UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA E ELETRÔNICA
CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Vitor Augusto Woyakewicz

Implementação de um sistema de sincronização de amostragem em FPGA para subestações digitais

Florianópolis
2022

Vitor Augusto Woyakewicz

Implementação de um sistema de sincronização de amostragem em FPGA para subestações digitais

Trabalho de Conclusão de Curso submetida ao Curso de Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina como requisito para a obtenção do título de Bacharel em Engenharia Elétrica.
Orientador: Prof. Dr. André Luis Kirsten

Florianópolis
2022

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Woyakewicz, Vitor Augusto
Implementação de um sistema de sincronização de amostragem em FPGA para subestações digitais / Vitor Augusto Woyakewicz ; orientador, André Luís Kirsten, 2022.
79 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia Elétrica, Florianópolis, 2022.

Inclui referências.

1. Engenharia Elétrica. 2. Subestações Digitais. 3. Circuitos Digitais. I. Kirsten, André Luís. II. Universidade Federal de Santa Catarina. Graduação em Engenharia Elétrica. III. Título.

Vitor Augusto Woyakewicz

Implementação de um sistema de sincronização de amostragem em FPGA para subestações digitais

Este Trabalho Conclusão de Curso foi julgado adequado para obtenção do Título de "Bacharel em Engenharia Elétrica" e aceito, em sua forma final, pelo Curso de Graduação em Engenharia Elétrica.

Florianópolis, 17 de Março de 2022

Documento assinado digitalmente
Jean Vianei Leite
Data: 25/03/2022 14:28:57-0300
CPF: 003.474.909-80
Verifique as assinaturas em https://v.ufsc.br

Prof. Jean Vianei Leite, Dr.
Coordenador do Curso de Graduação em Engenharia Elétrica

Banca Examinadora:

Documento assinado digitalmente
Andre Luis Kirsten
Data: 25/03/2022 09:00:45-0300
CPF: 013.272.690-40
Verifique as assinaturas em https://v.ufsc.br

Prof. André Luis Kirsten, Dr.
Orientador
Universidade Federal de Santa Catarina

Documento assinado digitalmente
Samir Ahmad Mussa
Data: 29/03/2022 08:57:07-0300
CPF: 401.044.430-49
Verifique as assinaturas em https://v.ufsc.br

Prof. Samir Ahmad Mussa, Dr.
Universidade Federal de Santa Catarina

Documento assinado digitalmente
CELSO LUIS DE SOUZA
Data: 25/03/2022 14:05:32-0300
CPF: 101.895.278-03
Verifique as assinaturas em https://v.ufsc.br

Eng. Celso Luis de Souza, Me.
AQTech

Este trabalho é dedicado ao meu pai Douglas David Woyakewicz

AGRADECIMENTOS

Primeiramente aos meus pais, Helenice Mezadri Woyakewicz e Douglas David Woyakewicz e ao meu irmão Vinícius David Woyakewicz, que sempre me apoiaram e incentivaram durante toda a graduação e sempre serviram como inspiração ao longo da minha vida.

Ao meu orientador, Prof. Dr. André Luis Kirsten pelo direcionamento durante a realização deste trabalho.

Aos meus amigos e colegas de classe, pela troca de conhecimento e apoio na minha vida acadêmica e pessoal.

Agradeço a AQTech pela oportunidade que me proporcionaram e aos meus colegas da empresa pela troca de conhecimentos e auxilio no desenvolvimento desse trabalho, sempre proporcionando suporte quando necessário.

“Ninguém quer aprender com os erros, mas não podemos aprender o suficiente com os sucessos para ir além do estado da arte”

(Petroski, 1985)

RESUMO

Subestações digitais substituem os métodos de medição de grandezas de potência, detecção de falha e proteção convencionais, como transformadores de corrente, transformadores de tensão e relés, por equipamentos de medida não convencionais que utilizam sensoriamento digital. Dessa forma, a utilização de cabeamento de cobre é reduzida, bem como o risco a exposição a alta tensão. Sua instalação e substituição também é simplificada, tendo uma maior flexibilidade, padronização e interoperabilidade entre os equipamentos. A digitalização desses processos proporcionam maior acurácia e um controle melhor de seus processos. Subestações digitais incorporam diversos desses sensores, dentre equipamentos de detecção de falha, proteção de falha, oscilografia e processamento por *software*, trocando informações via protocolos de comunicação industriais. Contudo, para garantir a sincronização de todos esses equipamentos faz-se uso de uma fonte de sincronismo, em geral um GNSS, conectada a eles. A sincronização é essencial para o ordenamento correto dos eventos, principalmente em caso de falha onde é preciso fazer uma auditoria. O foco deste trabalho é apresentar uma solução para a amostragem síncrona de dispositivos de medida dentro das subestações digitais. Para tanto foi escolhido a utilização de dispositivos FPGA para essa implementação, pelo seu grande paralelismo e capacidade de processamento em tempo real. Os resultados obtidos estão dentro das expectativas, respeitando os limites de erro estipulados pela norma, sendo assim adequado para a utilização nos equipamentos propostos.

Palavras-chave: Sincronismo. VHDL. FPGA. PPS. IRIG-B. Subestações digitais.

ABSTRACT

Digital substations replace the methods of acquiring power measurements, fault detection and conventional protection, such as current transformers, voltage transformers and relays, by unconventional measurement equipments that use digital sensing. This way, the use of copper cabling is reduced, as well as the risk of exposure to high voltage. Its installation and replacement is simplified, having greater flexibility, standardization and interoperability between equipment. The digitization of these processes provide a better accuracy and control of its processes. Digital substations incorporate several of these sensors, including fault detection equipment, fault protection, oscillography and software processing, communicating via industrial communication protocols. However, to ensure synchronization of all these devices, a synchronization source is used, usually a GNSS, connected to them. Synchronization is essential for the correct ordering of events, especially in fault cases, where an audit is required. This work is focused in presenting a solution for the synchronous sampling of measurement devices inside digital substations. For this purpose, the use of FPGA devices was chosen for the target implementation, due to its great parallelism and real time processing capacity. The results obtained are within expectations, respecting error limitations stipulated by the standards, being therefore suitable for use in the proposed equipment.

Keywords: Synchronism. VHDL. FPGA. PPS. IRIG-B. Digital substations

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ASIC	*Application Specific Integrated Circuit*
BCD	*Binary-Coded Decimal*
BNC	Conector *Bayonet Neil Councilman*
CC	Corrente Contínua
CLB	*Configurable Logic Blocks*
CMEMS	Sistemas Osciladores Eletromecânicos CMOS
CORDIC	*Coordinate Rotation Digital Computer*
CPLD	Dispositivo Lógico Complexo Programável
DMA	*Direct Memory Access*
DNP	*Distributed Network Protocol*
FPGA	*Field Programable Gate Array*
GLONASS	Sistema de Navegação Global por Satélite
GNSS	Sistema de Satélite de Navegação Global
GOOSE	*Generic Object Oriented Substation Event*
GPIO	*General Purpose Input Output*
GPS	*Global Positioning System*
HDL	*Hardware Description Language*
IDE	Ambiente de Desenvolvimento Integrado
IEC	*International Electrotechnical Commission*
IED	*Intelligent Electronic Devices*
IEEE	*Institute of Electrical and Electronics Engineers*
IHM	Interface Homem-Máquina
IO	*Input Output*
IP	*Internet Protocol*
IRIG-B	*Inter-Range Instrumentation Group Time Code B*
LED	Diodo Emissor de Luz
LUT	*Look up Table*
MMS	*Manufacturing Message Specification*
MU	*Merging Unit*
NCO	*Numerically Controlled Oscilator*
NTP	*Network Time Protocol*
PAC	*Phase to amplitude Converter*
PI	Proporcional Integral
PLL	*Phase Locked Loop*
PPM	*Pulse Per Minute*
PPS	*Pulse Per Second*
PTP	*Precision Time Protocol*

RDP	Registrador Digital de Perturbação
RTL	*Register Transfer Level*
SAS	Sistemas de Automação de Subestação
SB	*Straight binary*
SCADA	Sistema de Supervisão e Aquisição de Dados
SoC	*System-on-a-Chip*
ST	Conector *Bayonet-lock*
SV	*Sampled Values*
TCL	*Tool Command Language*
TTL	Lógica Transistor-Transistor
UDP	*User Datagram Protocol*
USB	*Universal Serial Bus*
VHDL	*VHSIC Hardware Description Language*
VHSIC	*Very High Speed Integrated Circuit*

LISTA DE SÍMBOLOS

K_c	Ganho do controlador
τI	Constante Integral
$e(t)$	Erro entre valor desejado e valor medido no instante t
$u(t)$	Saída do sistema em um instante t
u_{bias}	Constante de viés
dt	Mudança infinitesimal de t
n_t	Número de amostras
Δt	Período entre amostras
t_{su}	Tempo de *setup*
t_h	Tempo de *hold*
t_{co}	Atraso *clock-to-output*

SUMÁRIO

1 INTRODUÇÃO

O avanço da sociedade e o desenvolvimento tecnológico decorrente dos últimos anos acarretaram em um maior consumo energético. Naturalmente, foram necessários avanços nos sistemas de geração, transmissão e distribuição de energia para aumentar o volume de energia gerado, com mais eficiência, segurança e comodidade. A partir dessa necessidade surgiram os sistemas de automação de subestação (SAS), e subestações totalmente digitais.

As subestações digitais e os seus sistemas de automação de subestação promovem uma comunicação moderna e eficiente entre seus dispositivos, melhor controle da gestão de ativos, engenharia integrada e interoperabilidade, digitalização do nível de subestação e melhorias de segurança e ciber-segurança. Também é dado grande ênfase na maior segurança dos trabalhadores expostos a esses equipamentos.

Nessas condições os transformadores de corrente e tensão convencionas são substituídos por transformadores de instrumentação não convencionais. Esses equipamentos se diferenciam principalmente pelo modo em que são medidas essas grandezas, usando métodos não intrusivos. Ao invés de se usarem transdução eletromagnética por acoplamento indutivo entre enrolamentos montados sobre núcleos ferromagnéticos e/ou em divisores capacitivos, seu funcionamento se baseia em efeitos ópticos como efeito Faraday e outros métodos não intrusivos como bobinas de Rogowski. A sua comunicação também se difere, visto que ao invés de cabeamento de cobre é utilizado cabos de fibra óptica.

Por sua vez a interface dos dados desses transformadores é feito por meio de *Merging Units* (MU), que podem ser conectados a equipamentos de oscilografia como os Registradores Digitais de Perturbação (RDP). Esses equipamentos fazem a aquisição das medidas e viabilizam sua análise, se comunicando com outros equipamentos via ethernet em padrões de comunicação para subestações.

