';␣--␣diz␣que␣o␣modo␣esta␣em␣leitura␣(leitura)
␣␣␣␣␣␣␣␣sig_en_w_flags_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    <='0';␣--␣habilita␣escrita␣no␣registrador␣flags(n␣habilita)
␣␣␣␣␣␣␣␣sig_en_w_offset_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='0';␣--␣
    habilita␣escrita␣no␣registrador␣offset(n␣habilita)
␣␣␣␣␣␣␣␣sig_en_w_counter_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='0';␣
    --␣habilita␣escrita␣no␣registrador␣counter(n␣habilita)
␣␣␣␣␣␣␣␣sig_sl_w_counter_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='0';␣
    --␣seleciona␣o␣dado␣a␣ser␣escrito␣no␣registrador␣counter(
    vinda␣de␣increment)
␣␣␣␣␣␣␣␣sig_en_w_addr_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';
    ␣--␣habilita␣escrita␣no␣registrador␣addr_f(habilita)
␣␣␣␣␣␣␣␣sig_sl_w_addr_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='0';
    ␣--␣seleciona␣o␣dado␣a␣ser␣escrito␣no␣registrador␣addr_f(
    vinda␣de␣increment)
```

```
␣␣␣␣␣␣␣␣sig_en_w_addr_b_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';
    ␣--␣habilita␣escrita␣no␣registrador␣addr_b(habilita)
␣␣␣␣␣␣␣␣sig_sl_w_addr_b_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='0';
    ␣--␣seleciona␣o␣dado␣a␣ser␣escrito␣no␣registrador␣addr_b(
    vinda␣de␣increment)
␣␣␣␣␣␣␣␣sig_en_w_addr_f_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';␣--␣
    habilita␣escrita␣no␣barramento␣de␣dados␣superior␣pela␣saida␣
    do␣registrador␣addr_f(habilita)
␣␣␣␣␣␣␣␣sig_en_w_addr_b_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='0';␣--␣
    habilita␣escrita␣no␣barramento␣de␣dados␣superior␣pela␣saida␣
    do␣registrador␣addr_b(n␣habilita)
␣␣␣␣␣␣␣␣sig_en_w_addr_b_b_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';␣--␣
    habilita␣escrita␣no␣barramento␣de␣dados␣inferior␣pela␣saida␣
    do␣registrador␣addr_b(habilita)
␣␣␣␣␣␣␣␣sig_en_w_db_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';
    ␣--␣habilita␣escrita␣no␣barramento␣de␣dados␣superior(
    habilita)
␣␣␣␣␣␣␣␣sig_en_w_db_b_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    <='1';␣--␣habilita␣escrita␣no␣barramento␣de␣dados␣inferior(
    habilita)
␣␣␣␣␣␣␣␣sig_en_r_db_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    <='0';␣--␣habilita␣leitura␣no␣barramento␣de␣dados␣superior(n
    ␣habilita)
␣␣␣␣␣␣␣␣sig_en_r_db_b_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    <='1';␣␣␣␣␣␣␣--␣habilita␣leitura␣no␣barramento␣de␣dados␣
    inferior(n␣habilita)
␣␣␣␣␣␣␣␣wait␣for␣10␣ns;
␣␣␣␣␣␣␣␣assert␣false␣report␣\"Testbench:␣testa␣8␣bits\"␣
    severity␣note;
␣␣␣␣␣␣␣␣sig_in_data␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣<=X\"AAAAA000BBBBB000000F0004\";
␣␣␣␣␣␣␣␣--␣addr_f=AAAAA000
␣␣␣␣␣␣␣␣--␣addr_b=BBBBB000
␣␣␣␣␣␣␣␣--␣counter=000F
␣␣␣␣␣␣␣␣--␣flags␣=0004
␣␣␣␣␣␣␣␣␣--(1~0)␣->␣0␣=␣8␣bits
```

```
␣␣␣␣␣␣␣␣␣--(3~2)␣->␣1␣=␣m-d
␣␣␣␣␣␣␣␣␣--(5~4)␣->␣0␣=␣simples
␣␣␣␣␣␣␣␣␣--(7~6)␣->␣0␣=␣incrementa␣sup/inf
␣␣␣␣␣␣␣␣␣--(9~8)␣->␣0␣=␣nao␣bloquear␣incremento
␣␣␣␣␣␣␣␣sig_sl_op_mode␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    <='0';␣--␣diz␣que␣o␣modo␣esta␣em␣leitura␣(leitura)
␣␣␣␣␣␣␣␣sig_en_w_flags_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    <='1';␣--␣habilita␣escrita␣no␣registrador␣flags(habilita)
␣␣␣␣␣␣␣␣sig_en_w_offset_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';␣--␣
    habilita␣escrita␣no␣registrador␣offset(habilita)
␣␣␣␣␣␣␣␣sig_en_w_counter_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';␣
    --␣habilita␣escrita␣no␣registrador␣counter(habilita)
␣␣␣␣␣␣␣␣sig_sl_w_counter_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';␣
    --␣seleciona␣o␣dado␣a␣ser␣escrito␣no␣registrador␣counter(
    vinda␣de␣ch␣data)
␣␣␣␣␣␣␣␣sig_en_w_addr_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';
    ␣--␣habilita␣escrita␣no␣registrador␣addr_f(habilita)
␣␣␣␣␣␣␣␣sig_sl_w_addr_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';
    ␣--␣seleciona␣o␣dado␣a␣ser␣escrito␣no␣registrador␣addr_f(
    vinda␣de␣ch␣data)
␣␣␣␣␣␣␣␣sig_en_w_addr_b_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';
    ␣--␣habilita␣escrita␣no␣registrador␣addr_b(habilita)
␣␣␣␣␣␣␣␣sig_sl_w_addr_b_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';
    ␣--␣seleciona␣o␣dado␣a␣ser␣escrito␣no␣registrador␣addr_b(
    vinda␣de␣ch␣data)
␣␣␣␣␣␣␣␣sig_en_w_addr_f_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';␣--␣
    habilita␣escrita␣no␣barramento␣de␣dados␣superior␣pela␣saida␣
    do␣registrador␣addr_f(habilita)
␣␣␣␣␣␣␣␣sig_en_w_addr_b_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='0';␣--␣
    habilita␣escrita␣no␣barramento␣de␣dados␣superior␣pela␣saida␣
    do␣registrador␣addr_b(n␣habilita)
␣␣␣␣␣␣␣␣sig_en_w_addr_b_b_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';␣--␣
    habilita␣escrita␣no␣barramento␣de␣dados␣inferior␣pela␣saida␣
    do␣registrador␣addr_b(habilita)
␣␣␣␣␣␣␣␣sig_en_w_db_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';
    ␣--␣habilita␣escrita␣no␣barramento␣de␣dados␣superior(
```

```
habilita)
␣␣␣␣␣␣␣␣sig_en_w_db_b_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
<='1';␣--␣habilita␣escrita␣no␣barramento␣de␣dados␣inferior(
habilita)
␣␣␣␣␣␣␣␣sig_en_r_db_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
<='1';␣--␣habilita␣leitura␣no␣barramento␣de␣dados␣superior(n
␣habilita)
␣␣␣␣␣␣␣␣sig_en_r_db_b_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
<='0';␣␣␣␣␣␣␣--␣habilita␣leitura␣no␣barramento␣de␣dados␣
inferior(n␣habilita)
␣␣␣␣␣␣␣␣wait␣for␣10␣ns;
␣␣␣␣␣␣␣␣sig_sl_op_mode␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
<='0';␣--␣diz␣que␣o␣modo␣esta␣em␣leitura␣(leitura)
␣␣␣␣␣␣␣␣sig_en_w_flags_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
<='0';␣--␣habilita␣escrita␣no␣registrador␣flags(n␣habilita)
␣␣␣␣␣␣␣␣sig_en_w_offset_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='0';␣--␣
habilita␣escrita␣no␣registrador␣offset(n␣habilita)
␣␣␣␣␣␣␣␣sig_en_w_counter_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='0';␣
--␣habilita␣escrita␣no␣registrador␣counter(n␣habilita)
␣␣␣␣␣␣␣␣sig_sl_w_counter_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='0';␣
--␣seleciona␣o␣dado␣a␣ser␣escrito␣no␣registrador␣counter(
vinda␣de␣increment)
␣␣␣␣␣␣␣␣sig_en_w_addr_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';
␣--␣habilita␣escrita␣no␣registrador␣addr_f(habilita)
␣␣␣␣␣␣␣␣sig_sl_w_addr_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='0';
␣--␣seleciona␣o␣dado␣a␣ser␣escrito␣no␣registrador␣addr_f(
vinda␣de␣increment)
␣␣␣␣␣␣␣␣sig_en_w_addr_b_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';
␣--␣habilita␣escrita␣no␣registrador␣addr_b(habilita)
␣␣␣␣␣␣␣␣sig_sl_w_addr_b_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='0';
␣--␣seleciona␣o␣dado␣a␣ser␣escrito␣no␣registrador␣addr_b(
vinda␣de␣increment)
␣␣␣␣␣␣␣␣sig_en_w_addr_f_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';␣--␣
habilita␣escrita␣no␣barramento␣de␣dados␣superior␣pela␣saida␣
do␣registrador␣addr_f(habilita)
```

```
␣␣␣␣␣␣␣␣sig_en_w_addr_b_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='0';␣--␣
    habilita␣escrita␣no␣barramento␣de␣dados␣superior␣pela␣saida␣
    do␣registrador␣addr_b(n␣habilita)
␣␣␣␣␣␣␣␣sig_en_w_addr_b_b_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';␣--␣
    habilita␣escrita␣no␣barramento␣de␣dados␣inferior␣pela␣saida␣
    do␣registrador␣addr_b(habilita)
␣␣␣␣␣␣␣␣sig_en_w_db_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';
    ␣--␣habilita␣escrita␣no␣barramento␣de␣dados␣superior(
    habilita)
␣␣␣␣␣␣␣␣sig_en_w_db_b_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    <='1';␣--␣habilita␣escrita␣no␣barramento␣de␣dados␣inferior(
    habilita)
␣␣␣␣␣␣␣␣sig_en_r_db_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    <='0';␣--␣habilita␣leitura␣no␣barramento␣de␣dados␣superior(n
    ␣habilita)
␣␣␣␣␣␣␣␣sig_en_r_db_b_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    <='1';␣␣␣␣␣␣␣--␣habilita␣leitura␣no␣barramento␣de␣dados␣
    inferior(n␣habilita)
␣␣␣␣␣␣␣␣wait␣for␣10␣ns;
␣␣␣␣␣␣␣␣sig_in_data␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣<=X\"AAAAA000BBBBB000000F0005\";
␣␣␣␣␣␣␣␣assert␣false␣report␣\"Testbench:␣testa␣16␣bits\"␣
    severity␣note;
␣␣␣␣␣␣␣␣--␣addr_f=AAAAA000
␣␣␣␣␣␣␣␣--␣addr_b=BBBBB000
␣␣␣␣␣␣␣␣--␣counter=000F
␣␣␣␣␣␣␣␣--␣flags␣=0005
␣␣␣␣␣␣␣␣␣--(1~0)␣->␣1␣=␣8␣bits
␣␣␣␣␣␣␣␣␣--(3~2)␣->␣1␣=␣m-d
␣␣␣␣␣␣␣␣␣--(5~4)␣->␣0␣=␣simples
␣␣␣␣␣␣␣␣␣--(7~6)␣->␣0␣=␣incrementa␣sup/inf
␣␣␣␣␣␣␣␣␣--(9~8)␣->␣0␣=␣nao␣bloquear␣incremento
␣␣␣␣␣␣␣␣sig_sl_op_mode␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    <='0';␣--␣diz␣que␣o␣modo␣esta␣em␣leitura␣(leitura)
␣␣␣␣␣␣␣␣sig_en_w_flags_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    <='1';␣--␣habilita␣escrita␣no␣registrador␣flags(habilita)
```

```
␣␣␣␣␣␣␣␣sig_en_w_offset_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';␣--␣
    habilita␣escrita␣no␣registrador␣offset(habilita)
␣␣␣␣␣␣␣␣sig_en_w_counter_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';␣
    --␣habilita␣escrita␣no␣registrador␣counter(habilita)
␣␣␣␣␣␣␣␣sig_sl_w_counter_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';␣
    --␣seleciona␣o␣dado␣a␣ser␣escrito␣no␣registrador␣counter(
    vinda␣de␣ch␣data)
␣␣␣␣␣␣␣␣sig_en_w_addr_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';
    ␣--␣habilita␣escrita␣no␣registrador␣addr_f(habilita)
␣␣␣␣␣␣␣␣sig_sl_w_addr_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';
    ␣--␣seleciona␣o␣dado␣a␣ser␣escrito␣no␣registrador␣addr_f(
    vinda␣de␣ch␣data)
␣␣␣␣␣␣␣␣sig_en_w_addr_b_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';
    ␣--␣habilita␣escrita␣no␣registrador␣addr_b(habilita)
␣␣␣␣␣␣␣␣sig_sl_w_addr_b_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';
    ␣--␣seleciona␣o␣dado␣a␣ser␣escrito␣no␣registrador␣addr_b(
    vinda␣de␣ch␣data)
␣␣␣␣␣␣␣␣sig_en_w_addr_f_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';␣--␣
    habilita␣escrita␣no␣barramento␣de␣dados␣superior␣pela␣saida␣
    do␣registrador␣addr_f(habilita)
␣␣␣␣␣␣␣␣sig_en_w_addr_b_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='0';␣--␣
    habilita␣escrita␣no␣barramento␣de␣dados␣superior␣pela␣saida␣
    do␣registrador␣addr_b(n␣habilita)
␣␣␣␣␣␣␣␣sig_en_w_addr_b_b_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';␣--␣
    habilita␣escrita␣no␣barramento␣de␣dados␣inferior␣pela␣saida␣
    do␣registrador␣addr_b(habilita)
␣␣␣␣␣␣␣␣sig_en_w_db_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';
    ␣--␣habilita␣escrita␣no␣barramento␣de␣dados␣superior(
    habilita)
␣␣␣␣␣␣␣␣sig_en_w_db_b_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    <='1';␣--␣habilita␣escrita␣no␣barramento␣de␣dados␣inferior(
    habilita)
␣␣␣␣␣␣␣␣sig_en_r_db_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    <='1';␣--␣habilita␣leitura␣no␣barramento␣de␣dados␣superior(n
    ␣habilita)
```