A sincronização dos dispositivos dentro de uma subestação se faz necessária para garantir a ordem de eventos e interoperabilidade entre eles, principalmente em casos de falhas e interrupções de energia, que podem ser ocasionadas por equipamentos defeituosos, sobrecarga, desastres naturais, erros humanos dentre outros. No caso de uma perturbação em um instante de tempo, devemos garantir a correlação individual de cada dispositivo com o evento. A localização de falhas pode ser calculada usando métodos como ondas viajantes (RAJU; ALEX, 2015).

A sincronização é feita por meio de um relógio mestre que fornece um sinal de sincronismo para todos os dispositivos. Esse relógio normalmente consiste em um GNSS extremamente preciso que se comunica com os dispositivos via fibra óptica ou *ethernet*, utilizando diversas interfaces e protocolos diferentes.

Uma solução para geração de frequências sincronizadas a esses relógios, que

são frequências utilizadas para aquisição de medidas, são os disciplinadores de *clock*. Disciplinadores de *clock* são circuitos que geram *clock*s autoajustáveis em tempo real que seguem um sinal de sincronismo referência. Esses *clock*s serão utilizados então para fazer a aquisição síncrona de dados e enviá-los por um protocolo via rede.

Sua variação e acurácia deve seguir as normas pré-estabelecidas para os equipamentos e suas aplicações. Portanto a precisão, confiabilidade e latência desses circuitos são de extrema importância. Dessa forma a implementação desse circuito foi realizada em um circuito digital dentro de um FPGA. FPGA's são dispositivos lógicos programáveis muito utilizados nessa área por sua confiabilidade, paralelismo, flexibilidade e mais importante capacidade de suportar processamento em tempo real.

O trabalho foi realizado dentro do projeto de P&D intitulado : PD-00403-0047/2019 "Cabeça de Série do Transformador Eletrônico de Corrente Óptico – TECO-MR". O mesmo é um convênio de cooperação técnico-científica – Projeto de P&D ENGIE – ANEEL tendo como proponentes ENGIE e executoras AQTech e PowerOpticks. O trabalho aqui desenvolvido visa ser utilizado na *Merging Unit* para o Transformador Eletrônico de Corrente Óptico.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Implementar em linguagem VHDL um circuito digital para geração de frequências sincronizadas a uma referência, visando sua utilização em equipamentos de amostragem de dados dentro de subestações digitais.

1.1.2 Objetivos Específicos

- Validar e deixar o sistema apto para diferentes padrões de comunicação dentro das subestações: IRIG-B e PPS.
- Comparar a eficácia do sistema com diferentes osciladores: CMEMS e oscilador de cristal
- Atender as especificações de tempo da norma IEC 61850
- Criar um sistema confiável, robusto e que possa se recuperar de perdas de sinais de referência.
- Avaliar seu comportamento em um equipamento real dentro de um ambiente controlado

2 FUNDAMENTAÇÃO TEÓRICA

2.1 SUBESTAÇÕES DIGITAIS

Uma subestação é uma instalação elétrica de alta potência de grande importância na transmissão e distribuição de energia elétrica. Nela estão compreendidos os dispositivos de manobra, controle e proteção das extremidades de linha de transmissão e/ou de distribuição e que abrange as obras civis e estruturas de montagem. Nela pode incluir também transformadores, equipamentos conversores e/ou outros equipamentos(ONS, 2021)

A subestação tem como seus principais objetivos rebaixar ou elevar a tensão da energia, de forma a adequá-la às necessidades de transmissão, distribuição e consumo. Os principais tipos de subestações são:

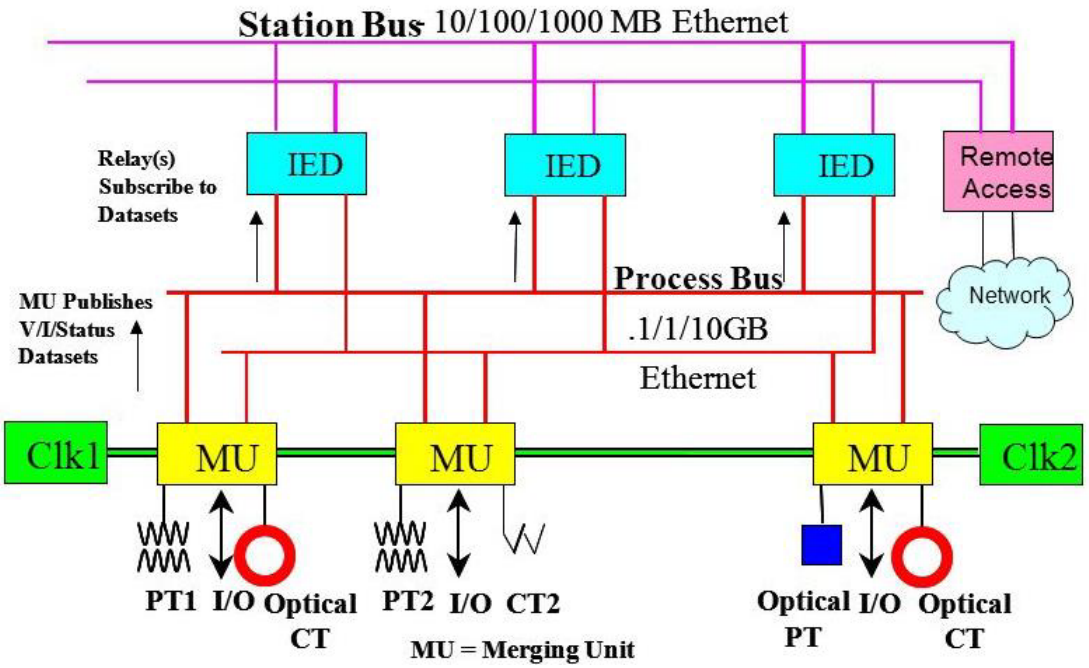
- Subestação elevadora: Tem por finalidade elevar a tensão para que a energia seja transmitida com uma corrente menor e sem grandes perdas. São normalmente instaladas junto a usinas geradoras.
- Subestação abaixadora: Estão localizadas nas periferias das cidades. Tem como seu papel reduzir a tensão gerada para níveis mais baixos, evitando inconveniências para a população.
- Subestação de distribuição: Diminuem as tensão para o nível de distribuição primária

As subestações digitais surgiram da necessidade de aumentar a disponibilidade, confiabilidade e sustentabilidade do fornecimento de energia. Elas podem ser classificadas por subestações nas quais a operação é realizada através de *Intelligent Electronic Devices* (IEDS) interconectados por redes de comunicação. A comunicação é feita utilizando protocolos de comunicação voltados para o sistema de energia, como por exemplo DNP(3.0) e, no caso deste trabalho, IEC 61850 que será abordada na próxima seção. Na Figura 1 é apresentado um modelo de arquitetura de subestação baseada em IEC 61850.

As subestações digitais apresentam diversos benefícios em comparação com subestações convencionais, dentre eles podemos citar (WALEED *et al.*, 2019):

- Menor utilização de cobre, sendo substituídos por fibras óticas. O que reduz o número de conexões entre equipamentos e reduz cabeamento que precisa ser feito para fontes de energia e outras conexões. As maiores vantagens da fibra óptica sobre cabeamento de cobre são maior segurança, maior vida útil, maior facilidade de manutenção e menor custo de infraestrutura.
- Transporte reduzido devido a utilização de fibras óticas ao invés de cobre e pelo uso de transformadores de instrumentação óticos ao invés dos convencionais.

Figura 1 – Arquitetura de subestação baseada em IEC 61850.



Fonte: (WALEED *et al.*, 2019)

- Redução do espaço em relação aos IOs convencionais, o que acarreta uma integração maior dos sistemas de proteção e controle.
- Redução no tempo de instalação e de interrupção. Pelo fato de existirem menos cabos a instalação se torna mais rápida e simples
- Menor custo operacional. A supervisão e manutenção é reduzida pelo fato de todos os dados comunicados serem supervisionados

Contudo, alguns pontos negativos também devem ser destacados como:

- Investimento maior em *cyber* segurança. Pelo fato de todos os processos serem realizados via ethernet há uma preocupação maior em relação a invasões e vazamentos de dados.
- Capacitação dos trabalhadores. Pelo fato da operação e manuseio dos equipamentos ser diferente em comparação com subestações convencionais, os operadores das subestações precisam ser capacitados para trabalhar com tais equipamentos não convencionais.
- Demora na implementação do sistema. A integração dos IEDS e dos sistemas de aquisição e controle pode ser lenta comparada as subestações convencionais

Um estudo conduzido pela marketsandmarkets aponta que o mercado de subestações digitais é projetado a ter um aumento de USD 6,4 bilhões em 2020 para

USD 9,1 bilhões em 2025. É esperado um crescimento de CAGR de 7,1% durante o período previsto(GARG, 2022).

2.2 IEC 61850

A norma IEC 61850 faz parte do IEC TC57 (*International Electrotechnical Commission's Technical Committee* 57) e padroniza a troca de informações entre IEDS e os sistemas supervisórios nos sistemas de automação de subestações. Sendo assim, ela proporciona a integração de toda a proteção, controle, informações de medidas e monitoramento de funções dentro das subestações digitais proporcionando interoperabilidade entre os dispositivos.

A norma estabelece três níveis hierárquicos, sendo eles o nível de estação, nível de processo e nível de *Bay*. O nível de estação corresponde as interfaces Homem-Máquinas (IHMs) supervisórios e *gateways* de comunicação, possibilitando o monitoramento da subestação. Já no nível de *Bay* se encontram os IED's responsáveis pela proteção e controle da subestação. No nível de processo estão todos os equipamentos do pátio da subestação, como disjuntores, seccionadoras, transformadores e também as *Merging Units*. As MUs são equipamentos que integram os transformadores de instrumentação não convencionais as subestações. Elas recebem valores amostrados dos transformadores e por meio de um *frame Ethernet* definido pela norma, os enviam para a rede.

Seus barramentos são divididos em barramento de processo e barramento de estação. O barramento de estação comunica os níveis de estação e *Bay* no modo vertical, para operadores e centros de controle remoto, através do protocolo MMS (Manufatuing Message Specification), já no modo horizontal, entre IEDs através de mensagens GOOSE (Generic Object Oriented Substation Event). O barramento de processo realiza a troca de leituras analógicas da subestação através do protocolo SV (Sampled Values). Nesse barramento as medidas adquiridas pelas *Merging Units* enviam *Sampled Values* utilizando fibra óptica como meio físico.

Dentre os benefícios da utilização da norma IEC 61850 estão (ALMEIDA, 2011):

- Universalização do protocolo de comunicação entre IEDs
- Redução de cabeamento e dos pontos de entradas e saídas dos equipamentos digitais
- As informações entre os equipamentos e subsistemas são compartilhadas entre si
- Transmissão em tempo real do estado de chaves seccionadoras e disjuntores do sistema de controle e supervisão
- Transdução digital das medidas de corrente e tensão de cada fase.

2.3 MEDIÇÕES E OSCILOGRAFIA

Os equipamentos para medição e oscilografia são de extrema importância para detecção de perturbações, falhas, desbalanceamento, variações na frequência e um controle geral da tensão e corrente nas linhas de transmissão. Ela pode ser definida como o registro de medidas de potência, como medidas de corrente e tensão. Diferentemente dos sistemas de proteção que atuam em tempo real em resposta aos distúrbios provenientes nas linhas, os equipamentos de medição e oscilografia viabilizam a análise após a ocorrência de um distúrbio.

Dentre as primeiras versões de oscilógrafos podemos citar o ondógrafo de Hospitalier, desenvolvido na França em 1901. Esse aparelho registrava o gráfico da variação no tempo de um fenômeno elétrico diretamente em uma banda de papel. Já na década de 80 começaram a surgir os primeiros registradores digitais de perturbação.

A oscilografia é um estudo de grande importância dentro dos sistemas elétricos de potência. Dentre os equipamentos utilizados podemos citar os registradores de perturbação que tem por finalidade fazer a aquisição de sinais digitais, como grandezas de corrente e tensão de fase ou linha e gravá-las em continuamente em uma memória cíclica (MORETO; ROLIM, 2010). Outro equipamento são as *Merging Units* que fazem a interface com transformadores de instrumentação convencionais ou não-convencionais e entregam *Sampled Values* com medida de tempo via rede *ethernet*.

A sincronização nesses equipamentos são de extrema importância para ordenação de eventos e detecção da localidade das falhas. É muito comum a utilização dos dados da oscilografia para fazer a detecção de distúrbios na linha de transmissão, porém se os equipamentos não estiverem devidamente sincronizados entre si utilizando um *Master Clock* os resultados serão inconclusivos.

2.4 SINCRONISMO

A tarefa de analisar a grande quantidade de dispositivos de proteção e controle interconectados dentro de uma subestação se torna complexa se os mesmos não estiverem com estampas de tempo coerentes e sincronizadas. Dentro de uma subestação digital o sincronismo normalmente é realizado através de um aparelho GNSS que será o *Master Clock*. Essa informação de sincronismo pode ser enviada para os aparelhos por diversos padrões de comunicação e protocolos, sendo os mais comuns PPS, IRIG-B, NTP e o PTP.

Abaixo cada um desses padrões e protocolos serão discutidos com mais detalhes:

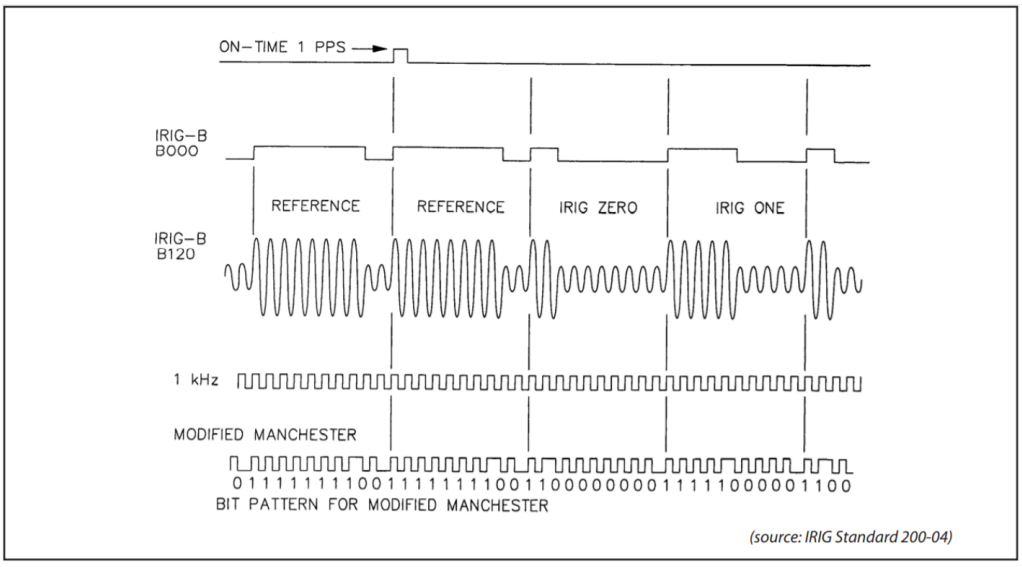
2.4.1 1-PPS

O sinal PPS ou 1-PPS consiste em um sinal de um pulso por segundo com amplitude configurável pelo equipamento, sendo um sinal com frequência perfeita com um *jitter* residual. Na maioria dos receptores de GPS o erro relativo está na casa das dezenas de nanosegundos (GASPARINI *et al.*, 2007).

2.4.2 IRIG-B

A codificação de tempo IRIG foi originalmente desenvolvido pelo Inter-range instrumentation group (IRIG), parte do exército americano. Sua última revisão é a IRIG standard 200-04 (RCC, 2004). O protocolo IRIG-B, que é apenas um dos formatos de codificação de tempo IRIG, é largamente usado em sistemas de potência para assegurar sincronização de tempo precisas de sistemas de dispositivos de potência como Relés. Esses sinais possuem uma taxa de pulso de 100 PPS, ou seja, 100 pulsos por segundo, com um intervalo de contagem para cada índice de 10 ms. Os sinais IRIG-B usam codificação por largura de pulso, com o valor binário “0” tendo a duração de 20% do intervalo de índice, o valor binário “1” tendo a duração de 50% do índice e finalmente os indicadores de posição “P” tendo a duração de 80%, sendo utilizados como marcadores de referência. Dois marcadores consecutivos P indicam o começo de um *frame* e um PPS. A codificação de tempo IRIG-B pode ser modulada (com portadora sinusoidal), não modulada (com mudança de nível CC) ou Manchester Modificado (com portadora de onda quadrada). A Figura 2 ilustra os diferentes tipos de codificação.

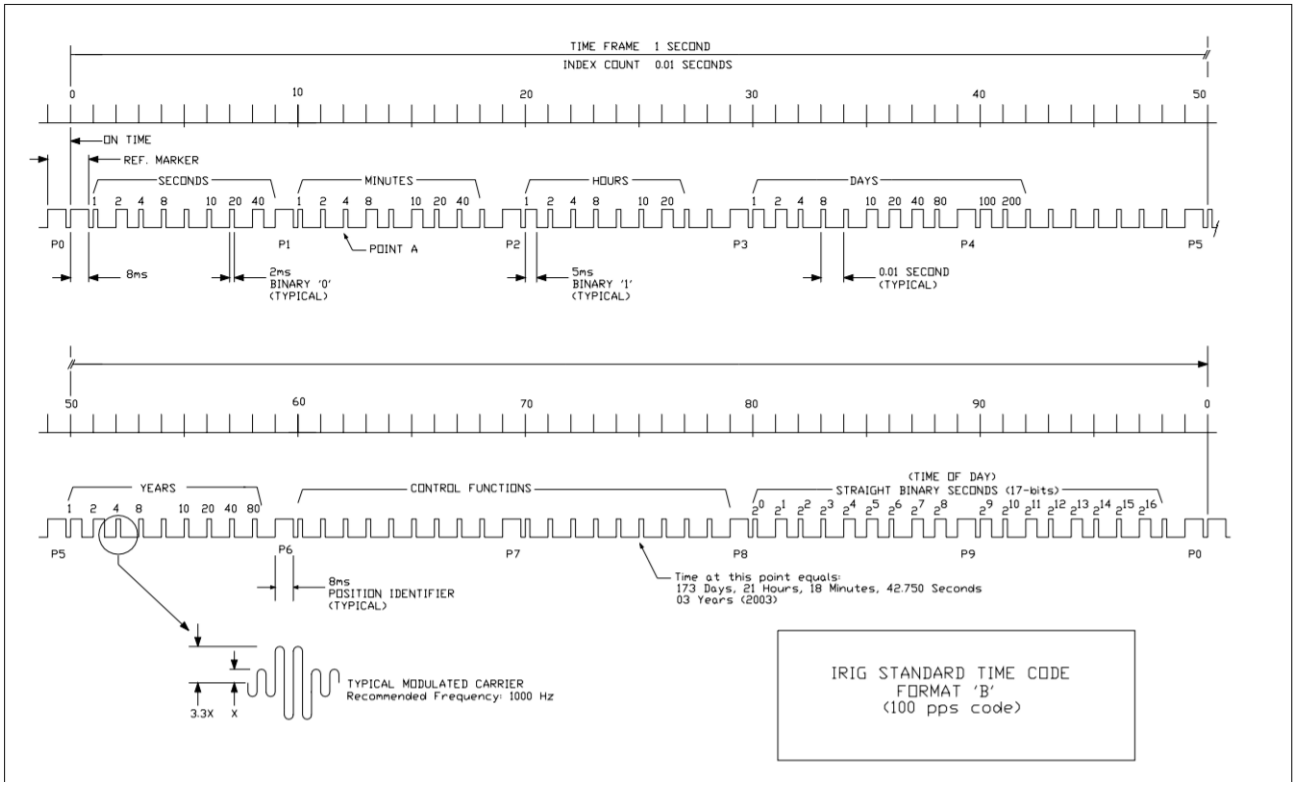
Figura 2 – Codificações do sinal IRIG-B



Fonte: (RCC, 2004)

O IRIG-B possui 100 bits por *frame*, contendo diversos tipos de informação. Desses 100 bits, 30 são de informação de tempo do ano em BCD em dias, horas, minutos e segundos, 17 bits de SB segundos por dia, 9 bits de informação do ano e 18 bits de controle de funções como mostra a figura 3.