```
␣␣␣␣␣␣␣␣sig_en_r_db_b_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    <='0';␣␣␣␣␣␣−−␣habilita␣leitura␣no␣barramento␣de␣dados␣
    inferior(n␣habilita)
␣␣␣␣␣␣␣␣wait␣for␣10␣ns;
␣␣␣␣␣␣␣␣sig_sl_op_mode␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    <='0';␣−−␣diz␣que␣o␣modo␣esta␣em␣leitura␣(leitura)
␣␣␣␣␣␣␣␣sig_en_w_flags_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    <='0';␣−−␣habilita␣escrita␣no␣registrador␣flags(n␣habilita)
␣␣␣␣␣␣␣␣sig_en_w_offset_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='0';␣−−␣
    habilita␣escrita␣no␣registrador␣offset(n␣habilita)
␣␣␣␣␣␣␣␣sig_en_w_counter_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='0';␣
    −−␣habilita␣escrita␣no␣registrador␣counter(n␣habilita)
␣␣␣␣␣␣␣␣sig_sl_w_counter_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='0';␣
    −−␣seleciona␣o␣dado␣a␣ser␣escrito␣no␣registrador␣counter(
    vinda␣de␣increment)
␣␣␣␣␣␣␣␣sig_en_w_addr_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';
    ␣−−␣habilita␣escrita␣no␣registrador␣addr_f(habilita)
␣␣␣␣␣␣␣␣sig_sl_w_addr_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='0';
    ␣−−␣seleciona␣o␣dado␣a␣ser␣escrito␣no␣registrador␣addr_f(
    vinda␣de␣increment)
␣␣␣␣␣␣␣␣sig_en_w_addr_b_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';
    ␣−−␣habilita␣escrita␣no␣registrador␣addr_b(habilita)
␣␣␣␣␣␣␣␣sig_sl_w_addr_b_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='0';
    ␣−−␣seleciona␣o␣dado␣a␣ser␣escrito␣no␣registrador␣addr_b(
    vinda␣de␣increment)
␣␣␣␣␣␣␣␣sig_en_w_addr_f_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';␣−−␣
    habilita␣escrita␣no␣barramento␣de␣dados␣superior␣pela␣saida␣
    do␣registrador␣addr_f(habilita)
␣␣␣␣␣␣␣␣sig_en_w_addr_b_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='0';␣−−␣
    habilita␣escrita␣no␣barramento␣de␣dados␣superior␣pela␣saida␣
    do␣registrador␣addr_b(n␣habilita)
␣␣␣␣␣␣␣␣sig_en_w_addr_b_b_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';␣−−␣
    habilita␣escrita␣no␣barramento␣de␣dados␣inferior␣pela␣saida␣
    do␣registrador␣addr_b(habilita)
␣␣␣␣␣␣␣␣sig_en_w_db_f_co␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';
    ␣−−␣habilita␣escrita␣no␣barramento␣de␣dados␣superior(
```

```
    habilita)
        sig_en_w_db_b_co                                    <='1'; -- habilita escrita no barramento de dados inferior(
    habilita)
        sig_en_r_db_f_co                                    <='0'; -- habilita leitura no barramento de dados superior(n
    habilita)
        sig_en_r_db_b_co                                    <='1';       -- habilita leitura no barramento de dados
    inferior(n habilita)

--                    if (sig_r_status='1') then
--                                assert false report \"caso 5:
    r_status: ok\" severity note;
--                    else
--                                assert false report \"caso 5:
    r_status: fail\" severity error;
--                    end if;
        -- fim teste caso 5

        wait for 5 ns;
        assert false report \"Testbench: MAIN_OPERATION
    concluido\" severity note;
        wait;
end process;


end TB_MAIN_OPERATION;
```

```
configuration CFG_TB_MAIN_OPERATION of MAIN_OPERATION_TB is
    for TB_MAIN_OPERATION
    end for;
end CFG_TB_MAIN_OPERATION;
```

```
    ";
if ($gerar_tb){
        if ($grava){
                gravar("tb_main_operation.vhd", $conteudo_tb);
                }
        if ($tela){
                tela("tb_main_operation.vhd", $conteudo_tb);
                }
}
}
?>
```

B.0.5 dados_canal.php

```
<?
//include ("funcoes.php");
$gerar_tb=$_GET['gerar_tb'];
$nro_canal=$_GET['nro_canal'];
$grava=$_GET['grava'];
$tela=$_GET['tela'];

function dados_canal($gerar_tb, $nro_canal,$grava,$tela){
$conteudo="

library␣IEEE;
use␣IEEE.STD_LOGIC_1164.all;
use␣IEEE.STD_LOGIC_ARITH.all;
```

```
entity␣CHANNEL_DATA␣is
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣port(
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣clk␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣in␣
    std_logic;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣reset␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣in␣std_logic;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_in␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣in␣
    std_logic_vector␣␣(95␣downto␣00);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣out␣
    std_logic_vector␣(95␣downto␣0);

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_w_data␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣in␣std_logic;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_r_ch␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣in␣
    std_logic_vector␣(4␣downto␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_ch␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣in␣
    std_logic_vector␣(4␣downto␣0)
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣);
end␣CHANNEL_DATA;

ARCHITECTURE␣BHV_CHANNEL_DATA␣OF␣CHANNEL_DATA␣IS
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣COMPONENT␣reg␣is
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣Generic␣(
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣N␣:␣Integer␣:=96
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣port(

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣reset,␣clk,␣carga␣:␣IN␣␣␣
    STD_LOGIC;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣D␣:␣␣␣IN␣␣␣␣␣␣std_logic_vector␣
    (N-1␣DOWNTO␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣Q␣:␣␣␣OUT␣␣␣␣␣std_logic_vector␣
    (N-1␣DOWNTO␣0)
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣component;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_data_out_ch00␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣std_logic_vector␣(95␣DOWNTO␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_en_w_ch00␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣STD_LOGIC;
";
for ($i=1;$i<$nro_canal;$i++){
        if ($i<10){
```

```
                    $conteudo.="
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_data_out_ch0$i␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣std_logic_vector␣(95␣DOWNTO␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_en_w_ch0$i␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣STD_LOGIC;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣";
          }
          else{
                    $conteudo.="
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_data_out_ch$i␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣std_logic_vector␣(95␣DOWNTO␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣signal␣sig_en_w_ch$i␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣:␣STD_LOGIC;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣";
          }
}
$conteudo.="
BEGIN

␣␣␣␣␣␣␣␣chn00␣␣␣:␣reg␣␣␣generic␣map(␣N␣␣=>␣96␣)␣port␣map␣(␣
    reset,␣clk,␣sig_en_w_ch00,␣data_in,␣sig_data_out_ch00
    ␣␣␣␣␣␣␣␣);
";
for ($i=1;$i<$nro_canal;$i++){
          if ($i<10){
                    $conteudo.="
␣␣␣␣␣␣␣␣chn0$i␣␣:␣reg␣␣␣generic␣map(␣N␣␣=>␣96␣)␣port␣map␣(␣
    reset,␣clk,␣sig_en_w_ch0$i,␣data_in,␣sig_data_out_ch0$i
    ␣␣␣␣␣␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣";
          }
          else{
                    $conteudo.="
␣␣␣␣␣␣␣␣chn$i␣␣␣:␣reg␣␣␣generic␣map(␣N␣␣=>␣96␣)␣port␣map␣(␣
    reset,␣clk,␣sig_en_w_ch$i,␣data_in,␣sig_data_out_ch$i
    ␣␣␣␣␣␣␣␣);
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣";
          }
}
$conteudo.="


␣␣␣␣␣␣␣␣--␣processa␣a␣habilitacao␣de␣gravar␣dados␣no␣banco
␣␣␣␣␣␣␣␣process␣(clk,␣en_w_data,␣sl_w_ch)
␣␣␣␣␣␣␣␣begin
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(clk'EVENT␣and␣clk='1')␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(en_w_data='1')␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if(
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_ch=\"00000\"␣or
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_ch=\"00001\"␣or
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_ch=\"00010\"␣or
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_ch=\"00011\"␣or
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_ch=\"00100\"␣or
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_ch=\"00101\"␣or
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_ch=\"00110\"␣or
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_ch=\"00111\"␣or
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_ch=\"01000\"␣or
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_ch=\"01001\"␣or
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_ch=\"01010\"␣or
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_ch=\"01011\"␣or
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_ch=\"01100\"␣or
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_ch=\"01101\"␣or
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_ch=\"01110\"␣or
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_ch=\"01111\"␣or
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_ch=\"10000\"␣or
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_ch=\"10001\"␣or
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_ch=\"10010\"␣or
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_ch=\"10011\"␣or
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_ch=\"10100\"␣or
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_ch=\"10101\"␣or
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_ch=\"10110\"␣or
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_ch=\"10111\"␣or
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_ch=\"11000\"␣or
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_ch=\"11001\"␣or
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_ch=\"11010\"␣or
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_ch=\"11011\"␣or
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_ch=\"11100\"␣or
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_ch=\"11101\"␣or
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_ch=\"11110\"␣or
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_ch=\"11111\"
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣)␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_ch00␣␣␣<=␣(not
  ␣sl_w_ch(4))␣and␣(not␣sl_w_ch(3))␣and␣(not␣sl_w_ch(2))␣and␣(
  not␣sl_w_ch(1))␣and␣(not␣sl_w_ch(0))␣;
";
$habilita=lista_habilitados_dados_canal("en_w_data","sl_w_ch");
for ($i=1;$i<$nro_canal;$i++){
        if ($i<10){
                $conteudo.="
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_ch0$i␣␣<=␣".
  $habilita[$i].";";
        }
        else{
                $conteudo.="
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_ch$i␣␣␣<=␣".
  $habilita[$i].";";
        }
}

$conteudo.="
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_ch00␣␣␣<=␣'0';
";
for ($i=1;$i<$nro_canal;$i++){
        if ($i<10){
                $conteudo.="
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_ch0$i␣␣<=␣'0';
  ";
```

```
        }
        else{
                $conteudo.="
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_ch$i␣␣␣<=␣'0';
    ";
        }
}$conteudo.="
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
";
for ($i=0;$i<$nro_canal;$i++){
        if ($i<10){
                $conteudo.="
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_ch0$i␣␣<=␣'0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣";
        }
        else{
                $conteudo.="
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_ch$i␣␣␣<=␣'0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣";
        }
}
$conteudo.="
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣end␣process;

␣␣␣␣␣␣␣␣--␣processa␣a␣selecao␣da␣saida␣de␣dados␣do␣banco

␣␣␣␣␣␣␣␣process␣(clk,␣sl_r_ch,␣sig_data_out_ch00";
for ($i=1;$i<$nro_canal;$i++){
        if ($i<10){
                $conteudo.=",sig_data_out_ch0$i";
        }
        else{
                $conteudo.=",sig_data_out_ch$i";
```

```
        }
}
        $conteudo.=")
␣␣␣␣␣␣␣␣begin
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(clk'EVENT␣and␣clk='1')␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣case␣(sl_r_ch)␣is
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out␣␣␣␣␣␣␣␣
   ␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣sig_data_out_ch00;
";
$lista=lista_binario_dado_inteiro($nro_canal);
for ($i=1;$i<$nro_canal;$i++){
        if ($i<10){
                $conteudo.="
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"".$lista[$i]."\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out␣␣␣␣␣␣␣␣
   ␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣sig_data_out_ch0$i;
";
        }
        else{
                $conteudo.="
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"".$lista[$i]."\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out␣␣␣␣␣␣␣␣
   ␣␣␣␣␣␣␣␣<=␣␣␣␣␣␣sig_data_out_ch$i;
";
        }
}

$conteudo.="
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣others␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out␣␣␣␣␣␣␣␣
   ␣␣␣␣␣␣␣␣<=␣(others=>'0');
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣case;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣end␣process;
```

```
END␣BHV_CHANNEL_DATA;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
   ␣␣␣␣␣␣␣␣";
if ($grava){
        gravar("channel_data.vhd", $conteudo);
        }
if ($tela){
        tela("channel_data.vhd", $conteudo);
}
$h96="
   \"11111111111111111111111111111111111111111111111111111111111111111
   ";
$l96="
   \"00000000000000000000000000000000000000000000000000000000000000000
   ";
$conteudo_tb="
library␣ieee;
use␣ieee.std_logic_1164.all;
use␣ieee.std_logic_arith.all;

entity␣CHANNEL_DATA_TB␣is
end␣CHANNEL_DATA_TB;

architecture␣TB_CHANNEL_DATA␣of␣CHANNEL_DATA_TB␣is

␣␣␣␣component␣CHANNEL_DATA
␣␣␣␣␣␣␣␣port(
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣clk␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣in␣
   std_logic;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣reset␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣in␣std_logic;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_in␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣in␣
   std_logic_vector␣␣(95␣downto␣00);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣out␣
   std_logic_vector␣(95␣downto␣0);
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_w_data␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣in␣std_logic;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_r_ch␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣in␣
    std_logic_vector␣(4␣downto␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_ch␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣in␣
    std_logic_vector␣(4␣downto␣0)
␣␣␣␣␣␣␣␣);
␣␣␣␣end␣component;

␣␣␣␣--␣sinais␣necessarios␣para␣realizacao␣do␣TB

␣␣␣␣signal␣␣␣␣␣␣sig_clk␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic;
␣␣␣␣signal␣␣␣␣␣␣sig_reset␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic;

--in
␣␣␣␣␣␣␣␣signal␣␣sig_data_in␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣
    std_logic_vector␣␣(95␣downto␣00);
␣␣␣␣␣␣␣␣signal␣␣sig_en_w_data␣␣␣:␣␣␣␣␣␣␣std_logic;
␣␣␣␣␣␣␣␣signal␣␣sig_sl_r_ch␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣
    std_logic_vector␣(4␣downto␣0);
␣␣␣␣␣␣␣␣signal␣␣sig_sl_w_ch␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣
    std_logic_vector␣(4␣downto␣0);
--out
␣␣␣␣␣␣␣␣signal␣␣sig_data_out␣␣␣␣:␣␣␣␣␣␣␣std_logic_vector␣(95␣
    downto␣0);


begin

--␣Instanciamento␣do␣dados␣o␣canal
␣␣␣␣␣␣␣␣chn_data␣:␣CHANNEL_DATA␣port␣map
␣␣␣␣␣␣␣␣(␣␣␣␣␣␣␣sig_clk,␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_reset,␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣sig_data_in,␣␣␣␣sig_data_out,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_data,␣␣sig_sl_r_ch,␣␣␣␣sig_sl_w_ch
␣␣␣␣␣␣␣␣);
```

```
-- geracao do clock
process
begin
        sig_clk <= '0';
        wait for 1 ns;
        sig_clk <= '1';
        wait for 1 ns;
end process;

process
begin

        -- caso 0 : teste do sinal de saida para cada canal com
    reset habilitado
                sig_reset                                <='1';
                sig_data_in                              <=$h96;

                sig_en_w_data                    <='0';

                sig_sl_w_ch                              
    <=\"00000\";
                wait for 4 ns;
                sig_sl_r_ch                              
    <=\"00000\";
                wait for 4 ns;

                -- testando canal 0

                if (sig_data_out=$l96) then
                        assert false report \"caso 0.0:
    data_out: ok\" severity note;
                else
                        assert false report \"caso 0.0:
    data_out: fail\" severity error;
                end if;
";
```

```
for ($i=1;$i<$nro_canal;$i++){
$conteudo_tb.="

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_sl_w_ch␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=\"".
    $lista[$i]."\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣4␣ns;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_sl_r_ch␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=\"".
    $lista[$i]."\";

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣4␣ns;



␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣testando␣canal␣$i

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_data_out=$l96)␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣0.$i:␣
    data_out:␣ok\"␣severity␣note;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣0.$i:␣
    data_out:␣fail\"␣severity␣error;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
";
}
$conteudo_tb.="
␣␣␣␣␣␣␣␣--␣fim␣teste␣caso␣0

␣␣␣␣␣␣␣␣wait␣for␣4␣ns;

␣␣␣␣␣␣␣␣--␣caso␣1␣:␣teste␣do␣sinal␣de␣saida␣para␣cada␣canal␣com
    ␣reset␣desabilitado␣E␣sinal␣de␣carga␣desabilitado
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_reset␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_data_in␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=$h96;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_data␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_sl_w_ch␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    <=\"00000\";
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣4␣ns;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_sl_r_ch␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    <=\"00000\";

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣4␣ns;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣testando␣canal␣0

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_data_out=$l96)␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣1.0:␣
    data_out:␣ok\"␣severity␣note;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣1.0:␣
    data_out:␣fail\"␣severity␣error;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
";
for ($i=1;$i<$nro_canal;$i++){
$conteudo_tb.="