Figura 3 – Formato do código IRIG-B



Fonte: (RCC, 2004)

2.4.3 NTP

O *Network Time Protocol*, que teve sua primeira versão em 1985, é um protocolo que sincroniza dispositivos dentro de uma rede a partir de fontes confiáveis. Seu funcionamento consiste em dividir a *subnet* em uma hierarquia de 3 níveis, chamados de estratos, definidos pelas suas distâncias em nós até uma dessas fontes registradas. No estrato 0 ficam os relógios atômicos, já no estrato 1 ficam os servidores conectados aos relógios e por fim, no estrato 2, ficam os computadores conectados a estes servidores e assim por diante (MILLS, 2021). Comparado aos outros protocolos o NTP tem uma precisão ruim, não conseguindo ultrapassar os 50 μs, sendo normalmente de 1 ms. Desta forma ela é raramente utilizada quando precisão é necessária, sendo mais comum para sincronizar relógios de computadores.

2.4.4 PTP

O *Precision Time Protocol* foi originalmente definido na norma de número 1588 do *Institute of Electrical and Electronics Engineers* (IEEE) em 2002 e em 2008 teve uma segunda versão com alguns padrões revisados e aprimorados. O PTP utiliza redes ethernets para obter sincronismo preciso de *clocks*, obtendo uma precisão na casa dos nanosegundos. O protocolo também inclui mapeamento para UDP/IP, *DeviceNet* e implementação na segunda camada Ethernet (**IEE**). Apesar do seu ótimo desempenho, muitos equipamentos, principalmente os mais antigos, não possuem suporte a PTP dentro de uma subestação. Dessa forma o protocolo ainda precisa de um período de adaptação por parte da infraestrutura das subestações.

2.5 GNSS

O Sistema de Satélite de Navegação Global (GNSS) é a constelação de satélites que por sua vez possibilita o posicionamento em tempo real, bem como fornecendo informação temporal. Receptores GNSS são fontes de sinais de sincronização temporal referenciados a satélites GPS e GLONASS que a partir deles realiza uma triangularização para identificar a sua posição precisa e em caso de sincronização, o tempo exato.

Receptores GNSS precisam de no mínimo 3 satélites visíveis para funcionar corretamente, o que nem sempre ocorre caso a antena receptora esteja em um local fechado ou esteja bloqueada por algum outro motivo. Os formatos e protocolos normalmente utilizados para sincronizar os relógios internos de equipamentos a partir de um *Grand-Master Clock* são o PPS, IRIG-B, PTP, NTP e seus derivados.

2.6 DISCIPLINADORES DE CLOCK

Disciplinadores de *clock* são circuitos capazes de gerarem frequências de *clock* para sistemas de sincronismo, sincronizadas em fase a um *clock* referência gerado por outro equipamento ou dispositivo. Esses circuitos recebem como entrada essa referência e é realimentado pela sua saída, de forma a comparar o sinal gerado virtualmente com a sua referência para reduzir o *jitter*. A falta de sincronismo ou grandes taxas de deriva em equipamentos de medição pode acarretar em erros de medidas, bem como defasagem entre aquisições de diferentes equipamentos.

As especificações de taxa de deriva devem respeitar as normas e padrões, uma delas é a especificação do *jitter* em regime permanente que de acordo com a IEC 61850-5 (IEC, 2013) deve ser menor que 1 μs.

Sistemas de sincronismo também devem ser resistentes a perdas, funcionando mesmo com a falta de uma referência. Na volta dessa referência o sistema deve se ajus-

tar ao sinal de referência novamente, evitando saltos ou comportamentos inusitados no processo.

2.7 CONTROLADORES PROPORCIONAIS INTEGRAIS DISCRETOS

Os algoritmos de controle PI são um dos algoritmos de controle mais utilizados na indústria, sendo que nos casos em que se é utilizado no domínio amostrado é usado sua versão discretizada. Controladores proporcionais integrais são utilizados para controlar a saída de sistemas em uma malha fechada, ou seja, usando o princípio da realimentação. Esses controladores podem ser ajustados de forma a obter uma resposta desejada, manter a saída em um valor estável e manter o erro em regime permanente de acordo com dadas especificações. A equação de controle analógica é proporcional ao erro $e(t)$ e a sua integral, tendo sua saída $u(t)$ dada pela Equação (1).

$$u(t) = u_{bias} + K_c e(t) + \frac{K_c}{\tau I} \int_0^t e(t)dt \tag{1}$$

O primeiro termo u_{bias} é uma constante que representa uma estimativa inicial de $u(t)$ ou um valor nulo. O segundo termo $K_c e(t)$ corresponde a parte do controlador proporcional ao erro e(t). O último termo $\frac{K_c}{\tau I} \int_0^t e(t)dt$ é a parte que entregará uma resposta a integral do erro. As constantes mais importantes de ajuste do controlador PI são o k_c e τI. k_c é uma constante de ganho da integral que influenciará na agressividade do controlador e τI é a constante integral (ou *reset time*) que influencia a velocidade da resposta do sistema.

Para a utilização do controlador no domínio amostrado é necessário discretizar a Equação (1). Para tanto a integração do tempo no erro será aproximado por uma soma de erros entre instantes de tempo amostrados em um número n_t de amostras, gerando a Equação (2).

$$u(t) = u_{bias} + K_c e(t) + \frac{K_c}{\tau I} \sum_0^{n_t} ei(t)\Delta t \tag{2}$$

2.8 NCO

Um *Numerically Controlled Oscilator* (NCO) é um oscilador criado usando lógica digital para gerar ondas senoidais ou outras formas de onda periódicas como ondas quadradas. Quando utilizado para gerar ondas senoidais, ou outras ondas mais complexas, é necessário a utilização de *Look up Tables* (LUT's) ou *Coordinate Rotation Digital Computer* (CORDIC). NCO's apresentam diversas vantagens em relação a outros tipos de osciladores em termos de precisão, estabilidade e confiabilidade, sendo usado em várias arquiteturas digitais como PLLs(LATTICE, 2021).

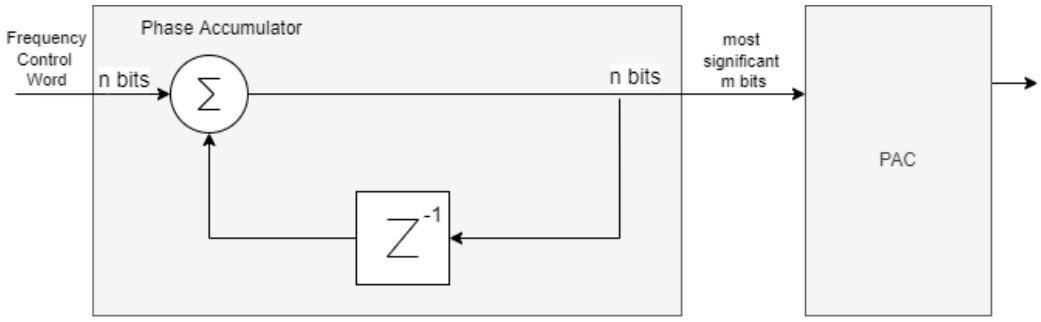
Dentre as inúmeras utilidades para um NCO podemos citar(GISSELQUIST, 2017):

- Geração de sinais senoidais extremamente precisos
- Geração de *clocks* síncronos
- Utilização em moduladores e demoduladores em comunicação digital

2.8.1 Funcionamento

O NCO é constituído por dois blocos fundamentais, *phase accumulator* que acumula uma palavra de controle de frequência e um *phase to amplitude converter*, que irá converter o valor do acumulador de fase para geração de um sinal especificado. Essa estrutura pode ser observada na Figura 4 abaixo.

Figura 4 – Diagrama de blocos de um NCO



Fonte: do autor

2.8.1.1 Phase Accumulator

O acumulador de fase é constituído por um bloco de soma e um registrador de n *bits*. A entrada *frequency control word*, ou também K, é uma constante que deverá ser cuidadosamente escolhida para gerar a frequência de saída correspondente. A cada ciclo de *clock* o valor do *frequency control word* será adicionado ao registrador. Esse acumulo irá gerar uma rampa que irá transbordar quando o valor dessa soma exceder o valor absoluto de $2^n - 1$. Essa saída será posteriormente utilizada pelo *Phase-to-Amplitude Converter* para geração de sinais.

A frequência de saída gerada por esse bloco pode ser dada pela Equação (3).

$$F_{out} = \frac{KxF_{clock}}{2^n} \tag{3}$$

Onde F_{out} é a frequência de saída, K é o *frequency control word*, Fclock é a frequência do clock e n é a quantidade de *bits* de K. A resolução, ou o menor

incremento possível de frequência é dado pela Equação (4).

$$F_{res} = \frac{clock}{2^n} \tag{4}$$

Dessa forma, a resolução do NCO está intimamente ligada a acurácia e variação do *clock* utilizado, quantidade de *bits* utilizado e a frequência de *clock* utilizada para determinada frequência de saída.

2.8.1.2 Phase-to-Amplitude Converter

O PAC constitui de *Lookup tables*, memórias, CORDICS ou outros algoritmos para geração de ondas precisas. Dependendo da sua indexação é utilizado os m bits mais significativos do *Phase Accumulator* pelo PAC. No caso de geração de ondas quadradas é utilizado apenas o *bit* mais significativo, a alternância entre os *bits* 0 e 1 acabam gerando uma forma de onda quadrada.

A precisão e acurácia do NCO está intimamente ligado ao tamanho do registrador do *Phase Acumulator*, do tamanho da *lookup table*, quando usada, e a acurácia do *clock* utilizada. Acumuladores de fase com registradores grandes implicam em uma acurácia grande na frequência instantânea.

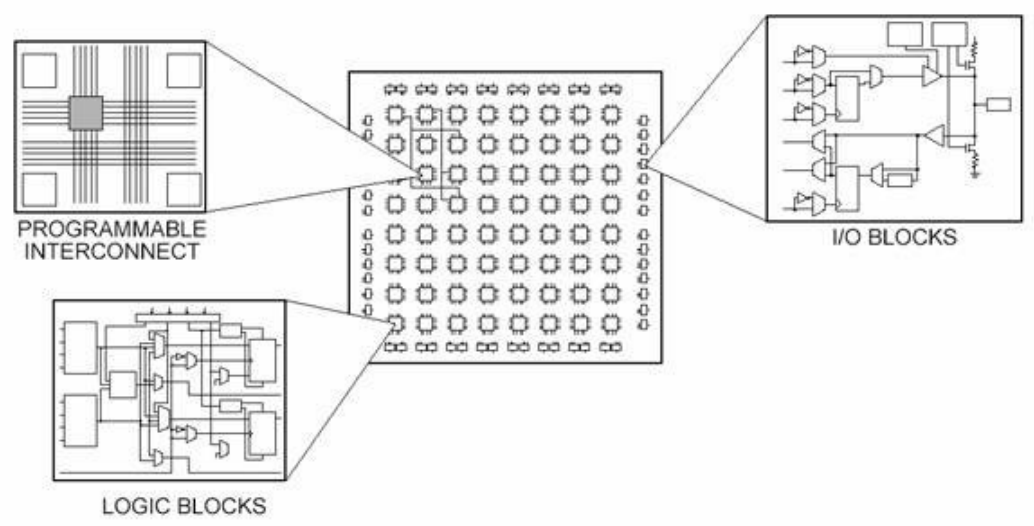
2.9 FPGA

O desenvolvimento do Hardware RTL do projeto foi feito baseado nos dispositivos FPGA. A sigla FPGA representa ***F**ield **P**rogramable **G**ate **A**rray*, ou em português "Arranjo de Portas Programáveis em Campo". São dispositivos lógicos que diferentemente de chips como ASIC podem ser reprogramados pelo usuário implementando circuitos digitais. Para tanto faz-se necessário o uso de uma linguagem de descrição de Hardware (HDL) como Verilog, VHDL ou *System Verilog*.

FPGAs são baseados em uma matriz de blocos lógicos configuráveis (CLBs) conectadas por interconexões programáveis (XILINX, 2021). Os blocos lógicos configuráveis, que são a unidade lógica básica dos FPGA, são constituídas por *flip-flops* e *lookup tables* (LUTs). As diferentes famílias de FPGAS diferem no jeito que os *flip-flops* e LUTs são empacotados juntos. A Figura 5 mostra a estrutura de um FPGA com as entradas (I/O), blocos lógicos (CLBs) e as interconexões (*Programable Interconect*).

FPGAs são muito utilizados por fornecer flexibilidade, confiabilidade e um ótimo desempenho com paralelismo em aplicações em tempo real. Esses dispositivos se destacam principalmente em aplicações em que o tempo de processamento é crítico. São muito utilizados em aplicações de áudio, setor elétrico, setor automobilístico, militar e outros.

Figura 5 – Estrutura do FPGA



Fonte: (NI, 2021)

2.10 LINGUAGEM VHDL

VHDL representa VHSIC (*very high speed integrated circuit*) HDL, ela é uma linguagem de descrição de *hardware* criada em meados da década de 1980 pelo Departamento de Defensa dos Estados Unidos da América utilizada para modelar circuitos digitais (CHU, 2006). Junto com *Verilog* e *System Verilog* ela é uma das linguagens de descrição de *hardware* mais utilizadas, sendo suportadas pela maioria das ferramentas de *software*. Apesar das suas diferenças, suas capacidades, desempenhos e escopos são extremamente similares. A escolha do HDL depende muito do nível de abstração utilizado no projeto, bibliotecas e códigos disponíveis para o projeto, nesse caso foi utilizado a linguagem VHDL pela familiaridade com a linguagem e sua utilização em outros projetos.

A seguir será feita uma breve introdução a estrutura da linguagem VHDL para melhor entendimento do projeto.

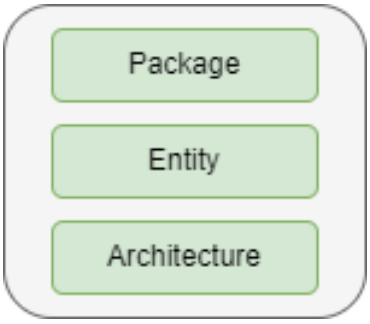
A estrutura de um programa VHDL pode ser dividida como a Figura 6:

Onde:

- *Package*: Bibliotecas e pacotes
- *Entity*: Declaração da entidade e dos seus pinos de entrada e saída
- *Architecture*: Uma entidade pode ter várias arquiteturas, sendo que cada arquitetura descreve um circuito.

Para melhor descrever os conceitos e estrutura do VHDL será mostrado um exemplo de um circuito detector de paridade par. Esse circuito recebe 3 *bits* de entrada a(0), a(1) e a(2) e a saída ira sinalizar paridade quando houver um número par de 1,

Figura 6 – Estrutura do VHDL



Fonte: do autor

ou seja quando tiver 0 ou 2 *bits* 1 de entrada. A tabela verdade e o circuito em portas lógicas são apresentados nas Figuras 7 e 8.

Figura 7 – Tabela verdade do circuito de paridade

a(2)	a(1)	a(0)	par
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Fonte: do autor

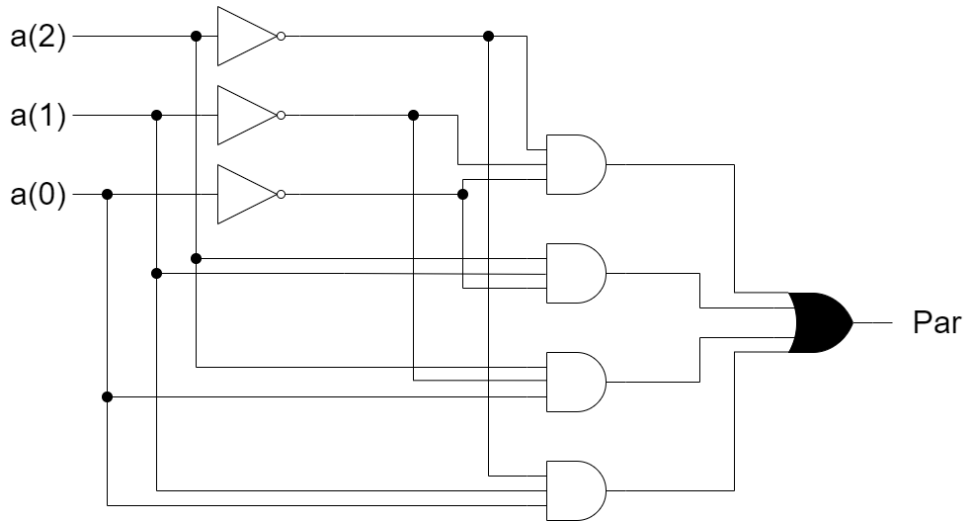
A seguir será apresentado o código em VHDL equivalente para esse implementação na Figura 9.

Primeiramente é importante destacar que as palavras em azul são palavras reservadas da linguagem VHDL e comentários são feitos com “–”. A primeira parte, correspondente as duas primeiras linhas de código, são os *packages*, ou seja as bibliotecas e os pacotes a serem utilizados.

As duas grandes unidades do código são a declaração de entidade e o corpo da arquitetura. Na entidade é declarado seu nome, as entradas e saídas bem como seus tipos. Nesse caso é declarado um vetor lógico “a” de 3 *bits* e uma saída lógica “*even*” de 1 *bit*.

Em seguida a arquitetura específica todo o funcionamento e operação do circuito digital. Primeiramente são declarados 4 sinais de 1 *bit*, p1, p2, p3 e p4. Esses

Figura 8 – Circuito detector de paridade



Fonte: do autor

Figura 9 – Implementação do detector de paridade em VHDL

```vhdl
library ieee;
use ieee.std_logic_1164.all;


-- decalração da entidade
entity even_parity is
    port(
            a: in std_logic_vector(2 downto 0);
            even: out std_logic
        );

end even_parity;

--Corpo da arquitetura
architecture even_parity_arch of even_parity is
    --Declaração dos sinais
    signal p1, p2, p3, p4 : std_logic;
begin
    --concurrent statements
    even <= (p1 or p2) or (p3 or p4);
    p1 <= (not a(2)) and (not a(1)) and (not a(0)) ;
    p2 <= (not a(2)) and a(1) and a(0);
    p3 <= a(2) and (not a(1)) and a(0);
    p4 <= a(2) and a(1) and (not a(0));
end even_parity_arch;
```


Fonte: do autor

sinais podem ser visto como “fios” que realizam interconexões. Após a palavra reservada *begin*, está contida a descrição da arquitetura, ela é composta por *concurrent statements*, ou em português declarações simultâneas. Cada declaração pode ser vista como parte do circuito e a ordem de cada declaração não influencia no funcionamento do circuito, já que seu comportamento não é sequencial. A parte da esquerda da declaração será o resultado da operação do lado direito da declaração. Se um sinal

aparecer tanto do lado direito quanto do lado esquerdo de outra operação, existe uma interconexão implícita entre esses sinais.

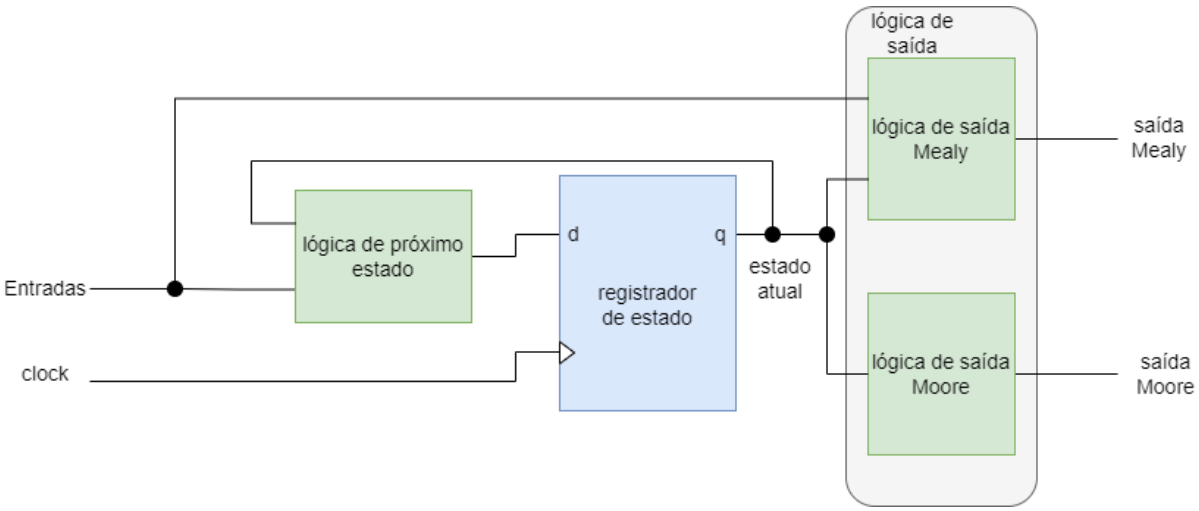
Esse capítulo foi apenas uma breve introdução a linguagem VHDL, um detalhamento maior sobre o tema pode ser encontrado em (MIT, 2020)

2.11 MÁQUINAS DE ESTADOS SÍNCRONAS

Em seguida serão introduzidas as máquinas de estados e como elas são codificadas em VHDL. Como boa parte do projeto depende de máquinas de estados elas serão revisadas para que seu detalhamento não precise ser visto sempre que uma máquina de estado for apresentada.