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_sl_w_ch␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=\"".
    $lista[$i]."\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣4␣ns;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_sl_r_ch␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=\"".
    $lista[$i]."\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣4␣ns;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣testando␣canal␣$i

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_data_out=$l96)␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣1.$i:␣
    data_out:␣ok\"␣severity␣note;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣1.$i:␣
    data_out:␣fail\"␣severity␣error;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
```

```
";
}
$vetor=vetor_verificacao($l96,$h96,32);
$conteudo_tb.="
␣␣␣␣␣␣␣␣--␣fim␣teste␣caso␣1


␣␣␣␣␣␣␣␣--␣caso␣2␣:␣Teste␣de␣carga␣dos␣registradores␣que␣
    armazemam␣dados␣do␣canal
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_reset␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_data_in␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=$h96;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_sl_w_ch␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    <=\"00000\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_data␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';

";
for ($canal=0;$canal<$nro_canal;$canal++){
        $conteudo_tb.="
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣4␣ns;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--Etapa␣$canal␣:␣dar␣carga␣no␣canal␣$canal␣e␣
    testar␣valor␣dos␣demais␣registradores

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_sl_w_ch␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣<=\"".int_para_binario($canal)."\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣4␣ns;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_sl_r_ch␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣<=\"00000\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣4␣ns;


␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣testando␣canal␣0

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_data_out=".$vetor[
    $canal][0].")␣then
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"
    caso␣2.$canal.0:␣data_out:␣ok\"␣severity␣note;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"
    caso␣2.$canal.0:␣data_out:␣fail\"␣severity␣error;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣";
            for ($i=1;$i<$nro_canal;$i++){
            $conteudo_tb.="

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_sl_r_ch␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣<=\"".$lista[$i]."\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣4␣ns;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣testando␣canal␣$i

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_data_out=".$vetor[
    $canal][$i].")␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"
    caso␣2.$canal.$i:␣data_out:␣ok\"␣severity␣note;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"
    caso␣2.$canal.$i:␣data_out:␣fail\"␣severity␣error;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣";
            }
}
$conteudo_tb.="
␣␣␣␣␣␣␣␣--␣fim␣teste␣caso␣2

␣␣␣␣␣␣␣␣wait␣for␣5␣ns;
␣␣␣␣␣␣␣␣assert␣false␣report␣\"Testbench:␣CHANNEL_DATA␣concluido
    \"␣severity␣note;
␣␣␣␣␣␣␣␣wait;
end␣process;
```

```
end␣TB_CHANNEL_DATA;
```

```
configuration␣CFG_TB_CHANNEL_DATA␣of␣CHANNEL_DATA_TB␣is
␣␣␣␣for␣TB_CHANNEL_DATA
␣␣␣␣end␣for;
end␣CFG_TB_CHANNEL_DATA;
```

```
";
if ($gerar_tb){
        if($gravar){
                gravar("tb_channel_data.vhd", $conteudo_tb);
                }
        if ($tela){
                tela("tb_channel_data.vhd", $conteudo_tb);
                }
}

}
?>
```

B.0.6 decodificador.php

```
<?
//include ("funcoes.php");
$gerar_tb=$_GET['gerar_tb'];
$nro_canal=$_GET['nro_canal'];
$grava=$_GET['grava'];
$tela=$_GET['tela'];

function decodificador($mapeamento,$gerar_tb, $nro_canal,$grava
    ,$tela){
$conteudo="
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_ARITH.all;
```

```
entity DECODER is
                port(
                        clk                             : in
    std_logic;
                        reset                   : in std_logic;

                        addr_f                  : in
    std_logic_vector (31 downto 0);
                        db_f                    : in
    std_logic_vector (31 downto 0);

                        en_decoder              : out std_logic
    ;
                        data_out                : out
    std_logic_vector (95 downto 0);

                        step                    : out
    std_logic_vector (1 downto 0);
                        sl_w_ch_out             : out
    std_logic_vector (4 downto 0);

                        r_status                : out std_logic
                );
end DECODER;

ARCHITECTURE BHV_DECODER OF DECODER IS

        COMPONENT reg is
```

```
␣␣␣␣␣␣␣␣Generic␣(
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣N␣:␣Integer␣:=32
␣␣␣␣␣␣␣␣);
␣␣␣␣␣␣␣␣port(
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣reset,␣clk,␣carga␣:␣IN␣␣␣STD_LOGIC;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣D␣:␣␣␣IN␣␣␣␣␣␣std_logic_vector␣(N-1␣DOWNTO␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣Q␣:␣␣␣OUT␣␣␣␣␣std_logic_vector␣(N-1␣DOWNTO␣0)
␣␣␣␣␣␣␣␣);
␣␣␣␣␣␣␣␣end␣component;


␣␣␣␣␣␣␣␣signal␣sig_en_w_data0␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣std_logic;
␣␣␣␣␣␣␣␣signal␣sig_en_w_data1␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣std_logic;
␣␣␣␣␣␣␣␣signal␣sig_en_w_data2␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣std_logic;
␣␣␣␣␣␣␣␣signal␣sig_db_f␣␣␣␣␣␣␣␣␣:␣std_logic_vector␣(31␣downto␣
    0);

BEGIN
--␣␣␣␣␣␣␣␣␣␣␣[31␣␣␣␣␣␣␣␣␣␣␣~␣␣␣␣␣␣␣␣␣␣␣␣7]␣[6~2]␣[1~0]
--␣␣␣␣␣␣␣␣␣␣␣[␣DMA␣mapped␣into␣memory␣␣␣␣␣␣␣]␣[ch]␣␣[step]
--␣addr_f␣=␣\"XXXX␣XXXX␣XXXX␣XXXX␣XXXX␣XXXX␣X␣XXXXX␣␣XX\"


␣␣␣␣␣␣␣␣data0␣␣␣:␣reg␣␣␣generic␣map(␣N␣␣=>␣32␣)␣port␣map␣(␣
    reset,␣clk,␣sig_en_w_data0,␣sig_db_f,␣data_out(95␣downto␣64)
    );
␣␣␣␣␣␣␣␣data1␣␣␣:␣reg␣␣␣generic␣map(␣N␣␣=>␣32␣)␣port␣map␣(␣
    reset,␣clk,␣sig_en_w_data1,␣sig_db_f,␣data_out(63␣downto␣32)
    );
␣␣␣␣␣␣␣␣data2␣␣␣:␣reg␣␣␣generic␣map(␣N␣␣=>␣32␣)␣port␣map␣(␣
    reset,␣clk,␣sig_en_w_data2,␣sig_db_f,␣data_out(31␣downto␣00)
    );
```

```
--␣processo␣que␣habilita␣o␣sinal␣de␣escrita␣dos␣registradores␣
   internos␣e␣indica␣a␣maquina␣de␣estados␣que␣estao␣utilizando␣
   o␣endereco␣mapeado␣o␣passo␣e␣o␣canal␣alvo
process␣(clk,␣addr_f,reset)
begin
␣␣␣␣␣␣␣␣if␣(reset='1')␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_decoder␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
   ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣step␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
   ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=\"00\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_ch_out␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
   ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=\"00000\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣r_status␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
   ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣'0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_data0␣␣␣␣␣␣␣␣␣␣<=␣'0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_data1␣␣␣␣␣␣␣␣␣␣<=␣'0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_data2␣␣␣␣␣␣␣␣␣␣<=␣'0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_db_f␣␣␣␣␣␣<=x\"00000000\";
␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣if␣(clk'EVENT␣and␣clk='1')␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(addr_f(31␣downto␣7)␣=␣\"$mapeamento
   \")then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_decoder␣␣␣␣␣␣␣␣␣␣␣␣␣␣
   ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣step␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
   ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=addr_f(1␣
   downto␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_ch_out␣␣␣␣␣␣␣␣␣␣␣␣␣
   ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=addr_f(6␣downto␣2)
   ;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_db_f␣␣␣␣␣␣␣␣␣␣<=db_f;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(addr_f(1␣downto␣0)=\"00\")␣
   then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_data0␣␣
   ␣␣␣␣␣␣␣␣<=␣'1';
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_data1␣␣
    ␣␣␣␣␣␣␣␣<=␣'0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_data2␣␣
    ␣␣␣␣␣␣␣␣<=␣'0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣r_status␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣'0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(addr_f(1␣downto␣0)
    =\"01\")␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_data0␣␣
    ␣␣␣␣␣␣␣␣<=␣'0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_data1␣␣
    ␣␣␣␣␣␣␣␣<=␣'1';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_data2␣␣
    ␣␣␣␣␣␣␣␣<=␣'0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣r_status␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣'0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(addr_f(1␣downto␣0)
    =\"10\")␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_data0␣␣
    ␣␣␣␣␣␣␣␣<=␣'0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_data1␣␣
    ␣␣␣␣␣␣␣␣<=␣'0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_data2␣␣
    ␣␣␣␣␣␣␣␣<=␣'1';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣r_status␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣'0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣elsif␣(addr_f(1␣downto␣0)
    =\"11\")␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_data0␣␣
    ␣␣␣␣␣␣␣␣<=␣'0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_data1␣␣
    ␣␣␣␣␣␣␣␣<=␣'0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_data2␣␣
    ␣␣␣␣␣␣␣␣<=␣'0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣r_status␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣'1';
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_data0␣␣
    ␣␣␣␣␣␣␣␣<=␣'0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_data1␣␣
    ␣␣␣␣␣␣␣␣<=␣'0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_data2␣␣
    ␣␣␣␣␣␣␣␣<=␣'0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣r_status␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<=␣'0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_data0␣␣␣␣␣␣␣␣␣␣<=␣'0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_data1␣␣␣␣␣␣␣␣␣␣<=␣'0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_en_w_data2␣␣␣␣␣␣␣␣␣␣<=␣'0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_decoder␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='0';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣step␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣ <=\"00\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;

␣␣␣␣␣␣␣␣end␣if;
end␣process;
END␣BHV_DECODER;




";

if ($grava){
        gravar("decoder.vhd", $conteudo);
}
if ($tela){
        tela("decoder.vhd", $conteudo);
        }
```

```
$h96="
   \"1111111111111111111111111111111111111111111111111111111111111111111111
   ";
$l96="
   \"0000000000000000000000000000000000000000000000000000000000000000000000
   ";

$conteudo_tb="
library␣ieee;
use␣ieee.std_logic_1164.all;
use␣ieee.std_logic_arith.all;

entity␣DECODER_TB␣is
end␣DECODER_TB;

architecture␣TB_DECODER␣of␣DECODER_TB␣is

␣␣␣␣component␣DECODER
␣␣␣␣␣␣␣␣port(
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣clk␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣in␣std_logic;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣reset␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣in␣std_logic;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣addr_f␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣in␣std_logic_vector␣
   (31␣downto␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣db_f␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣in␣std_logic_vector␣
   (31␣downto␣0);

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_decoder␣␣␣␣␣␣␣␣␣␣␣␣:␣out␣std_logic;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣out␣std_logic_vector␣
   (95␣downto␣0);

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣step␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣out␣std_logic_vector␣
   (1␣downto␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sl_w_ch_out␣␣␣␣␣␣␣␣␣␣␣:␣out␣std_logic_vector␣
   (4␣downto␣0);
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣r_status␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣out␣std_logic
␣␣␣␣␣␣␣␣);
␣␣␣␣end␣component;

␣␣␣␣--␣sinais␣necessarios␣para␣realizacao␣do␣TB

␣␣␣␣signal␣␣␣␣␣␣sig_clk␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic;
␣␣␣␣signal␣␣␣␣␣␣sig_reset␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣std_logic;
␣␣␣␣signal␣␣␣␣␣␣sig_addr_f␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣
    std_logic_vector␣(31␣downto␣0);
␣␣␣␣signal␣␣␣␣␣␣sig_db_f␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣
    std_logic_vector␣(31␣downto␣0);
␣␣␣␣␣␣␣␣signal␣␣sig_en_decoder␣␣:␣␣␣␣␣␣␣std_logic;
␣␣␣␣␣␣␣␣signal␣␣sig_data_out␣␣␣␣:␣␣␣␣␣␣␣std_logic_vector␣(95␣
    downto␣0);

␣␣␣␣␣␣␣␣signal␣␣sig_step␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣␣␣␣␣␣␣
    std_logic_vector␣(1␣downto␣0);
␣␣␣␣␣␣␣␣signal␣␣sig_sl_w_ch_out␣:␣␣␣␣␣␣␣std_logic_vector␣(4␣
    downto␣0);

␣␣␣␣␣␣␣␣signal␣␣sig_r_status␣␣␣␣:␣␣␣␣␣␣␣std_logic;
begin
--Testes␣desejados:
--␣␣␣␣␣␣*␣Testar␣sinais␣quando␣endereco␣de␣entrada␣nao␣
    corresponde␣ao␣mapeado␣ao␣dmac
--␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣esperado:␣nenhuma␣alteracao␣nos␣sinais
--␣␣␣␣␣␣*␣Testar␣sinais␣quando␣endereco␣de␣entrada␣corresponde␣
    ao␣mapeado␣ao␣dmac
--␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣esperado:␣en_decoder␣habilitar␣e␣demais␣sinais␣
    de␣acordo␣com␣o␣endereco
--␣␣␣␣␣␣␣␣␣␣␣␣␣␣+␣Testar␣cada␣mudanca␣de␣sinal
--␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣esperado:␣cada␣sinal␣reagir␣de␣acordo
```

```
--␣Instanciamento␣do␣decodificador
␣␣␣␣␣␣␣␣decod␣:␣decoder␣port␣map
␣␣␣␣␣␣␣␣(␣␣␣␣␣␣␣sig_clk ,␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_reset ,␣␣␣␣␣␣
    sig_addr_f ,␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_db_f ,␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    sig_en_decoder ,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_data_out ,␣␣␣sig_step ,␣␣␣␣␣␣␣sig_sl_w_ch_out
    ,␣␣␣␣␣␣␣␣sig_r_status
␣␣␣␣␣␣␣␣);


--␣geracao␣do␣clock
process
begin
␣␣␣␣␣␣␣␣sig_clk␣<=␣'0';
␣␣␣␣␣␣␣␣wait␣for␣5␣ns;
␣␣␣␣␣␣␣␣sig_clk␣<=␣'1';
␣␣␣␣␣␣␣␣wait␣for␣5␣ns;
end␣process;

process
begin
␣␣␣␣␣␣␣␣--caso␣0␣:␣resetar␣arquitetura␣e␣fazer␣teste␣de␣
    sanidade␣(no␣reset␣tudo␣deve␣ser␣0)
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_reset␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<='1';

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_addr_f␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    <=\"11111111111111111111111111111111\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_db_f␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    <=\"11111111111111111111111111111111\";

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣10␣ns;␣␣--␣deixado␣passar␣2␣ciclos␣de␣
    relogio

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣testando␣en_decoder
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_en_decoder='0')␣then
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣0:␣
    en_decoder:␣ok\"␣severity␣note;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣0:␣
    en_decoder:␣fail\"␣severity␣error;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;


␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣1␣ns;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣testando␣data_out
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_data_out=$196)␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣0:␣
    data_out:␣ok\"␣severity␣note;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣0:␣
    data_out:␣fail\"␣severity␣error;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣1␣ns;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣testando␣step
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_step=\"00\")␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣0:␣
    step:␣ok\"␣severity␣note;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣0:␣
    step:␣fail\"␣severity␣error;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣1␣ns;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣testando␣sl_w_ch_out
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_sl_w_ch_out=\"00000\")␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣0:␣
    sl_w_ch_out:␣ok\"␣severity␣note;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣0:␣
    sl_w_ch_out:␣fail\"␣severity␣error;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣1␣ns;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣testando␣r_status
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_r_status='0')␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣0:␣
    r_status:␣ok\"␣severity␣note;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣0:␣
    r_status:␣fail\"␣severity␣error;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣--␣fim␣teste␣caso␣0