Apesar de que praticamente qualquer implementação possa ser realizada apenas com lógica digital, máquinas de estados síncronas se mostram uma forma mais simples e dinâmica de representar controles sequenciais. Uma máquina de estados é um circuito baseado em estados internos que tem como objetivo realizar operações em uma forma sequencial de passos. Serão abordados somente máquinas de estado síncronas, regidas por bordas de subidas de *clock*. Um diagrama de blocos de máquinas de estados se encontra na figural Figura 10.

Figura 10 – Diagrama de blocos de uma máquina de estados



Fonte: do autor

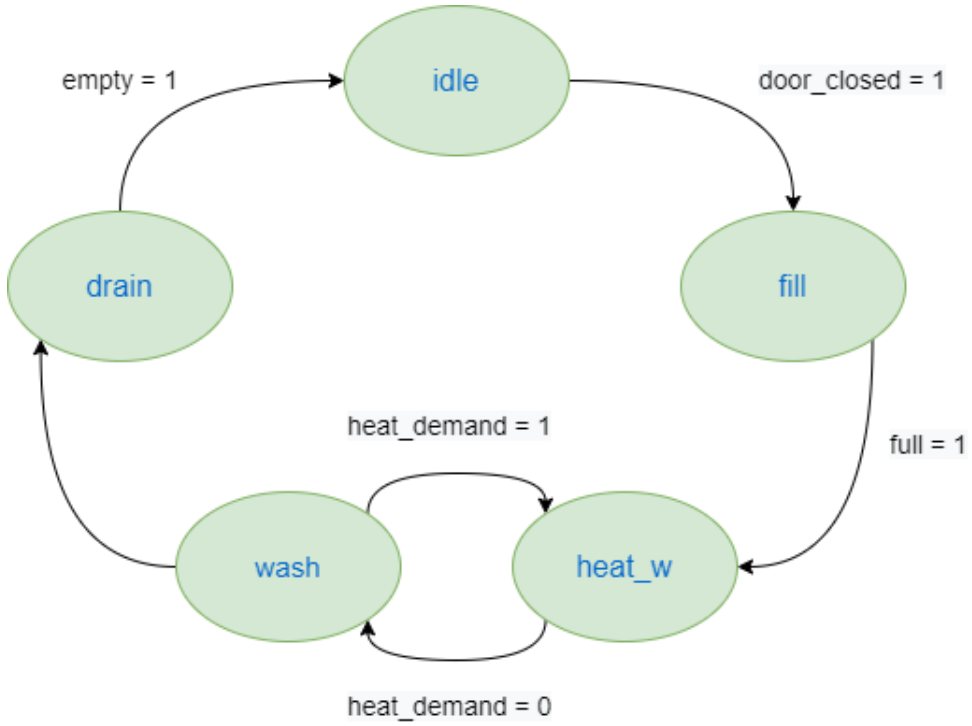
A lógica de próximo estado é uma lógica que a a partir das entradas e do estado atual irá decidir se manter em um estado ou ir para um próximo estado. o Registrador de estado é um elemento de memória regido por um *clock* global e pela saída da lógica de próximo estado. Sua função é manter o valor de estado que a máquina de estados se encontra. A lógica de saída pode ser do tipo *Moore* ou *Mealy*. Uma lógica de saída *Moore* é função apenas do estado atual da máquina de estados. Já uma

lógica de saída do tipo *Mealy* é função do estado atual e das entradas. Geralmente uma máquina de estados complexa pode ter saídas do tipo *Mealy* e *Moore*.

2.11.1 Exemplo de máquina de estados

A seguir será apresentado um exemplo de uma implementação de uma máquina de lavar em VHDL. Um diagrama de estados da máquina de lavar se encontra abaixo:

Figura 11 – Máquina de estados da máquina de lavar



Fonte: do autor

A máquina é gerida por 5 estados começando em *idle*. Ela recebe como entrada os sinais *empty*, *door_closed*, *full* e *heat_demand*, transicionando entre os estados de acordo com essas entradas. Na Figura 12 pode se ver a implementação do registrador de estado, juntamente com o registro de saídas. Fazer o registro das saídas é uma boa prática de codificação em VHDL.

O registrador de saídas quando houver um sinal de *reset* irá reiniciar os valores das saídas e do estado para valores padrões, já quando houver uma borda de subida de *clock* irá registrar os valores de saída e estados. A próxima parte do código é a lógica de próximo estado que se encontra na Figura 13 .

A lógica de próxima estado recebe como um valor padrão o registro do estado sempre que o estado for atualizado. A máquina de estados começa em *IDLE*, seguindo para *FILL* quando a porta da máquina for fechada, *door_closed*. Em *FILL* a máquina irá para *HEAT_W* quando ela estiver cheia, ou seja, recebeu a entrada *full*. Quando em

Figura 12 – Registro de estado e saída

```
41  -- Registrador de estado e saida
42  process (clk, reset_n)
43  begin
44    if (reset_n = '0') then
45      water_reg <= '0';
46      spin_reg  <= '0';
47      heat_reg <= '0';
48      pump_reg <= '0';
49      washing_state_reg <= IDLE;
50    elsif rising_edge (clk) then
51      water_reg <= water_next;
52      spin_reg <= spin_next;
53      heat_reg <= heat_next;
54      pump_reg <= pump_next;
55      washing_state_reg <= washing_state_next;
56    end if;
57  end process;
```

Fonte: do autor

Figura 13 – Lógica de Próximo estado

```
59  --lógica de próximo estado
60  process(washing_state_reg, door_closed, full, done, empty, heat_demand)
61  begin
62    -- default
63    washing_state_next <= washing_state_reg;
64
65    case washing_state_reg is
66
67      when IDLE =>
68        if (door_closed = '1') then
69          washing_state_next <= FILL;
70        end if;
71
72      when FILL =>
73        if (full = '1') then
74          washing_state_next <= HEAT_W;
75        end if;
76
77      when HEAT_W =>
78        if (heat_demand = '0') then
79          washing_state_next <= WASH;
80        end if;
81
82      when WASH =>
83        if (heat_demand = '1') then
84          washing_state_next <= HEAT_W;
85        elsif (done = '1') then
86          washing_state_next <= DRAIN;
87        end if;
88
89      when DRAIN =>
90        if (empty = '1') then
91          washing_state_next <= IDLE;
92        end if;
93
94      when others => null;
95    end case;
96  end process;
```

Fonte: do autor

HEAT será verificado se ainda há demanda de calor, entrada *heat_demand*, se não a máquina seguirá para *WASH*. Quando a o ciclo de lavagem estiver pronto, recebeu uma entrada *done*, e não houver demanda por aquecimento, entrada *heat_demand*, o estado sairá de *WASH* para *DRAIN*. Quando em *DRAIN*, após a máquina esvaziar, recebeu uma entrada *empty*, o estado retornará para o início em *IDLE*. Por fim temos a lógica de saída na Figura 14.

A lógica de saída irá atribuir o valor das saídas *water*, *spin*, *heat* e *pump*. Inicialmente é atribuído valores padrões para essas saídas caso nenhuma lógica infira

Figura 14 – Lógica de Saída

```
 98    --lógica de saida
 99    process(washing_state_reg, full, heat_demand, empty)
100    begin
101    --default
102     water_next <= '0';
103     spin_next <= '0';
104     heat_next <= '0';
105     pump_next <= '0';
106
107     case washing_state_reg is
108     when IDLE =>
109     water_next <= '0';
110     spin_next <= '0';
111     heat_next <= '0';
112     pump_next <= '0';
113
114     when FILL =>
115     if (full = '1') then
116        water_next <= '0';
117     else
118        water_next <= '1';
119     end if;
120
121     when HEAT_W =>
122     spin_next <= '1';
123     if (heat_demand = '0') then
124        heat_next <= '0';
125     else
126        heat_next <= '1';
127     end if;
128
129     when WASH =>
130     spin_next <= '1';
131
132     when DRAIN =>
133     if (empty = '1') then
134        spin_next <= '0';
135        pump_next <= '0';
136     else
137        spin_next <= '1';
138        pump_next <= '1';
139     end if;
140   end case;
141 end process;
```

Fonte: do autor

um valor a essas saídas. Em *FILL* se a entrada *full* estiver ativa, a saída *water* estará em nível lógico ato. Já em *HEAT_W*, se a entrada *spin* irá para nível lógico alto e se *heat_demand* não estiver ativo *heat* irá para o nível lógico baixo, caso contrário irá para nível lógico alto. Em *WASH* spin ficará em nível lógico alto. E por fim em *DRAIN* *spin* e *pump* ficarão em nível lógico alto apenas se *empty* estiver em nível lógico baixo.

Esse foi apenas um exemplo e existem muito mais detalhes sobre a semântica e sintaxe da linguagem VHDL, contudo ensinar VHDL não é o objetivo desse documento. Um detalhamento mais aprofundado sobre a linguagem pode ser encontrado em (CHU, 2006).

3 DESENVOLVIMENTO

3.1 REQUISITOS DO TRABALHO

A taxa de amostragem de saída de dados para aplicações em proteção e medidas em geral é de 4,8 kHz e para aplicações de medição de qualidade é de 14,4 kHz de acordo com a IEC 61869-9(IEC, 2019). A IEC 61869-9 define as especificações da interface digital para transformadores de instrumentação. Desta forma será adotado para esse projeto uma frequência de saída de 14,4 kHz. Para condições em regime permanente a IEC 61850-5(IEC, 2013) especifica que o *jitter* da resposta do sistema deve ser menor que 1 μs. Também quando houver a perda de um sinal de sincronização o sistema deve se comportar normalmente por no mínimo 5 segundos (IEC, 2019), quando do retorno do sinal de referência ele deve se ajustar sem ter grandes variações.

O sistema deve receber dois dos padrões mais comumente usados nesse tipo de aplicação, de modo a ter interoperabilidade entre instalações e aparelhos, IRIG-B e PPS.