␣␣␣␣␣␣␣␣wait␣for␣5␣ns;
␣␣␣␣␣␣␣␣sig_reset␣<=␣'0';
␣␣␣␣␣␣␣␣wait␣for␣10␣ns;

␣␣␣␣␣␣␣␣--␣caso␣1␣:␣Testar␣resposta␣da␣arquitetura␣quando␣
    endereco␣nao␣corresponde␣ao␣mapeado␣ao␣DMA
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_addr_f␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    <=\"11111111111111111111111111111111\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_db_f␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    <=\"11111111111111111111111111111111\";

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣10␣ns;␣␣--␣deixado␣passar␣2␣ciclos␣de␣
    relogio

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣testando␣en_decoder
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_en_decoder='0')␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣1:␣
    en_decoder:␣ok\"␣severity␣note;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣1:␣
    en_decoder:␣fail\"␣severity␣error;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣1␣ns;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣testando␣data_out
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_data_out=$196)␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣1:␣
    data_out:␣ok\"␣severity␣note;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣1:␣
    data_out:␣fail\"␣severity␣error;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣1␣ns;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣testando␣step
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_step=\"00\")␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣1:␣
    step:␣ok\"␣severity␣note;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣1:␣
    step:␣fail\"␣severity␣error;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣1␣ns;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣testando␣sl_w_ch_out
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_sl_w_ch_out=\"00000\")␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣1:␣
    sl_w_ch_out:␣ok\"␣severity␣note;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣1:␣
    sl_w_ch_out:␣fail\"␣severity␣error;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣1␣ns;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣testando␣r_status
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_r_status='0')␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣1:␣
    r_status:␣ok\"␣severity␣note;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣1:␣
    r_status:␣fail\"␣severity␣error;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣--␣fim␣teste␣caso␣1
```

```
␣␣␣␣␣␣␣␣wait␣for␣10␣ns;

␣␣␣␣␣␣␣␣--␣caso␣2␣:␣Testar␣resposta␣da␣arquitetura␣quando␣
    endereco␣corresponde␣ao␣mapeado␣ao␣DMA
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--selecionado␣canal␣31␣e␣passo␣0
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_addr_f␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    <=\"00000000000000000000000001111000\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_db_f␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    <=\"11111111111111111111111111111111\";

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣15␣ns;␣␣--␣deixado␣passar␣2␣ciclos␣de␣
    relogio

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣testando␣en_decoder
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_en_decoder='1')␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣2:␣
    en_decoder:␣ok\"␣severity␣note;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣2:␣
    en_decoder:␣fail\"␣severity␣error;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣1␣ns;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣testando␣data_out
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_data_out
    =\"11111111111111111111111111111110000000000000000000000000000000
    ␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣2:␣
    data_out:␣ok\"␣severity␣note;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣2:␣
    data_out:␣fail\"␣severity␣error;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣1␣ns;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣testando␣step
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_step=\"00\")␣then
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣2:␣
    step:␣ok\"␣severity␣note;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣2:␣
    step:␣fail\"␣severity␣error;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣1␣ns;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣testando␣sl_w_ch_out
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_sl_w_ch_out=\"11110\")␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣2:␣
    sl_w_ch_out:␣ok\"␣severity␣note;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣2:␣
    sl_w_ch_out:␣fail\"␣severity␣error;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣testando␣r_status
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_r_status='0')␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣1:␣
    r_status:␣ok\"␣severity␣note;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣1:␣
    r_status:␣fail\"␣severity␣error;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣--␣fim␣teste␣caso␣2

␣␣␣␣␣␣␣␣wait␣for␣5␣ns;

␣␣␣␣␣␣␣␣--␣caso␣3␣:␣Testar␣resposta␣da␣arquitetura␣quando␣
    endereco␣corresponde␣ao␣mapeado␣ao␣DMA
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--selecionado␣canal␣31␣e␣passo␣1
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_addr_f␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    <=\"00000000000000000000000001111001\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_db_f␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    <=\"11111111111111111111111111111111\";
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣10␣ns;␣␣--␣deixado␣passar␣2␣ciclos␣de␣
    relogio

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣testando␣en_decoder
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_en_decoder='1')␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣3:␣
    en_decoder:␣ok\"␣severity␣note;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣3:␣
    en_decoder:␣fail\"␣severity␣error;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣1␣ns;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣testando␣data_out
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_data_out
    =\"1111111111111111111111111111111111111111111111111111111111111110
    ␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣3:␣
    data_out:␣ok\"␣severity␣note;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣3:␣
    data_out:␣fail\"␣severity␣error;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣1␣ns;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣testando␣step
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_step=\"01\")␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣3:␣
    step:␣ok\"␣severity␣note;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣3:␣
    step:␣fail\"␣severity␣error;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣1␣ns;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣testando␣sl_w_ch_out
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_sl_w_ch_out=\"11110\")␣then
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣3:␣
    sl_w_ch_out:␣ok\"␣severity␣note;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣3:␣
    sl_w_ch_out:␣fail\"␣severity␣error;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣1␣ns;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣testando␣r_status
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_r_status='0')␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣3:␣
    r_status:␣ok\"␣severity␣note;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣3:␣
    r_status:␣fail\"␣severity␣error;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣--␣fim␣teste␣caso␣3

␣␣␣␣␣␣␣␣wait␣for␣5␣ns;

␣␣␣␣␣␣␣␣--␣caso␣4␣:␣Testar␣resposta␣da␣arquitetura␣quando␣
    endereco␣corresponde␣ao␣mapeado␣ao␣DMA
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--selecionado␣canal␣31␣e␣passo␣2
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_addr_f␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    <=\"00000000000000000000000001111010\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_db_f␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    <=\"11111111111111111111111111111111\";

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣10␣ns;␣␣--␣deixado␣passar␣2␣ciclos␣de␣
    relogio

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣testando␣en_decoder
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_en_decoder='1')␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣4:␣
    en_decoder:␣ok\"␣severity␣note;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣4:␣
    en_decoder:␣fail\"␣severity␣error;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣1␣ns;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣−−␣testando␣data_out
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_data_out
    =\"1111111111111111111111111111111111111111111111111111111111111111
    ␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣4:␣
    data_out:␣ok\"␣severity␣note;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣4:␣
    data_out:␣fail\"␣severity␣error;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣1␣ns;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣−−␣testando␣step
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_step=\"10\")␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣4:␣
    step:␣ok\"␣severity␣note;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣4:␣
    step:␣fail\"␣severity␣error;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣1␣ns;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣−−␣testando␣sl_w_ch_out
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_sl_w_ch_out=\"11110\")␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣4:␣
    sl_w_ch_out:␣ok\"␣severity␣note;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣4:␣
    sl_w_ch_out:␣fail\"␣severity␣error;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣1␣ns;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣−−␣testando␣r_status
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_r_status='0')␣then
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣4:␣
    r_status:␣ok\"␣severity␣note;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣4:␣
    r_status:␣fail\"␣severity␣error;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣——␣fim␣teste␣caso␣4

␣␣␣␣␣␣␣␣wait␣for␣5␣ns;

␣␣␣␣␣␣␣␣——␣caso␣5␣:␣Testar␣resposta␣da␣arquitetura␣quando␣
    endereco␣corresponde␣ao␣mapeado␣ao␣DMA
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣——selecionado␣canal␣31␣e␣passo␣3␣(solicitacao␣
    de␣leitura␣do␣registrador␣status)
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_addr_f␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    <=\"00000000000000000000000001111011\";
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_db_f␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    <=\"11111111111111111111111111111111\";

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣10␣ns;␣␣——␣deixado␣passar␣2␣ciclos␣de␣
    relogio

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣——␣testando␣en_decoder
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_en_decoder='1')␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣5:␣
    en_decoder:␣ok\"␣severity␣note;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣5:␣
    en_decoder:␣fail\"␣severity␣error;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣1␣ns;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣——␣testando␣data_out
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_data_out
    =\"1111111111111111111111111111111111111111111111111111111111111111
    ␣then
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣5:␣
    data_out:␣ok\"␣severity␣note;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣5:␣
    data_out:␣fail\"␣severity␣error;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣1␣ns;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣testando␣step
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_step=\"11\")␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣5:␣
    step:␣ok\"␣severity␣note;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣5:␣
    step:␣fail\"␣severity␣error;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣1␣ns;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣testando␣sl_w_ch_out
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_sl_w_ch_out=\"11110\")␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣5:␣
    sl_w_ch_out:␣ok\"␣severity␣note;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣5:␣
    sl_w_ch_out:␣fail\"␣severity␣error;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wait␣for␣1␣ns;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--␣testando␣r_status
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_r_status='1')␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣5:␣
    r_status:␣ok\"␣severity␣note;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣5:␣
    r_status:␣fail\"␣severity␣error;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if;
␣␣␣␣␣␣␣␣--␣fim␣teste␣caso␣5
```

```
␣␣␣␣␣␣␣␣wait␣for␣5␣ns;
␣␣␣␣␣␣␣␣assert␣false␣report␣\"Testbench:␣decoder␣concluido\"␣
    severity␣note;
␣␣␣␣␣␣␣␣wait;
end␣process;

end␣TB_DECODER;
```

```
configuration␣CFG_TB_DECODER␣of␣DECODER_TB␣is
␣␣␣␣for␣TB_DECODER
␣␣␣␣end␣for;
end␣CFG_TB_DECODER;
```

```
";
if ($gerar_tb){
        if ($grava){
                gravar("tb_decoder.vhd", $conteudo_tb);
        }
        if ($tela){
                tela("tb_decoder.vhd", $conteudo_tb);
                }
}
}
?>
```

B.0.7 dmac.php

```
<?
//include ("funcoes.php");
$gerar_tb=$_GET['gerar_tb'];
$nro_canal=$_GET['nro_canal'];
$grava=$_GET['grava'];
$tela=$_GET['tela'];
```

```
function reg($gerar_tb, $nro_canal,$grava,$tela){

$conteudo="
LIBRARY␣ieee;
use␣IEEE.STD_LOGIC_1164.all;
use␣IEEE.STD_LOGIC_ARITH.all;
--use␣IEEE.STD_LOGIC_unsigned.all;

ENTITY␣reg␣IS
GENERIC␣(
␣␣␣␣␣␣␣N␣:␣INTEGER:=␣32
);
PORT␣(␣Reset,␣clk,␣carga␣:␣IN␣␣␣STD_LOGIC;
␣␣␣␣␣␣␣D␣:␣␣␣IN␣␣␣␣␣␣std_logic_vector␣(N-1␣DOWNTO␣0);
␣␣␣␣␣␣␣Q␣:␣␣␣OUT␣␣␣␣␣std_logic_vector␣(N-1␣DOWNTO␣0));
END␣reg;

ARCHITECTURE␣bhv_reg␣OF␣reg␣IS
BEGIN
␣␣␣␣PROCESS␣(clk,␣Reset␣)
␣␣␣␣BEGIN
␣␣␣␣␣␣␣␣␣␣IF␣Reset␣=␣'1'␣THEN
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣Q␣<=␣␣(OTHERS␣=>␣'0'␣);--D0␣(N-1␣downto␣0);--(
    OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣ELSIF␣clk'EVENT␣AND␣clk␣=␣'1'␣THEN
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣IF␣carga='1'␣THEN
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣Q␣␣<=␣D;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣END␣IF;
␣␣␣␣␣␣␣␣␣␣END␣IF;
␣␣␣␣END␣PROCESS;
END␣bhv_reg;␣";

        if ($grava){
                gravar("reg.vhd", $conteudo);
                }
        if ($tela){
```

```
                tela("reg.vhd", $conteudo);
                }
$conteudo_tb="

";
        if ($gerar_tb){
                if ($grava){
                        gravar("tb_reg.vhd", $conteudo);
                        }
                if ($tela){
                        tela("tb_reg.vhd", $conteudo);
                        }
        }
}
function mem_reg($gerar_tb, $nro_canal,$grava,$tela){
$conteudo_mem="
library␣IEEE;
use␣IEEE.STD_LOGIC_1164.all;
use␣IEEE.STD_LOGIC_ARITH.all;
```

```
entity␣mem␣is
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣port(
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣address␣:␣IN␣STD_LOGIC_VECTOR␣(7␣DOWNTO␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣clock␣␣␣␣␣␣␣␣␣␣:␣IN␣STD_LOGIC␣␣:=␣'1';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data␣␣␣␣␣␣␣␣␣␣␣:␣IN␣STD_LOGIC_VECTOR␣(31␣
    DOWNTO␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wren␣␣␣␣␣␣␣␣␣␣␣:␣IN␣STD_LOGIC␣;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣q␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣OUT␣STD_LOGIC_VECTOR␣
    (31␣DOWNTO␣0)
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣);
end␣mem;

ARCHITECTURE␣bhv_mem␣OF␣mem␣IS
␣␣␣␣␣␣␣␣COMPONENT␣BUFFER_BLOCK␣is
␣␣␣␣␣␣␣␣GENERIC␣(
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣N␣:␣INTEGER:=␣32
␣␣␣␣␣␣␣␣);
␣␣␣␣␣␣␣␣port(
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣clk␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣in␣std_logic;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣reset␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣in␣std_logic;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_in␣␣␣␣␣␣␣␣␣:␣in␣std_logic_vector␣␣(N-1␣
    downto␣00);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣out␣std_logic_vector␣
    (N-1␣downto␣00);

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣en_w_data␣␣␣␣␣␣␣:␣in␣std_logic;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣addr␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣in␣std_logic_vector␣
    (7␣downto␣0)
␣␣␣␣␣␣␣␣);
␣␣␣␣␣␣␣␣end␣component;
␣␣␣␣␣␣␣␣constant␣low␣:␣std_logic␣:='0';
BEGIN
␣␣␣␣␣␣␣␣buffer_blk␣:␣buffer_block␣generic␣map(␣N␣␣=>␣32␣)␣port␣
    map(
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣clock,␣␣low,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data,␣␣␣␣␣␣␣␣␣␣␣q,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wren,␣␣␣␣␣␣␣␣␣␣␣address
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣);
END␣bhv_mem;
";
$conteudo_buf="


library␣IEEE;
use␣IEEE.STD_LOGIC_1164.all;
use␣IEEE.STD_LOGIC_ARITH.all;
```

```
entity␣BUFFER_BLOCK␣is
GENERIC␣(
```

```
        N : INTEGER := 32
);
                port(
                        clk                     : in std_logic;
                        reset                   : in std_logic;

                        data_in         : in std_logic_vector
    (N-1 downto 00);
                        data_out                : out
    std_logic_vector (N-1 downto 00);

                        en_w_data       : in std_logic;
                        addr                    : in
    std_logic_vector (7 downto 0)
                );
end BUFFER_BLOCK;

ARCHITECTURE BHV_BUFFER_BLOCK OF BUFFER_BLOCK IS
        signal sig_q000 :       std_logic_vector (N-1 downto 0)
    ;

        signal sig_q001 :       std_logic_vector (N-1 downto 0)
    ;

        signal sig_q002 :       std_logic_vector (N-1 downto 0)
    ;

        signal sig_q003 :       std_logic_vector (N-1 downto 0)
    ;

        signal sig_q004 :       std_logic_vector (N-1 downto 0)
    ;

        signal sig_q005 :       std_logic_vector (N-1 downto 0)
    ;
```

```
␣␣␣␣␣␣␣signal␣sig_q006␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q007␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q008␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q009␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q010␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q011␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q012␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q013␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q014␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q015␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q016␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q017␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;
```

```
␣␣␣␣␣␣␣signal␣sig_q018␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q019␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q020␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q021␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q022␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q023␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q024␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q025␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q026␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q027␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q028␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q029␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;
```

```
␣␣␣␣␣␣␣signal␣sig_q030␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q031␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q032␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q033␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q034␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q035␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q036␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q037␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q038␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q039␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q040␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q041␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;
```

```
␣␣␣␣␣␣␣signal␣sig_q042␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q043␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q044␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q045␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q046␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q047␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q048␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q049␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q050␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q051␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q052␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q053␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;
```

```
␣␣␣␣␣␣␣signal␣sig_q054␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q055␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q056␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q057␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q058␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q059␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q060␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q061␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q062␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q063␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q064␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q065␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;
```

```
␣␣␣␣␣␣␣signal␣sig_q066␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q067␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q068␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q069␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q070␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q071␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q072␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q073␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q074␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q075␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q076␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q077␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;
```

```
␣␣␣␣␣␣␣␣signal␣sig_q078␣:␣␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣␣signal␣sig_q079␣:␣␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣␣signal␣sig_q080␣:␣␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣␣signal␣sig_q081␣:␣␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣␣signal␣sig_q082␣:␣␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣␣signal␣sig_q083␣:␣␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣␣signal␣sig_q084␣:␣␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣␣signal␣sig_q085␣:␣␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣␣signal␣sig_q086␣:␣␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣␣signal␣sig_q087␣:␣␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣␣signal␣sig_q088␣:␣␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣␣signal␣sig_q089␣:␣␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;
```

```
␣␣␣␣␣␣␣signal␣sig_q090␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q091␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q092␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q093␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q094␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q095␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q096␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q097␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q098␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q099␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q100␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q101␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;
```

```
␣␣␣␣␣␣␣signal␣sig_q102␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q103␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q104␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q105␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q106␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q107␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q108␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q109␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q110␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q111␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q112␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q113␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;
```

```
␣␣␣␣␣␣␣signal␣sig_q114␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q115␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q116␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q117␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q118␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q119␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q120␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q121␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q122␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q123␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q124␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q125␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;
```

```
␣␣␣␣␣␣␣signal␣sig_q126␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q127␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q128␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q129␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q130␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q131␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q132␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q133␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q134␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q135␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q136␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q137␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;
```

```
␣␣␣␣␣␣␣␣signal␣sig_q138␣:␣␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣␣signal␣sig_q139␣:␣␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣␣signal␣sig_q140␣:␣␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣␣signal␣sig_q141␣:␣␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣␣signal␣sig_q142␣:␣␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣␣signal␣sig_q143␣:␣␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣␣signal␣sig_q144␣:␣␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣␣signal␣sig_q145␣:␣␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣␣signal␣sig_q146␣:␣␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣␣signal␣sig_q147␣:␣␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣␣signal␣sig_q148␣:␣␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣␣signal␣sig_q149␣:␣␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;
```

```
␣␣␣␣␣␣␣signal␣sig_q150␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q151␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q152␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q153␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q154␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q155␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q156␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q157␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q158␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q159␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q160␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q161␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;
```

```
␣␣␣␣␣␣␣signal␣sig_q162␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q163␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q164␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q165␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q166␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q167␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q168␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q169␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q170␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q171␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q172␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q173␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;
```

```
␣␣␣␣␣␣␣signal␣sig_q174␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q175␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q176␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q177␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q178␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q179␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q180␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q181␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q182␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q183␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q184␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q185␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;
```

```
␣␣␣␣␣␣␣signal␣sig_q186␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q187␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q188␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q189␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q190␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q191␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q192␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q193␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q194␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q195␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q196␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q197␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;
```

```
␣␣␣␣␣␣␣signal␣sig_q198␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q199␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q200␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q201␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q202␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q203␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q204␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q205␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q206␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q207␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q208␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q209␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;
```

```
␣␣␣␣␣␣␣signal␣sig_q210␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q211␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q212␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q213␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q214␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q215␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q216␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q217␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q218␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q219␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q220␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q221␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;
```

```
␣␣␣␣␣␣␣signal␣sig_q222␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q223␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q224␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q225␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q226␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q227␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q228␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q229␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q230␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q231␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q232␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q233␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;
```

```
␣␣␣␣␣␣␣signal␣sig_q234␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q235␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q236␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q237␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q238␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q239␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q240␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q241␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q242␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q243␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q244␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q245␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;
```

```
␣␣␣␣␣␣␣signal␣sig_q246␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q247␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q248␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q249␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q250␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q251␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q252␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q253␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q254␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;

␣␣␣␣␣␣␣signal␣sig_q255␣:␣␣␣␣␣␣std_logic_vector␣(N-1␣downto␣0)
    ;


BEGIN
␣␣␣␣␣␣␣PROCESS␣(clk,␣Reset,en_w_data,data_in,␣addr)
␣␣␣␣␣␣␣BEGIN
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣IF␣Reset␣=␣'1'␣THEN
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q000␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q001␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q002␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q003␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q004␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q005␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q006␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q007␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q008␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q009␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q010␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q011␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q012␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q013␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q014␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q015␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q016␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q017␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q018␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q019␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q020␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q021␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q022␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q023␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q024␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q025␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q026␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q027␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q028␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q029␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q030␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q031␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q032␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q033␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q034␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q035␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q036␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q037␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q038␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q039␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q040␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q041␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q042␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q043␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q044␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q045␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q046␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q047␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q048␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q049␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q050␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q051␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q052␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q053␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q054␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q055␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q056␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q057␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q058␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q059␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q060␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q061␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q062␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q063␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q064␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q065␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q066␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q067␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q068␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q069␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q070␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q071␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q072␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q073␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q074␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q075␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q076␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q077␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q078␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q079␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q080␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q081␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q082␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q083␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q084␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q085␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q086␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q087␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q088␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q089␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q090␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q091␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q092␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q093␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q094␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q095␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q096␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q097␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q098␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q099␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q100␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q101␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q102␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q103␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q104␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q105␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q106␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q107␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q108␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
```

```
                        sig_q109        <=  (OTHERS => '0');
                        sig_q110        <=  (OTHERS => '0');
                        sig_q111        <=  (OTHERS => '0');
                        sig_q112        <=  (OTHERS => '0');
                        sig_q113        <=  (OTHERS => '0');
                        sig_q114        <=  (OTHERS => '0');
                        sig_q115        <=  (OTHERS => '0');
                        sig_q116        <=  (OTHERS => '0');
                        sig_q117        <=  (OTHERS => '0');
                        sig_q118        <=  (OTHERS => '0');
                        sig_q119        <=  (OTHERS => '0');
                        sig_q120        <=  (OTHERS => '0');
                        sig_q121        <=  (OTHERS => '0');
                        sig_q122        <=  (OTHERS => '0');
                        sig_q123        <=  (OTHERS => '0');
                        sig_q124        <=  (OTHERS => '0');
                        sig_q125        <=  (OTHERS => '0');
                        sig_q126        <=  (OTHERS => '0');
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q127␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q128␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q129␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q130␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q131␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q132␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q133␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q134␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q135␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q136␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q137␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q138␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q139␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q140␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q141␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q142␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q143␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q144␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q145␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q146␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q147␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q148␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q149␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q150␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q151␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q152␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q153␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q154␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q155␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q156␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q157␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q158␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q159␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q160␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q161␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q162␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q163␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q164␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q165␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q166␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q167␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q168␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q169␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q170␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q171␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q172␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q173␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q174␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q175␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q176␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q177␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q178␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q179␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q180␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q181␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q182␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q183␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q184␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q185␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q186␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q187␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q188␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q189␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q190␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q191␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q192␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q193␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q194␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q195␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q196␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q197␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q198␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q199␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q200␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q201␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q202␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q203␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q204␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q205␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q206␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q207␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q208␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q209␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q210␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q211␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q212␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q213␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q214␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q215␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q216␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q217␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q218␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q219␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q220␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q221␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q222␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q223␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q224␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q225␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q226␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q227␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q228␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q229␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q230␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q231␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q232␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q233␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q234␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q235␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q236␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q237␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q238␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q239␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q240␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q241␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q242␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q243␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q244␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q245␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q246␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q247␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q248␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q249␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q250␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q251␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q252␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q253␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q254␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q255␣␣␣␣␣␣␣␣<=␣␣(OTHERS␣=>␣'0'␣);

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣ELSIF␣clk'EVENT␣AND␣clk␣=␣'1'␣THEN
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(en_w_data='1')␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣CASE␣(addr)␣is
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00000000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q000 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00000001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q001 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00000010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q002 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00000011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q003 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00000100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q004 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00000101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q005 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00000110\"␣=>
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q006 <=
   data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00000111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q007 <=
   data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00001000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q008 <=
   data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00001001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q009 <=
   data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00001010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q010 <=
   data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00001011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q011 <=
   data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00001100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q012 <=
   data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00001101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q013 <=
   data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00001110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q014 <=
   data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00001111\"␣=>
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q015 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00010000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q016 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00010001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q017 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00010010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q018 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00010011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q019 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00010100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q020 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00010101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q021 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00010110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q022 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00010111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q023 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00011000\"␣=>
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q024 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00011001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q025 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00011010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q026 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00011011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q027 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00011100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q028 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00011101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q029 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00011110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q030 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00011111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q031 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00100000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q032 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00100001\"␣=>
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q033 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00100010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q034 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00100011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q035 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00100100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q036 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00100101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q037 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00100110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q038 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00100111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q039 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00101000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q040 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00101001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q041 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00101010\"␣=>
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q042 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00101011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q043 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00101100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q044 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00101101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q045 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00101110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q046 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00101111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q047 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00110000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q048 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00110001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q049 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00110010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q050 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00110011\"␣=>
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q051 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00110100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q052 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00110101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q053 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00110110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q054 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00110111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q055 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00111000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q056 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00111001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q057 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00111010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q058 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00111011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q059 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00111100\"␣=>
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q060 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00111101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q061 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00111110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q062 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00111111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q063 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01000000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q064 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01000001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q065 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01000010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q066 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01000011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q067 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01000100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q068 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01000101\"␣=>
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q069 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01000110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q070 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01000111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q071 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01001000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q072 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01001001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q073 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01001010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q074 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01001011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q075 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01001100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q076 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01001101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q077 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01001110\"␣=>
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q078 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01001111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q079 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01010000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q080 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01010001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q081 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01010010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q082 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01010011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q083 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01010100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q084 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01010101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q085 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01010110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q086 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01010111\"␣=>
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q087 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01011000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q088 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01011001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q089 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01011010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q090 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01011011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q091 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01011100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q092 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01011101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q093 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01011110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q094 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01011111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q095 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01100000\"␣=>
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q096 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01100001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q097 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01100010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q098 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01100011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q099 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01100100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q100 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01100101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q101 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01100110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q102 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01100111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q103 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01101000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q104 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01101001\"␣=>
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q105 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01101010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q106 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01101011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q107 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01101100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q108 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01101101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q109 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01101110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q110 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01101111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q111 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01110000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q112 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01110001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q113 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01110010\"␣=>
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q114 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01110011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q115 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01110100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q116 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01110101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q117 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01110110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q118 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01110111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q119 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01111000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q120 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01111001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q121 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01111010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q122 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01111011\"␣=>
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q123 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01111100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q124 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01111101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q125 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01111110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q126 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01111111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q127 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10000000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q128 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10000001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q129 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10000010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q130 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10000011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q131 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10000100\"␣=>
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q132 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10000101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q133 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10000110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q134 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10000111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q135 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10001000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q136 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10001001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q137 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10001010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q138 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10001011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q139 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10001100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q140 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10001101\"␣=>
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q141 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10001110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q142 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10001111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q143 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10010000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q144 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10010001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q145 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10010010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q146 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10010011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q147 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10010100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q148 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10010101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q149 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10010110\"␣=>
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q150 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10010111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q151 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10011000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q152 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10011001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q153 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10011010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q154 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10011011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q155 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10011100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q156 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10011101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q157 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10011110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q158 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10011111\"␣=>
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q159 <=
   data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10100000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q160 <=
   data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10100001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q161 <=
   data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10100010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q162 <=
   data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10100011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q163 <=
   data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10100100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q164 <=
   data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10100101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q165 <=
   data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10100110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q166 <=
   data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10100111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q167 <=
   data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10101000\"␣=>
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q168 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10101001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q169 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10101010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q170 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10101011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q171 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10101100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q172 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10101101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q173 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10101110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q174 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10101111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q175 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10110000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q176 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10110001\"␣=>
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q177 <=
   data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10110010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q178 <=
   data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10110011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q179 <=
   data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10110100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q180 <=
   data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10110101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q181 <=
   data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10110110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q182 <=
   data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10110111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q183 <=
   data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10111000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q184 <=
   data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10111001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q185 <=
   data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10111010\"␣=>
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q186 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10111011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q187 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10111100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q188 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10111101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q189 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10111110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q190 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10111111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q191 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11000000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q192 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11000001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q193 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11000010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q194 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11000011\"␣=>
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q195 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11000100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q196 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11000101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q197 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11000110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q198 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11000111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q199 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11001000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q200 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11001001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q201 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11001010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q202 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11001011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q203 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11001100\"␣=>
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q204 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11001101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q205 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11001110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q206 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11001111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q207 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11010000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q208 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11010001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q209 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11010010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q210 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11010011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q211 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11010100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q212 <=
    data_in;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11010101\"␣=>
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q213 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11010110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q214 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11010111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q215 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11011000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q216 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11011001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q217 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11011010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q218 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11011011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q219 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11011100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q220 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11011101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q221 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11011110\"␣=>
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q222 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11011111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q223 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11100000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q224 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11100001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q225 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11100010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q226 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11100011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q227 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11100100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q228 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11100101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q229 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11100110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q230 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11100111\"␣=>
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q231 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11101000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q232 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11101001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q233 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11101010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q234 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11101011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q235 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11101100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q236 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11101101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q237 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11101110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q238 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11101111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q239 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11110000\"␣=>
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q240 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11110001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q241 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11110010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q242 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11110011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q243 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11110100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q244 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11110101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q245 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11110110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q246 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11110111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q247 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11111000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q248 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11111001\"␣=>
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q249 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11111010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q250 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11111011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q251 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11111100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q252 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11111101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q253 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11111110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q254 <=
    data_in ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11111111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q255 <=
    data_in ;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣others␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣NULL ;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣END␣CASE ;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end␣if ;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣END␣IF ;
␣␣␣␣␣␣␣␣END␣PROCESS ;