Devido aos requisitos de baixa latência, velocidade de processamento e paralelismo que o sistema deve respeitar, além de sua flexibilidade e fácil integração com equipamentos propostos, serão utilizados dispositivos FPGA. Implementações em *Hardware* permitem um maior controle do sistema, com tempo de resposta mais curtos, se comparado a aplicações em *Software*. Para programar tais dispositivos será utilizado a linguagem VHDL pela familiaridade com a linguagem. Alguns requisitos do circuito digital devem ser atendidos, como:

- Respeitar os requisitos de *Timing* (50 MHz)
- Respeitar os limites de elementos lógicos e memória presentes no FPGA

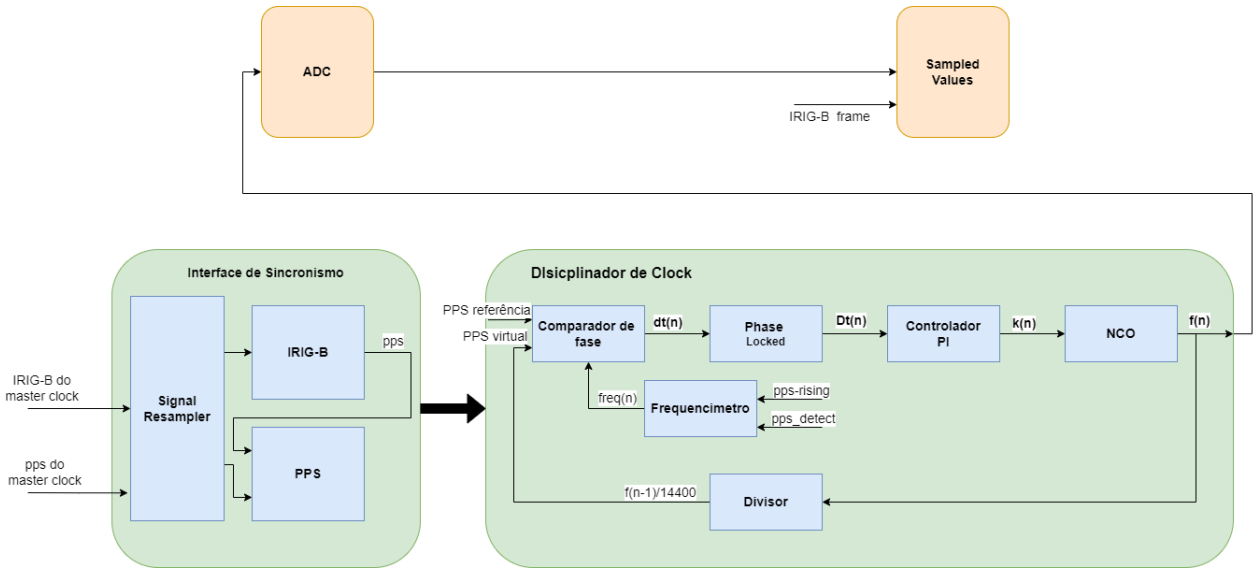
3.2 VISÃO GERAL

Visando os requisitos do projeto foi projetado um circuito digital para a tecnologia especificada. A Figura 15 apresenta uma visão geral da arquitetura que foi utilizada. Os elementos em laranja representam elementos que irão se aproveitar desses saídas em um equipamento, sendo assim não fazem parte do trabalho, mas apresentam uma contextualização.

3.3 ARQUITETURA

Em seguida serão apresentados os componentes individuais da arquitetura, bem como sua funcionalidade e implementação em lógica digital.

Figura 15 – Arquitetura do trabalho



Fonte: do autor

3.3.1 Interface de Sincronismo

Esse modulo tem por objetivo realizar a interface com os diferentes padrões e protocolos de sincronismo possíveis de forma a receber um pulso de referência. Nessa etapa devem ser processados os padrões IRIG-B e PPS, bem como inserir registradores para evitar metaestabilidade. Por fim será entregue um pulso de referência ao disciplinador de *clock*.

3.3.1.1 Signal Resampler

Esse módulo é de extrema importância para evitar problemas de metaestabilidade. Quando se trabalha com a recepção de sinais assíncronos, ou sinais provenientes de um domínio de *clock* diferente, pode aparecer o fenômeno da metaestabilidade que pode causar falha de sistema em dispositivos digitais. Para que não ocorra metaestabilidade, uma entrada de um registrador deve estar estável por um tempo mínimo antes da borda de um *clock* (conhecido como tempo de *setup* do registrador t_{su}) e por um tempo mínimo depois da borda de *clock* (conhecido como tempo de *hold* do registrador t_h). O registrador de saída então ficará disponível após um atraso especificado (conhecido como atraso clock-to-output t_{co}). Caso essas condições não sejam atendidas o registrador entra em um estado metaestável, onde a saída é indeterminada por certo período de tempo.

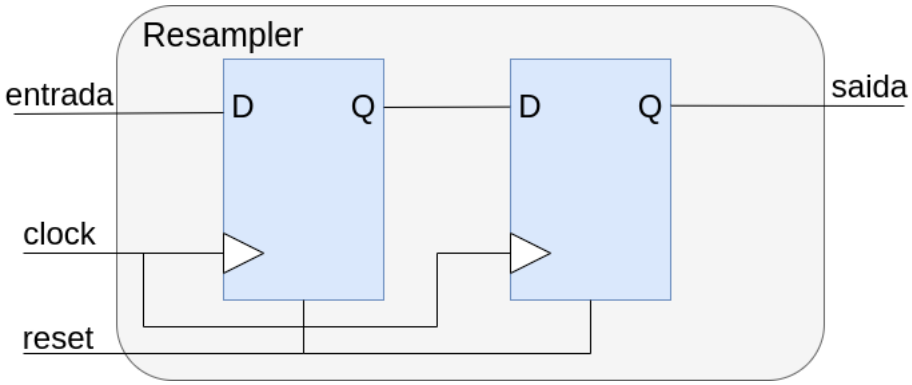
Quando em um estado metaestável, o registrador de saída transita entre um nível lógico alto e baixo por um certo tempo, o que significa que a saída é atrasada mais que o t_{co} especificado. Um sinal metaestável não necessariamente irá causar um

erro no circuito digital, se o dado de saída se resolver para um estado válido antes que o próximo registrador do circuito capture o dado, o circuito se resolve. Se o sinal não se resolver nesse tempo, pode ocorrer da lógica apresentar inconsistências, onde diferentes registradores capturam diferentes valores de saída do sinal metaestável. Mais sobre metaestabilidade pode ser encontrado em (ALTERA, 2022).

Como o sinal proveniente do GNSS é oriundo de um domínio de *clock* que não está relacionado com este circuito, não é possível garantir que o sinal vai respeitar os tempos de *setup* e *hold*. Uma forma de evitar esse comportamento indesejado é a utilização de uma cadeia de registradores. Esses registradores adicionam tempo para que o sinal potencialmente metaestável se resolva antes de ser utilizado pelo resto do circuito. Dessa forma conseguimos sincronizar o sinal de entrada ao domínio de *clock* do sistema e garantimos uma probabilidade muito maior de estabilidade.

Na arquitetura desse módulo foram declarados dois registradores em cadeia que irão registrar a entrada de sincronismo e entregar como saída um sinal sincronizado com o domínio de *clock* do circuito. Por ser um bloco simples não será entrado em detalhes sobre a sua implementação em VHDL. Um diagrama de blocos pode ser encontrado na Figura 16.

Figura 16 – Signal Resampler



Fonte: do autor

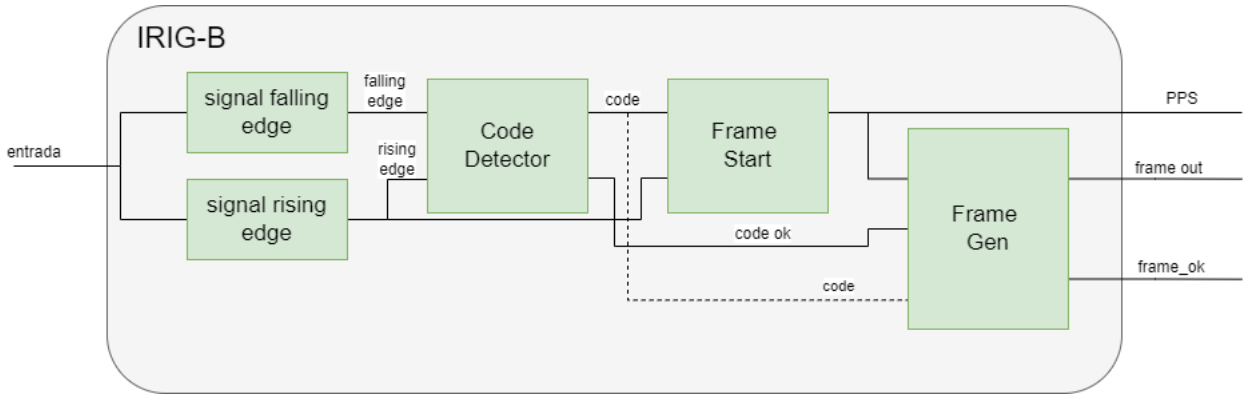
Essa arquitetura contudo adiciona um atraso no circuito que deve ser corrigido posteriormente, já que sem correção estaríamos referenciando a um pulso de entrada defasado em 2 ciclos de *clock*.

3.3.1.2 IRIG-B

Modulo de grande importância para a utilização do protocolo IRIG-B, ele irá desempacotar as informações seriais vinda do pacote IRIG-B e irá gerar um pulso de referência. Além do pulso de referência que será utilizado pelo disciplinador de *clock*, é muito importante decodificar a informação do *frame*. A informação temporal

contida nesse *frame* pode ser usada por exemplo por um processador para manter a informação de tempo dos seus processos. O *frame* também pode conter informação de qualidade do tempo que deve ser utilizada dentro dos *frames* de *Sampled Values*. O diagrama de blocos simplificado, sem os sinais de *clock* e *reset*, se encontra na Figura 17.

Figura 17 – Diagrama de blocos do modulo IRIG-B

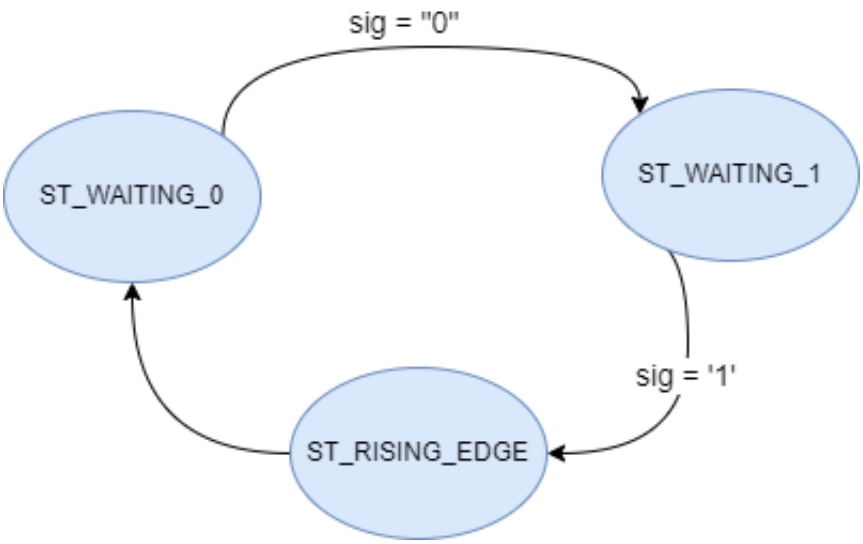


Fonte: do autor

3.3.1.2.1 Signal Rising Edge

A máquina de estados do módulo *Signal Rising Edge*, responsável por detectar bordas de subidas, é apresentada na Figura 18.