␣␣␣␣␣␣␣␣PROCESS␣(clk ,␣Reset , en_w_data , sig_q000 ,␣sig_q001 ,␣
    sig_q002 ,␣sig_q003 ,␣sig_q004 ,␣sig_q005 ,␣sig_q006 ,␣sig_q007 ,␣
    sig_q008 ,␣sig_q009 ,␣sig_q010 ,␣sig_q011 ,␣sig_q012 ,
```

␣␣␣␣␣␣␣␣sig_q013 ,␣sig_q014 ,␣sig_q015 ,␣sig_q016 ,␣sig_q017 ,␣
sig_q018 ,␣sig_q019 ,␣sig_q020 ,␣sig_q021 ,␣sig_q022 ,␣sig_q023 ,␣
sig_q024 ,␣sig_q025 ,␣sig_q026 ,␣sig_q027 ,␣sig_q028 ,
␣␣␣␣␣␣␣␣sig_q029 ,␣sig_q030 ,␣sig_q031 ,␣sig_q032 ,␣sig_q033 ,␣
sig_q034 ,␣sig_q035 ,␣sig_q036 ,␣sig_q037 ,␣sig_q038 ,␣sig_q039 ,␣
sig_q040 ,␣sig_q041 ,␣sig_q042 ,␣sig_q043 ,␣sig_q044 ,
␣␣␣␣␣␣␣␣sig_q045 ,␣sig_q046 ,␣sig_q047 ,␣sig_q048 ,␣sig_q049 ,␣
sig_q050 ,␣sig_q051 ,␣sig_q052 ,␣sig_q053 ,␣sig_q054 ,␣sig_q055 ,␣
sig_q056 ,␣sig_q057 ,␣sig_q058 ,␣sig_q059 ,␣sig_q060 ,
␣␣␣␣␣␣␣␣sig_q061 ,␣sig_q062 ,␣sig_q063 ,␣sig_q064 ,␣sig_q065 ,␣
sig_q066 ,␣sig_q067 ,␣sig_q068 ,␣sig_q069 ,␣sig_q070 ,␣sig_q071 ,␣
sig_q072 ,␣sig_q073 ,␣sig_q074 ,␣sig_q075 ,␣sig_q076 ,
␣␣␣␣␣␣␣␣sig_q077 ,␣sig_q078 ,␣sig_q079 ,␣sig_q080 ,␣sig_q081 ,␣
sig_q082 ,␣sig_q083 ,␣sig_q084 ,␣sig_q085 ,␣sig_q086 ,␣sig_q087 ,␣
sig_q088 ,␣sig_q089 ,␣sig_q090 ,␣sig_q091 ,␣sig_q092 ,
␣␣␣␣␣␣␣␣sig_q093 ,␣sig_q094 ,␣sig_q095 ,␣sig_q096 ,␣sig_q097 ,␣
sig_q098 ,␣sig_q099 ,␣sig_q100 ,␣sig_q101 ,␣sig_q102 ,␣sig_q103 ,␣
sig_q104 ,␣sig_q105 ,␣sig_q106 ,␣sig_q107 ,␣sig_q108 ,
␣␣␣␣␣␣␣␣sig_q109 ,␣sig_q110 ,␣sig_q111 ,␣sig_q112 ,␣sig_q113 ,␣
sig_q114 ,␣sig_q115 ,␣sig_q116 ,␣sig_q117 ,␣sig_q118 ,␣sig_q119 ,␣
sig_q120 ,␣sig_q121 ,␣sig_q122 ,␣sig_q123 ,␣sig_q124 ,
␣␣␣␣␣␣␣␣sig_q125 ,␣sig_q126 ,␣sig_q127 ,␣sig_q128 ,␣sig_q129 ,␣
sig_q130 ,␣sig_q131 ,␣sig_q132 ,␣sig_q133 ,␣sig_q134 ,␣sig_q135 ,␣
sig_q136 ,␣sig_q137 ,␣sig_q138 ,␣sig_q139 ,␣sig_q140 ,
␣␣␣␣␣␣␣␣sig_q141 ,␣sig_q142 ,␣sig_q143 ,␣sig_q144 ,␣sig_q145 ,␣
sig_q146 ,␣sig_q147 ,␣sig_q148 ,␣sig_q149 ,␣sig_q150 ,␣sig_q151 ,␣
sig_q152 ,␣sig_q153 ,␣sig_q154 ,␣sig_q155 ,␣sig_q156 ,
␣␣␣␣␣␣␣␣sig_q157 ,␣sig_q158 ,␣sig_q159 ,␣sig_q160 ,␣sig_q161 ,␣
sig_q162 ,␣sig_q163 ,␣sig_q164 ,␣sig_q165 ,␣sig_q166 ,␣sig_q167 ,␣
sig_q168 ,␣sig_q169 ,␣sig_q170 ,␣sig_q171 ,␣sig_q172 ,
␣␣␣␣␣␣␣␣sig_q173 ,␣sig_q174 ,␣sig_q175 ,␣sig_q176 ,␣sig_q177 ,␣
sig_q178 ,␣sig_q179 ,␣sig_q180 ,␣sig_q181 ,␣sig_q182 ,␣sig_q183 ,␣
sig_q184 ,␣sig_q185 ,␣sig_q186 ,␣sig_q187 ,␣sig_q188 ,
␣␣␣␣␣␣␣␣sig_q189 ,␣sig_q190 ,␣sig_q191 ,␣sig_q192 ,␣sig_q193 ,␣
sig_q194 ,␣sig_q195 ,␣sig_q196 ,␣sig_q197 ,␣sig_q198 ,␣sig_q199 ,␣
sig_q200 ,␣sig_q201 ,␣sig_q202 ,␣sig_q203 ,␣sig_q204 ,

```
␣␣␣␣␣␣␣␣sig_q205 ,␣sig_q206 ,␣sig_q207 ,␣sig_q208 ,␣sig_q209 ,␣
    sig_q210 ,␣sig_q211 ,␣sig_q212 ,␣sig_q213 ,␣sig_q214 ,␣sig_q215 ,␣
    sig_q216 ,␣sig_q217 ,␣sig_q218 ,␣sig_q219 ,␣sig_q220 ,
␣␣␣␣␣␣␣␣sig_q221 ,␣sig_q222 ,␣sig_q223 ,␣sig_q224 ,␣sig_q225 ,␣
    sig_q226 ,␣sig_q227 ,␣sig_q228 ,␣sig_q229 ,␣sig_q230 ,␣sig_q231 ,␣
    sig_q232 ,␣sig_q233 ,␣sig_q234 ,␣sig_q235 ,␣sig_q236 ,
␣␣␣␣␣␣␣␣sig_q237 ,␣sig_q238 ,␣sig_q239 ,␣sig_q240 ,␣sig_q241 ,␣
    sig_q242 ,␣sig_q243 ,␣sig_q244 ,␣sig_q245 ,␣sig_q246 ,␣sig_q247 ,␣
    sig_q248 ,␣sig_q249 ,␣sig_q250 ,␣sig_q251 ,␣sig_q252 ,
␣␣␣␣␣␣␣␣sig_q253 ,␣sig_q254 ,␣sig_q255␣)
␣␣␣␣␣␣␣␣BEGIN
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣IF␣clk 'EVENT␣AND␣clk␣=␣'1'␣THEN
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣CASE␣(addr)␣is
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00000000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q000 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00000001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q001 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00000010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q002 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00000011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q003 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00000100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q004 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00000101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q005 ;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00000110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q006 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00000111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q007 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00001000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q008 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00001001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q009 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00001010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q010 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00001011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q011 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00001100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q012 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00001101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q013 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00001110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q014 ;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00001111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q015;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00010000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q016;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00010001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q017;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00010010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q018;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00010011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q019;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00010100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q020;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00010101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q021;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00010110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q022;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00010111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q023;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00011000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q024;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00011001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q025;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00011010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q026;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00011011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q027;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00011100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q028;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00011101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q029;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00011110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q030;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00011111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q031;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00100000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q032;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00100001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q033 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00100010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q034 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00100011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q035 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00100100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q036 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00100101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q037 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00100110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q038 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00100111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q039 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00101000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q040 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00101001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q041 ;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00101010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q042;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00101011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q043;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00101100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q044;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00101101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q045;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00101110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q046;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00101111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q047;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00110000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q048;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00110001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q049;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00110010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q050;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00110011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q051 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00110100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q052 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00110101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q053 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00110110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q054 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00110111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q055 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00111000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q056 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00111001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q057 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00111010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q058 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00111011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q059 ;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00111100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q060 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00111101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q061 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00111110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q062 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"00111111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q063 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01000000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q064 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01000001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q065 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01000010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q066 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01000011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q067 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01000100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q068 ;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01000101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q069 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01000110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q070 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01000111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q071 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01001000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q072 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01001001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q073 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01001010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q074 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01001011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q075 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01001100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q076 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01001101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q077 ;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01001110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q078 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01001111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q079 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01010000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q080 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01010001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q081 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01010010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q082 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01010011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q083 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01010100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q084 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01010101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q085 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01010110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q086 ;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01010111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q087 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01011000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q088 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01011001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q089 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01011010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q090 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01011011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q091 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01011100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q092 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01011101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q093 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01011110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q094 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01011111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q095 ;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01100000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q096;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01100001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q097;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01100010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q098;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01100011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q099;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01100100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q100;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01100101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q101;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01100110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q102;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01100111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q103;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01101000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q104;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01101001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q105;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01101010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q106;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01101011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q107;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01101100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q108;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01101101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q109;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01101110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q110;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01101111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q111;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01110000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q112;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01110001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q113;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01110010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q114 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01110011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q115 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01110100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q116 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01110101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q117 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01110110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q118 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01110111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q119 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01111000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q120 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01111001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q121 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01111010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q122 ;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01111011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q123;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01111100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q124;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01111101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q125;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01111110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q126;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"01111111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q127;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10000000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q128;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10000001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q129;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10000010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q130;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10000011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q131;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10000100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q132 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10000101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q133 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10000110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q134 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10000111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q135 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10001000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q136 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10001001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q137 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10001010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q138 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10001011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q139 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10001100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q140 ;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10001101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q141 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10001110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q142 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10001111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q143 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10010000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q144 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10010001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q145 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10010010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q146 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10010011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q147 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10010100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q148 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10010101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q149 ;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10010110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q150;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10010111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q151;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10011000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q152;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10011001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q153;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10011010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q154;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10011011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q155;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10011100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q156;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10011101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q157;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10011110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q158;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10011111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q159 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10100000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q160 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10100001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q161 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10100010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q162 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10100011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q163 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10100100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q164 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10100101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q165 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10100110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q166 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10100111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q167 ;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10101000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q168 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10101001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q169 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10101010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q170 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10101011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q171 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10101100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q172 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10101101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q173 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10101110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q174 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10101111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q175 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10110000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q176 ;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10110001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q177 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10110010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q178 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10110011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q179 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10110100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q180 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10110101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q181 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10110110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q182 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10110111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q183 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10111000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q184 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10111001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q185 ;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10111010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q186 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10111011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q187 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10111100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q188 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10111101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q189 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10111110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q190 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"10111111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q191 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11000000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q192 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11000001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q193 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11000010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q194 ;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11000011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
   sig_q195;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11000100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
   sig_q196;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11000101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
   sig_q197;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11000110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
   sig_q198;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11000111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
   sig_q199;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11001000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
   sig_q200;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11001001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
   sig_q201;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11001010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
   sig_q202;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11001011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
   sig_q203;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11001100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q204 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11001101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q205 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11001110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q206 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11001111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q207 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11010000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q208 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11010001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q209 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11010010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q210 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11010011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q211 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11010100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q212 ;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11010101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
   sig_q213 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11010110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
   sig_q214 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11010111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
   sig_q215 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11011000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
   sig_q216 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11011001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
   sig_q217 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11011010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
   sig_q218 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11011011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
   sig_q219 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11011100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
   sig_q220 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11011101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
   sig_q221 ;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11011110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q222 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11011111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q223 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11100000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q224 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11100001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q225 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11100010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q226 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11100011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q227 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11100100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q228 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11100101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q229 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11100110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q230 ;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11100111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q231 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11101000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q232 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11101001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q233 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11101010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q234 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11101011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q235 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11101100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q236 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11101101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q237 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11101110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q238 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11101111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q239 ;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11110000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q240 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11110001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q241 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11110010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q242 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11110011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q243 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11110100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q244 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11110101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q245 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11110110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q246 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11110111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q247 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11111000\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q248 ;
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11111001\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q249 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11111010\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q250 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11111011\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q251 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11111100\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q252 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11111101\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q253 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11111110\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q254 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣\"11111111\"␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out <=
    sig_q255 ;

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣when␣others␣=>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_out␣<=␣␣(OTHERS␣=>
    ␣'0'␣);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣END␣CASE;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣END␣IF;
␣␣␣␣␣␣␣␣END␣PROCESS;
```

```
END␣BHV_BUFFER_BLOCK;

";
if ($grava){
                gravar("mem.vhd", $conteudo_mem);
                gravar("buffer_block.vhd", $conteudo_buf);
        }
}

function mem_altera($gerar_tb, $nro_canal,$grava,$tela){

$conteudo="
--␣megafunction␣wizard:␣%RAM:␣1-PORT%
--␣GENERATION:␣STANDARD
--␣VERSION:␣WM1.0
--␣MODULE:␣altsyncram

--␣============================================================
--␣File␣Name:␣mem.vhd
--␣Megafunction␣Name(s):
--␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣altsyncram
--
--␣Simulation␣Library␣Files(s):
--␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣altera_mf
--␣============================================================
--␣************************************************************
--␣THIS␣IS␣A␣WIZARD-GENERATED␣FILE.␣DO␣NOT␣EDIT␣THIS␣FILE!
--
--␣10.1␣Build␣153␣11/29/2010␣SJ␣Web␣Edition
--␣************************************************************


--Copyright␣(C)␣1991-2010␣Altera␣Corporation
--Your␣use␣of␣Altera␣Corporation's␣design␣tools,␣logic␣
    functions
```

```
--and␣other␣software␣and␣tools,␣and␣its␣AMPP␣partner␣logic
--functions,␣and␣any␣output␣files␣from␣any␣of␣the␣foregoing
--(including␣device␣programming␣or␣simulation␣files),␣and␣any
--associated␣documentation␣or␣information␣are␣expressly␣subject
--to␣the␣terms␣and␣conditions␣of␣the␣Altera␣Program␣License
--Subscription␣Agreement,␣Altera␣MegaCore␣Function␣License
--Agreement,␣or␣other␣applicable␣license␣agreement,␣including,
--without␣limitation,␣that␣your␣use␣is␣for␣the␣sole␣purpose␣of
--programming␣logic␣devices␣manufactured␣by␣Altera␣and␣sold␣by
--Altera␣or␣its␣authorized␣distributors.␣␣Please␣refer␣to␣the
--applicable␣agreement␣for␣further␣details.


LIBRARY␣ieee;
USE␣ieee.std_logic_1164.all;

LIBRARY␣altera_mf;
USE␣altera_mf.all;

ENTITY␣mem␣IS
␣␣␣␣␣␣␣␣PORT
␣␣␣␣␣␣␣␣(
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣address␣:␣IN␣STD_LOGIC_VECTOR␣(7␣DOWNTO␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣clock␣␣␣␣␣␣␣␣␣␣␣:␣IN␣STD_LOGIC␣␣:=␣'1';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data␣␣␣␣␣␣␣␣␣␣␣␣:␣IN␣STD_LOGIC_VECTOR␣(31␣
    DOWNTO␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wren␣␣␣␣␣␣␣␣␣␣␣␣:␣IN␣STD_LOGIC␣;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣q␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣OUT␣STD_LOGIC_VECTOR␣
    (31␣DOWNTO␣0)
␣␣␣␣␣␣␣␣);
END␣mem;


ARCHITECTURE␣SYN␣OF␣mem␣IS
```

```
␣␣␣␣␣␣␣␣SIGNAL␣sub_wire0␣␣␣␣␣␣␣␣:␣STD_LOGIC_VECTOR␣(31␣DOWNTO␣
    0);



␣␣␣␣␣␣␣␣COMPONENT␣altsyncram
␣␣␣␣␣␣␣␣GENERIC␣(
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣clock_enable_input_a␣␣␣␣␣␣␣␣␣␣␣:␣STRING;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣clock_enable_output_a␣␣␣␣␣␣␣␣␣␣:␣STRING;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣intended_device_family␣␣␣␣␣␣␣␣␣:␣STRING;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣lpm_hint␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣STRING;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣lpm_type␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣STRING;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣numwords_a␣␣␣␣␣␣␣␣␣␣␣␣␣:␣NATURAL;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣operation_mode␣␣␣␣␣␣␣␣␣:␣STRING;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣outdata_aclr_a␣␣␣␣␣␣␣␣␣:␣STRING;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣outdata_reg_a␣␣␣␣␣␣␣␣␣␣:␣STRING;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣power_up_uninitialized␣␣␣␣␣␣␣␣␣:␣STRING;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣widthad_a␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣NATURAL;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣width_a␣␣␣␣␣␣␣␣␣:␣NATURAL;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣width_byteena_a␣␣␣␣␣␣␣␣␣:␣NATURAL
␣␣␣␣␣␣␣␣);
␣␣␣␣␣␣␣␣PORT␣(
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣address_a␣␣␣␣␣␣␣:␣IN␣STD_LOGIC_VECTOR␣
    (7␣DOWNTO␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣clock0␣␣:␣IN␣STD_LOGIC␣;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data_a␣␣:␣IN␣STD_LOGIC_VECTOR␣(31␣
    DOWNTO␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wren_a␣␣:␣IN␣STD_LOGIC␣;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣q_a␣␣␣␣␣:␣OUT␣STD_LOGIC_VECTOR␣(31␣
    DOWNTO␣0)
␣␣␣␣␣␣␣␣);
␣␣␣␣␣␣␣␣END␣COMPONENT;

BEGIN
␣␣␣␣␣␣␣␣q␣␣␣␣<=␣sub_wire0(31␣DOWNTO␣0);
```

```
        altsyncram_component : altsyncram
        GENERIC MAP (
                clock_enable_input_a => \"BYPASS\",
                clock_enable_output_a => \"BYPASS\",
                intended_device_family => \"Cyclone II\",
                lpm_hint => \"ENABLE_RUNTIME_MOD=NO\",
                lpm_type => \"altsyncram\",
                numwords_a => 256,
                operation_mode => \"SINGLE_PORT\",
                outdata_aclr_a => \"NONE\",
                outdata_reg_a => \"CLOCK0\",
                power_up_uninitialized => \"FALSE\",
                widthad_a => 8,
                width_a => 32,
                width_byteena_a => 1
        )
        PORT MAP (
                address_a => address,
                clock0 => clock,
                data_a => data,
                wren_a => wren,
                q_a => sub_wire0
        );



END SYN;

-- ============================================================
-- CNX file retrieval info
-- ============================================================
-- Retrieval info: PRIVATE: ADDRESSSTALL_A NUMERIC \"0\"
-- Retrieval info: PRIVATE: AclrAddr NUMERIC \"0\"
-- Retrieval info: PRIVATE: AclrByte NUMERIC \"0\"
-- Retrieval info: PRIVATE: AclrData NUMERIC \"0\"
-- Retrieval info: PRIVATE: AclrOutput NUMERIC \"0\"
```

```
--␣Retrieval␣info:␣PRIVATE:␣BYTE_ENABLE␣NUMERIC␣\"0\"
--␣Retrieval␣info:␣PRIVATE:␣BYTE_SIZE␣NUMERIC␣\"8\"
--␣Retrieval␣info:␣PRIVATE:␣BlankMemory␣NUMERIC␣\"1\"
--␣Retrieval␣info:␣PRIVATE:␣CLOCK_ENABLE_INPUT_A␣NUMERIC␣\"0\"
--␣Retrieval␣info:␣PRIVATE:␣CLOCK_ENABLE_OUTPUT_A␣NUMERIC␣\"0\"
--␣Retrieval␣info:␣PRIVATE:␣Clken␣NUMERIC␣\"0\"
--␣Retrieval␣info:␣PRIVATE:␣DataBusSeparated␣NUMERIC␣\"1\"
--␣Retrieval␣info:␣PRIVATE:␣IMPLEMENT_IN_LES␣NUMERIC␣\"0\"
--␣Retrieval␣info:␣PRIVATE:␣INIT_FILE_LAYOUT␣STRING␣\"PORT_A\"
--␣Retrieval␣info:␣PRIVATE:␣INIT_TO_SIM_X␣NUMERIC␣\"0\"
--␣Retrieval␣info:␣PRIVATE:␣INTENDED_DEVICE_FAMILY␣STRING␣\"
    Cyclone␣II\"
--␣Retrieval␣info:␣PRIVATE:␣JTAG_ENABLED␣NUMERIC␣\"0\"
--␣Retrieval␣info:␣PRIVATE:␣JTAG_ID␣STRING␣\"NONE\"
--␣Retrieval␣info:␣PRIVATE:␣MAXIMUM_DEPTH␣NUMERIC␣\"0\"
--␣Retrieval␣info:␣PRIVATE:␣MIFfilename␣STRING␣\"\"
--␣Retrieval␣info:␣PRIVATE:␣NUMWORDS_A␣NUMERIC␣\"256\"
--␣Retrieval␣info:␣PRIVATE:␣RAM_BLOCK_TYPE␣NUMERIC␣\"0\"
--␣Retrieval␣info:␣PRIVATE:␣READ_DURING_WRITE_MODE_PORT_A␣
    NUMERIC␣\"3\"
--␣Retrieval␣info:␣PRIVATE:␣RegAddr␣NUMERIC␣\"1\"
--␣Retrieval␣info:␣PRIVATE:␣RegData␣NUMERIC␣\"1\"
--␣Retrieval␣info:␣PRIVATE:␣RegOutput␣NUMERIC␣\"1\"
--␣Retrieval␣info:␣PRIVATE:␣SYNTH_WRAPPER_GEN_POSTFIX␣STRING␣
    \"0\"
--␣Retrieval␣info:␣PRIVATE:␣SingleClock␣NUMERIC␣\"1\"
--␣Retrieval␣info:␣PRIVATE:␣UseDQRAM␣NUMERIC␣\"1\"
--␣Retrieval␣info:␣PRIVATE:␣WRCONTROL_ACLR_A␣NUMERIC␣\"0\"
--␣Retrieval␣info:␣PRIVATE:␣WidthAddr␣NUMERIC␣\"8\"
--␣Retrieval␣info:␣PRIVATE:␣WidthData␣NUMERIC␣\"32\"
--␣Retrieval␣info:␣PRIVATE:␣rden␣NUMERIC␣\"0\"
--␣Retrieval␣info:␣LIBRARY:␣altera_mf␣altera_mf.
    altera_mf_components.all
--␣Retrieval␣info:␣CONSTANT:␣CLOCK_ENABLE_INPUT_A␣STRING␣\"
    BYPASS\"
```

```
--␣Retrieval␣info:␣CONSTANT:␣CLOCK_ENABLE_OUTPUT_A␣STRING␣\"
    BYPASS\"
--␣Retrieval␣info:␣CONSTANT:␣INTENDED_DEVICE_FAMILY␣STRING␣\"
    Cyclone␣II\"
--␣Retrieval␣info:␣CONSTANT:␣LPM_HINT␣STRING␣\"
    ENABLE_RUNTIME_MOD=NO\"
--␣Retrieval␣info:␣CONSTANT:␣LPM_TYPE␣STRING␣\"altsyncram\"
--␣Retrieval␣info:␣CONSTANT:␣NUMWORDS_A␣NUMERIC␣\"256\"
--␣Retrieval␣info:␣CONSTANT:␣OPERATION_MODE␣STRING␣\"
    SINGLE_PORT\"
--␣Retrieval␣info:␣CONSTANT:␣OUTDATA_ACLR_A␣STRING␣\"NONE\"
--␣Retrieval␣info:␣CONSTANT:␣OUTDATA_REG_A␣STRING␣\"CLOCK0\"
--␣Retrieval␣info:␣CONSTANT:␣POWER_UP_UNINITIALIZED␣STRING␣\"
    FALSE\"
--␣Retrieval␣info:␣CONSTANT:␣WIDTHAD_A␣NUMERIC␣\"8\"
--␣Retrieval␣info:␣CONSTANT:␣WIDTH_A␣NUMERIC␣\"32\"
--␣Retrieval␣info:␣CONSTANT:␣WIDTH_BYTEENA_A␣NUMERIC␣\"1\"
--␣Retrieval␣info:␣USED_PORT:␣address␣0␣0␣8␣0␣INPUT␣NODEFVAL␣\"
    address[7..0]\"
--␣Retrieval␣info:␣USED_PORT:␣clock␣0␣0␣0␣0␣INPUT␣VCC␣\"clock\"
--␣Retrieval␣info:␣USED_PORT:␣data␣0␣0␣32␣0␣INPUT␣NODEFVAL␣\"
    data[31..0]\"
--␣Retrieval␣info:␣USED_PORT:␣q␣0␣0␣32␣0␣OUTPUT␣NODEFVAL␣\"q
    [31..0]\"
--␣Retrieval␣info:␣USED_PORT:␣wren␣0␣0␣0␣0␣INPUT␣NODEFVAL␣\"
    wren\"
--␣Retrieval␣info:␣CONNECT:␣@address_a␣0␣0␣8␣0␣address␣0␣0␣8␣0
--␣Retrieval␣info:␣CONNECT:␣@clock0␣0␣0␣0␣0␣clock␣0␣0␣0␣0
--␣Retrieval␣info:␣CONNECT:␣@data_a␣0␣0␣32␣0␣data␣0␣0␣32␣0
--␣Retrieval␣info:␣CONNECT:␣@wren_a␣0␣0␣0␣0␣wren␣0␣0␣0␣0
--␣Retrieval␣info:␣CONNECT:␣q␣0␣0␣32␣0␣@q_a␣0␣0␣32␣0
--␣Retrieval␣info:␣GEN_FILE:␣TYPE_NORMAL␣mem.vhd␣TRUE
--␣Retrieval␣info:␣GEN_FILE:␣TYPE_NORMAL␣mem.inc␣FALSE
--␣Retrieval␣info:␣GEN_FILE:␣TYPE_NORMAL␣mem.cmp␣TRUE
--␣Retrieval␣info:␣GEN_FILE:␣TYPE_NORMAL␣mem.bsf␣FALSE
--␣Retrieval␣info:␣GEN_FILE:␣TYPE_NORMAL␣mem_inst.vhd␣FALSE
```

```
--␣Retrieval␣info:␣LIB_FILE:␣altera_mf

";

        if ($grava){
                gravar("mem.vhd", $conteudo);
                }
        if ($tela){
                tela("mem.vhd", $conteudo);
                }
$conteudo_tb="

library␣ieee;
use␣ieee.std_logic_1164.all;
use␣ieee.std_logic_arith.all;

entity␣MEM_TB␣is
end␣MEM_TB;

architecture␣TB_MEM␣of␣MEM_TB␣is

␣␣␣␣component␣MEM
␣␣␣␣␣␣␣␣port(
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣clock␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣IN␣STD_LOGIC␣␣:=␣'1';
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣address␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣IN␣std_logic_vector␣
    (7␣DOWNTO␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣data␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣IN␣std_logic_vector␣
    (31␣DOWNTO␣0);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣wren␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣IN␣STD_LOGIC␣;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣q␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣OUT␣
    std_logic_vector␣(31␣DOWNTO␣0)␣␣␣␣);
␣␣␣␣end␣component;

␣␣␣␣--␣sinais␣necessarios␣para␣realizacao␣do␣TB

␣␣␣␣␣␣␣␣signal␣␣sig_clock␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣STD_LOGIC;
```

```
␣␣␣␣␣␣␣signal␣␣sig_address␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣
    std_logic_vector␣(7␣DOWNTO␣0);
␣␣␣␣␣␣␣signal␣␣sig_data␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣
    std_logic_vector␣(31␣DOWNTO␣0);
␣␣␣␣␣␣␣signal␣␣sig_wren␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣STD_LOGIC␣;
␣␣␣␣␣␣␣signal␣␣sig_q␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣:␣
    std_logic_vector␣(31␣DOWNTO␣0);

begin
--Testes␣desejados:
--␣␣␣␣␣␣*␣Testar␣sinais␣quando␣endereco␣de␣entrada␣nao␣
    corresponde␣ao␣mapeado␣ao␣dmac
--␣␣␣␣␣␣␣␣␣␣␣␣␣␣esperado:␣nenhuma␣alteracao␣nos␣sinais
--␣␣␣␣␣␣*␣Testar␣sinais␣quando␣endereco␣de␣entrada␣corresponde␣
    ao␣mapeado␣ao␣dmac
--␣␣␣␣␣␣␣␣␣␣␣␣␣␣esperado:␣en_MEM␣habilitar␣e␣demais␣sinais␣de␣
    acordo␣com␣o␣endereco
--␣␣␣␣␣␣␣␣␣␣␣␣␣+␣Testar␣cada␣mudanca␣de␣sinal
--␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣esperado:␣cada␣sinal␣reagir␣de␣acordo


--␣Instanciamento␣do␣decodificador
␣␣␣␣␣␣␣databuffer␣:␣mem␣port␣map
␣␣␣␣␣␣␣(␣␣␣␣␣␣␣sig_clock,␣␣␣␣␣␣␣␣␣␣␣␣␣sig_address,␣␣␣␣
    sig_data,
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_wren,␣␣␣␣␣␣␣␣␣␣␣␣␣␣sig_q
␣␣␣␣␣␣␣);


--␣geracao␣do␣clock
process
begin
␣␣␣␣␣␣␣sig_clock␣<=␣'0';
␣␣␣␣␣␣␣wait␣for␣5␣ns;
␣␣␣␣␣␣␣sig_clock␣<=␣'1';
␣␣␣␣␣␣␣wait␣for␣5␣ns;
```

```
end␣process;

process
begin
␣␣␣␣␣␣␣␣sig_address␣␣␣␣␣<=␣X\"00\";
␣␣␣␣␣␣␣␣sig_data␣␣␣␣␣␣␣␣<=X\"00000000\";
␣␣␣␣␣␣␣␣sig_wren␣␣␣␣␣␣␣␣<='0';
␣␣␣␣␣␣␣␣wait␣for␣10␣ns;
␣␣␣␣␣␣␣␣sig_address␣␣␣␣␣<=␣X\"00\";
␣␣␣␣␣␣␣␣sig_data␣␣␣␣␣␣␣␣<=X\"FFFFFFFF\";
␣␣␣␣␣␣␣␣sig_wren␣␣␣␣␣␣␣␣<='1';
␣␣␣␣␣␣␣␣wait␣for␣10␣ns;
␣␣␣␣␣␣␣␣sig_address␣␣␣␣␣<=␣X\"01\";
␣␣␣␣␣␣␣␣sig_data␣␣␣␣␣␣␣␣<=X\"FFFFFFFE\";
␣␣␣␣␣␣␣␣sig_wren␣␣␣␣␣␣␣␣<='1';
␣␣␣␣␣␣␣␣wait␣for␣10␣ns;
␣␣␣␣␣␣␣␣sig_address␣␣␣␣␣<=␣X\"02\";
␣␣␣␣␣␣␣␣sig_data␣␣␣␣␣␣␣␣<=X\"FFFFFFFD\";
␣␣␣␣␣␣␣␣sig_wren␣␣␣␣␣␣␣␣<='1';
␣␣␣␣␣␣␣␣wait␣for␣10␣ns;
␣␣␣␣␣␣␣␣sig_address␣␣␣␣␣<=␣X\"00\";
␣␣␣␣␣␣␣␣sig_data␣␣␣␣␣␣␣␣<=X\"00000000\";
␣␣␣␣␣␣␣␣sig_wren␣␣␣␣␣␣␣␣<='0';
␣␣␣␣␣␣␣␣wait␣for␣10␣ns;
␣␣␣␣␣␣␣␣sig_address␣␣␣␣␣<=␣X\"00\";
␣␣␣␣␣␣␣␣sig_data␣␣␣␣␣␣␣␣<=X\"00000001\";
␣␣␣␣␣␣␣␣sig_wren␣␣␣␣␣␣␣␣<='0';
␣␣␣␣␣␣␣␣wait␣for␣10␣ns;
␣␣␣␣␣␣␣␣sig_address␣␣␣␣␣<=␣X\"00\";
␣␣␣␣␣␣␣␣sig_data␣␣␣␣␣␣␣␣<=X\"00000002\";
␣␣␣␣␣␣␣␣sig_wren␣␣␣␣␣␣␣␣<='0';
␣␣␣␣␣␣␣␣wait␣for␣10␣ns;

--␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣(sig_r_status='1')␣then
--␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assert␣false␣report␣\"caso␣5:␣
    r_status:␣ok\"␣severity␣note;
```

```
--                    else
--                              assert false report \"caso 5:
    r_status: fail\" severity error;
--                    end if;
        -- fim teste caso 5

        wait for 5 ns;
        assert false report \"Testbench: MEM concluido\"
    severity note;
        wait;
end process;

end TB_MEM;
```

```
configuration CFG_TB_MEM of MEM_TB is
    for TB_MEM
    end for;
end CFG_TB_MEM;
```

```
";
        if ($gerar_tb){
                if ($grava){
                        gravar("tb_mem.vhd", $conteudo);
                        }
                if ($tela){
                        tela("tb_mem.vhd", $conteudo);
                        }
        }
}
function dmac_debug($gerar_tb, $nro_canal,$grava,$tela){

$conteudo_tb="
```

```
";
if ($gerar_tb){
        if ($grava){
                gravar("tb_dmac_debug.vhd", $conteudo);
                }
        if ($tela){
                tela("tb_dmac_debug.vhd", $conteudo);
                }
}
}
function projeto_altera($gerar_tb, $nro_canal,$grava,$tela,
   $memoria){

$conteudo="
#␣
   ____________________________________________________________________
   ␣#
#
#␣Copyright␣(C)␣1991-2010␣Altera␣Corporation
#␣Your␣use␣of␣Altera␣Corporation's␣design␣tools,␣logic␣
   functions
#␣and␣other␣software␣and␣tools,␣and␣its␣AMPP␣partner␣logic
#␣functions,␣and␣any␣output␣files␣from␣any␣of␣the␣foregoing
#␣(including␣device␣programming␣or␣simulation␣files),␣and␣any
#␣associated␣documentation␣or␣information␣are␣expressly␣subject
#␣to␣the␣terms␣and␣conditions␣of␣the␣Altera␣Program␣License
#␣Subscription␣Agreement,␣Altera␣MegaCore␣Function␣License
#␣Agreement,␣or␣other␣applicable␣license␣agreement,␣including,
#␣without␣limitation,␣that␣your␣use␣is␣for␣the␣sole␣purpose␣of
#␣programming␣logic␣devices␣manufactured␣by␣Altera␣and␣sold␣by
#␣Altera␣or␣its␣authorized␣distributors.␣␣Please␣refer␣to␣the
#␣applicable␣agreement␣for␣further␣details.
#
#␣
   ____________________________________________________________________
```

```
␣#
#
#␣Quartus␣II
#␣Version␣10.1␣Build␣153␣11/29/2010␣SJ␣Web␣Edition
#␣Date␣created␣=␣13:09:02␣␣May␣04,␣2011
#
#␣
    ____________________________________________________________________________
    ␣#
#
#␣Notes:
#
#␣1)␣The␣default␣values␣for␣assignments␣are␣stored␣in␣the␣file:
#␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣dmac_assignment_defaults.qdf
#␣␣␣␣If␣this␣file␣doesn't␣exist,␣see␣file:
#␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣assignment_defaults.qdf
#
#␣2)␣Altera␣recommends␣that␣you␣do␣not␣modify␣this␣file.␣This
#␣␣␣␣file␣is␣updated␣automatically␣by␣the␣Quartus␣II␣software
#␣␣␣␣and␣any␣changes␣you␣make␣may␣be␣lost␣or␣overwritten.
#
#␣
    ____________________________________________________________________________
    ␣#


set_global_assignment␣-name␣FAMILY␣\"Cyclone␣II\"
set_global_assignment␣-name␣DEVICE␣EP2C35F672C6
set_global_assignment␣-name␣TOP_LEVEL_ENTITY␣dmac
set_global_assignment␣-name␣ORIGINAL_QUARTUS_VERSION␣10.1
set_global_assignment␣-name␣PROJECT_CREATION_TIME_DATE␣
    \"13:09:02␣␣MAY␣04,␣2011\"
set_global_assignment␣-name␣LAST_QUARTUS_VERSION␣10.1
set_global_assignment␣-name␣MIN_CORE_JUNCTION_TEMP␣0
set_global_assignment␣-name␣MAX_CORE_JUNCTION_TEMP␣85
set_global_assignment␣-name␣ERROR_CHECK_FREQUENCY_DIVISOR␣1
```

```
";
if ($memoria!="altera")
        $conteudo.="set_global_assignment␣-name␣VHDL_FILE␣
            buffer_block.vhd";
$conteudo.="
set_global_assignment␣-name␣VHDL_FILE␣state_machine.vhd
set_global_assignment␣-name␣VHDL_FILE␣mem.vhd
set_global_assignment␣-name␣VHDL_FILE␣main_operation.vhd
set_global_assignment␣-name␣VHDL_FILE␣channel_data.vhd
set_global_assignment␣-name␣VHDL_FILE␣reg.vhd
set_global_assignment␣-name␣VHDL_FILE␣decoder.vhd
set_global_assignment␣-name␣STRATIX_DEVICE_IO_STANDARD␣\"3.3-V␣
   LVTTL\"
set_global_assignment␣-name␣MISC_FILE␣\"dmac.dpf\"
set_global_assignment␣-name␣VHDL_FILE␣dmac.vhd
set_global_assignment␣-name␣PARTITION_NETLIST_TYPE␣SOURCE␣-
   section_id␣Top
set_global_assignment␣-name␣PARTITION_FITTER_PRESERVATION_LEVEL
   ␣PLACEMENT_AND_ROUTING␣-section_id␣Top
set_global_assignment␣-name␣PARTITION_COLOR␣16764057␣-
   section_id␣Top
set_instance_assignment␣-name␣PARTITION_HIERARCHY␣
   root_partition␣-to␣|␣-section_id␣Top
";
$conteudo_tb="
#␣
   ------------------------------------------------------------------------
   ␣#
#
#␣Copyright␣(C)␣1991-2010␣Altera␣Corporation
#␣Your␣use␣of␣Altera␣Corporation's␣design␣tools,␣logic␣
   functions
#␣and␣other␣software␣and␣tools,␣and␣its␣AMPP␣partner␣logic
#␣functions,␣and␣any␣output␣files␣from␣any␣of␣the␣foregoing
#␣(including␣device␣programming␣or␣simulation␣files),␣and␣any
#␣associated␣documentation␣or␣information␣are␣expressly␣subject
```

```
#␣to␣the␣terms␣and␣conditions␣of␣the␣Altera␣Program␣License
#␣Subscription␣Agreement,␣Altera␣MegaCore␣Function␣License
#␣Agreement,␣or␣other␣applicable␣license␣agreement,␣including,
#␣without␣limitation,␣that␣your␣use␣is␣for␣the␣sole␣purpose␣of
#␣programming␣logic␣devices␣manufactured␣by␣Altera␣and␣sold␣by
#␣Altera␣or␣its␣authorized␣distributors.␣␣Please␣refer␣to␣the
#␣applicable␣agreement␣for␣further␣details.
#
#␣
    ____________________________________________________________________
    ␣#
#
#␣Quartus␣II
#␣Version␣10.1␣Build␣153␣11/29/2010␣SJ␣Web␣Edition
#␣Date␣created␣=␣13:09:02␣␣May␣04,␣2011
#
#␣
    ____________________________________________________________________
    ␣#

QUARTUS_VERSION␣=␣\"10.1\"
DATE␣=␣\"13:09:02␣␣May␣04,␣2011\"

#␣Revisions

PROJECT_REVISION␣=␣\"dmac\"

";
$conteudo_2="
#*************************************************************
#␣THIS␣IS␣A␣WIZARD-GENERATED␣FILE.
#
#␣Version␣10.1␣Build␣153␣11/29/2010␣SJ␣Web␣Edition
#
#*************************************************************
```

```
# Copyright (C) 1991-2010 Altera Corporation
# Your use of Altera Corporation's design tools, logic
    functions
# and other software and tools, and its AMPP partner logic
# functions, and any output files from any of the foregoing
# (including device programming or simulation files), and any
# associated documentation or information are expressly subject
# to the terms and conditions of the Altera Program License
# Subscription Agreement, Altera MegaCore Function License
# Agreement, or other applicable license agreement, including,
# without limitation, that your use is for the sole purpose of
# programming logic devices manufactured by Altera and sold by
# Altera or its authorized distributors.  Please refer to the
# applicable agreement for further details.



# Clock constraints

create_clock -name \"clk\" -period 20.000ns [get_ports {clk}] -
    waveform {0 10}


# Automatically constrain PLL and other generated clocks
derive_pll_clocks -create_base_clocks

# Automatically calculate clock uncertainty to jitter and other
    effects.
#derive_clock_uncertainty
# Not supported for family Cyclone II

# tsu/th constraints

set_input_delay -clock \"clk\" -max 1.000ps [get_ports {dreq
    [0]}]
```

```
set_input_delay␣-clock␣\"clk\"␣-min␣1.000ps␣[get_ports␣{dreq
    [0]}]
";
for ($i=1;$i<$nro_canal;$i++){
        $conteudo_2.="
set_input_delay␣-clock␣\"clk\"␣-max␣1.000ps␣[get_ports␣{dreq[$i
    ]}]
set_input_delay␣-clock␣\"clk\"␣-min␣1.000ps␣[get_ports␣{dreq[$i
    ]}]
";
}
$conteudo_2.="

set_input_delay␣-clock␣\"clk\"␣-max␣1.000ps␣[get_ports␣{hlda}]
set_input_delay␣-clock␣\"clk\"␣-min␣1.000ps␣[get_ports␣{hlda}]

set_input_delay␣-clock␣\"clk\"␣-max␣1.000ps␣[get_ports␣{ready}]
set_input_delay␣-clock␣\"clk\"␣-min␣1.000ps␣[get_ports␣{ready}]

set_input_delay␣-clock␣\"clk\"␣-max␣1.000ps␣[get_ports␣{reset}]
set_input_delay␣-clock␣\"clk\"␣-min␣1.000ps␣[get_ports␣{reset}]

#␣tco␣constraints

set_output_delay␣-clock␣\"clk\"␣-max␣1.000ps␣[get_ports␣{dack
    [0]}]
set_output_delay␣-clock␣\"clk\"␣-min␣-1.000ps␣[get_ports␣{dack
    [0]}]
";
for ($i=1;$i<$nro_canal;$i++){
        $conteudo_2.="
set_output_delay␣-clock␣\"clk\"␣-max␣1.000ps␣[get_ports␣{dack[
    $i]}]
set_output_delay␣-clock␣\"clk\"␣-min␣-1.000ps␣[get_ports␣{dack[
    $i]}]
";
```

```
}
$conteudo_2.="

set_output_delay␣-clock␣\"clk\"␣-max␣␣1.000ps␣[get_ports␣{hold
    }]
set_output_delay␣-clock␣\"clk\"␣-min␣-1.000ps␣[get_ports␣(hold
    }]

set_output_delay␣-clock␣\"clk\"␣-max␣␣1.000ps␣[get_ports␣{eop}]
set_output_delay␣-clock␣\"clk\"␣-min␣-1.000ps␣[get_ports␣{eop}]

set_output_delay␣-clock␣\"clk\"␣-max␣␣1.000ps␣[get_ports␣{io_r
    }]
set_output_delay␣-clock␣\"clk\"␣-min␣-1.000ps␣[get_ports␣{io_r
    }]

set_output_delay␣-clock␣\"clk\"␣-max␣␣1.000ps␣[get_ports␣{io_w
    }]
set_output_delay␣-clock␣\"clk\"␣-min␣-1.000ps␣[get_ports␣{io_w
    }]

set_output_delay␣-clock␣\"clk\"␣-max␣␣1.000ps␣[get_ports␣{mem_r
    }]
set_output_delay␣-clock␣\"clk\"␣-min␣-1.000ps␣[get_ports␣{mem_r
    }]

set_output_delay␣-clock␣\"clk\"␣-max␣␣1.000ps␣[get_ports␣{mem_w
    }]
set_output_delay␣-clock␣\"clk\"␣-min␣-1.000ps␣[get_ports␣{mem_w
    }]



#␣tpd␣constraints
```

```
";
                gravar("dmac.qsf", $conteudo);
                gravar("dmac.qpf", $conteudo_tb);
                gravar("dmac.sdc", $conteudo_2);
}

?>
```

B.0.8 dmac.html

```
<html>
        <title> Auto gera&ccedil;&atilde;o de controlador DMA <
           /title>
        <pre>
                Auto Gera&ccedil;&atilde;o de IP de um
                   controlador de acesso direto &agrave; mem&
                   oacute;ria (DMAC) com N canais

                Trabalho proposto de conclus&atilde;o de curso
                   de Ci&ecirc;ncias da Computa&ccedil;&atilde;
                   o pela Universidade Federal de Santa
                   Catarina
                Semestre: 2011/1
                Aluno: Carlos Eduardo de Campos

                Esse trabalho propoe a gera&ccedil;&atilde;o
                   autom&aacute;tica de um IP de um controlador
                    de acesso direto a mem&oacute;ria
                   selecionando o numero de canais desejados

                Requisitos para acesso do trabalho: um
                   descompactador ZIP

                Gera&ccedil;&atilde;o do Controlador
                <form action="gera_tudo.php" method=GET>
                Quantidade de canais: <select name=nro_canal>
                        <option value=01>01</option>
```

```
            <option value=02>02</option>
            <option value=03>03</option>
            <option value=04>04</option>
            <option value=05>05</option>
            <option value=06>06</option>
            <option value=07>07</option>
            <option value=08>08</option>
            <option value=09>09</option>
            <option value=10>10</option>
            <option value=11>11</option>
            <option value=12>12</option>
            <option value=13>13</option>
            <option value=14>14</option>
            <option value=15>15</option>
            <option value=16>16</option>
            <option value=17>17</option>
            <option value=18>18</option>
            <option value=19>19</option>
            <option value=20>20</option>
            <option value=21>21</option>
            <option value=22>22</option>
            <option value=23>23</option>
            <option value=24>24</option>
            <option value=25>25</option>
            <option value=26>26</option>
            <option value=27>27</option>
            <option value=28>28</option>
            <option value=29>29</option>
            <option value=30>30</option>
            <option value=31>31</option>
            <option value=32 selected>32</option>
        </select>
        <input type="checkbox" name=projeto_altera
           value=1>  Gerar Projeto altera
        <!--<input type="checkbox" name=gerar_tb value
           =1>  Gerar Test Bench-->
```

```
                <input type=hidden name=gerar_tb value=0>

                Endere&ccedil;o de Mapeamento do DMA (25 bits)
                   em BINARIO <input type=text size=30 name=
                   mapeamento value="0000000000000000000000000"
                   >
                !! Tenha CERTEZA que o mapeamento est&aacute;
                   CORRETO !!

                Buffer:
                <input type="radio" name="memoria" value="
                   altera" checked>Mapeamento Altera
                <input type="radio" name="memoria" value="
                   registrador">Mapeamento Banco de
                   registradores (sem pr&eacute; mapeamento)


                <input type=submit value="Gerar␣DMA␣Controller"
                   >
                </form>
        </pre>
</html>
```