Figura 18 – Máquina de estados do Signal Rising Edge



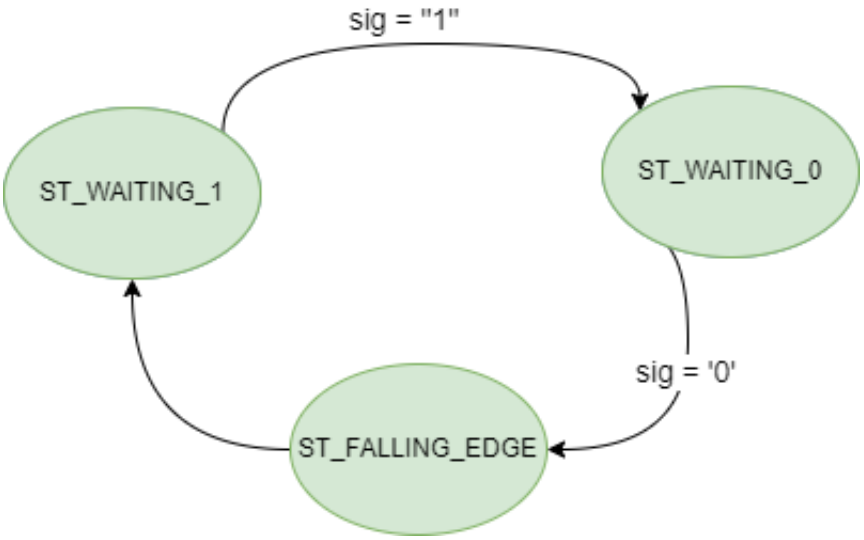
Fonte: do autor

Inicialmente, quando no estado ST_WAITING_0, o sistema espera que o sinal de entrada esteja em nível lógico baixo para ir para o estado ST_WAITING_1. No estado ST_WAITING_1 o sistema espera que o sinal de entrada esteja em nível lógico alto para ir para ST_RISING_EDGE onde será registrado um sinal de borda de subida e irá para o início da máquina de estados.

3.3.1.2.2 Signal Falling Edge

A implementação do detector de borda de descida é muito semelhante ao detector de borda de subida. A máquina de estados do módulo *Signal Falling Edge* se encontra na Figura 19

Figura 19 – Máquina de estados do Signal Falling Edge



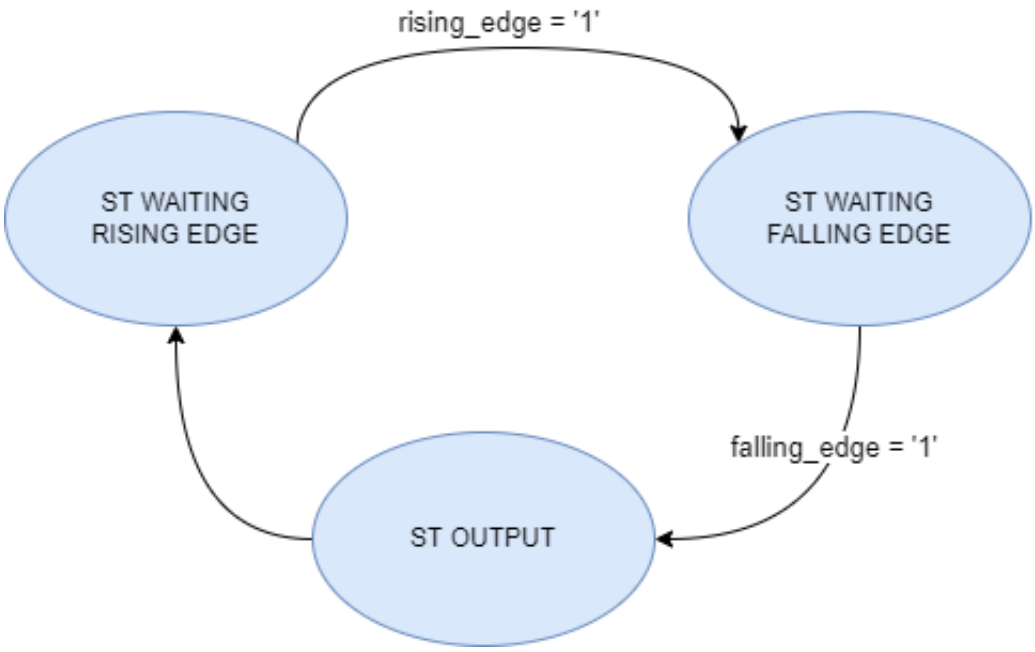
Fonte: do autor

Inicialmente, quando no estado ST_WAITING_1, o sistema espera que o sinal de entrada esteja em nível lógico alto para ir para o estado ST_WAITING_0. No estado ST_WAITING_1 o sistema espera que o sinal de entrada esteja em nível lógico baixo para ir para ST_FALLING_EDGE onde será registrado um sinal de borda de descida e irá para o início da máquina de estados.

3.3.1.2.3 Code Detector

O modulo *code detector* é responsável por identificar qual código está sendo transmitido pelo índice. O código é identificado por 2 bits, sendo que “00” representa o valor binário “0”, “01” o valor binário “1”, “10” o indicador de posição “P” e “11” é um pulso não identificado. A máquina de estados desse bloco se encontra na Figura 20.

Figura 20 – Máquina de estados do Code Detector



Fonte: do autor

A máquina de estados se inicia em ST WAITING RISING EDGE, enquanto ela se manter nesse estado o contador que irá contar a quantidade de pulsos de *clock* entre a borda de subida e descida do pulso de entrada é reiniciado para 0. Quando houver uma borda de subida esse *reset* é desabilitado e a máquina de estados irá prosseguir para ST WAITING FALLING EDGE. Nesse estado, quando houver uma borda de descida a máquina de estados irá para ST OUTPUT. Esse é o estado mais importante do modulo, nele será atribuído um novo valor ao registrador de código de acordo com o contador de pulsos de *clock* e com as constantes que vão identificar os códigos. Para cada código deve-se respeitar os limites superiores e inferiores, neles forem acrescidos um fator de segurança de acordo com a Tabela 1.

Tabela 1 – Limite inferiores e superiores do detector de códigos

Código	**Limite Inferior**	**Limite Superior**
0	(50000000MHz/100)*0.18 = 99000	(50000000MHz/100)*0.22 = 110000
1	(50000000MHz/100)*0.48 = 240000	(50000000MHz/100)*0.52 = 260000
P	(50000000MHz/100)*0.78 = 390000	(50000000MHz/100)*0.82 = 410000
Não identificado	outros	outros

Fonte: do autor

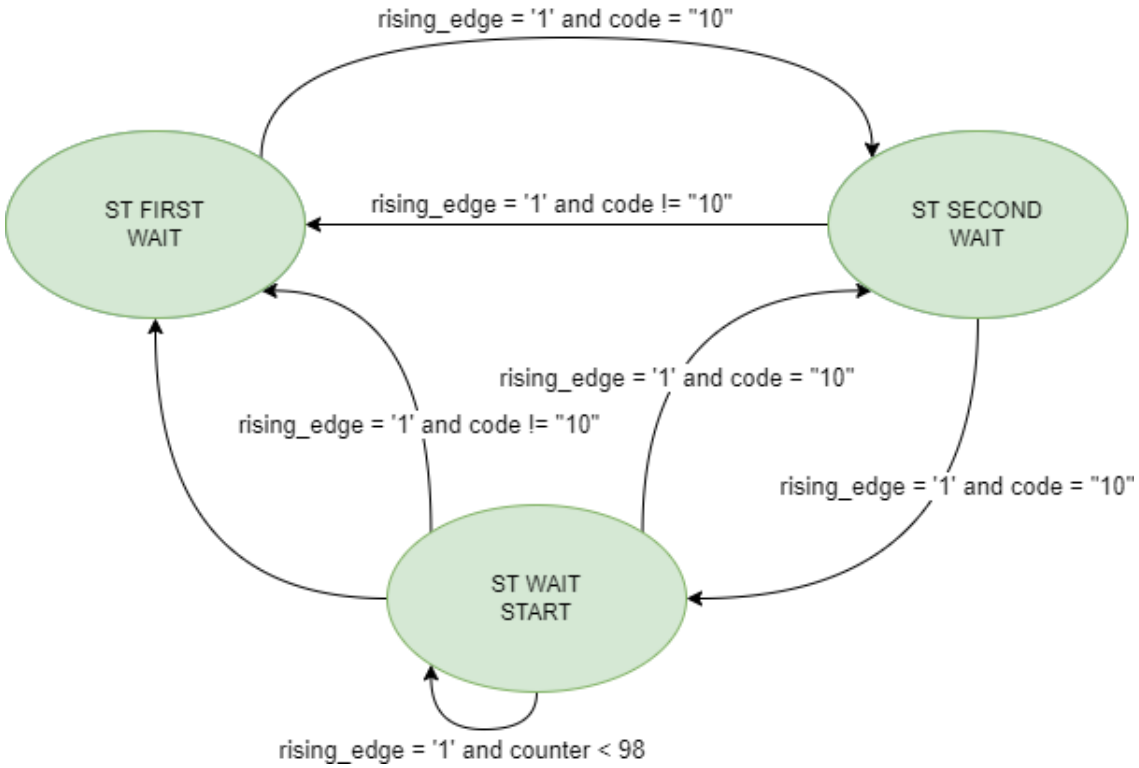
Também quando chega nesse estado o valor de code ok, identificando que tem um código novo vai para nível lógico alto, em todos os outros estados está em nível lógico baixo. A máquina de estados volta para ST WAITING RISING EDGE no próximo

ciclo de clock.

3.3.1.2.4 Frame Start

Frame Start é o modulo responsável por detectar o começo de um *frame* a partir dos códigos e de uma borda de subida do sinal de IRIG-B, gerando um sinal de PPS. Dois identificadores de posição identificam o começo de um *frame* e um PPS. Contudo depois que é identificado dois sinais de referência o tempo de PPS já passou, o PPS deve ser identificado na borda de subida do segundo P como na Figura 2. Dessa forma será detectado dois identificadores de referência consecutivos os quais não serão utilizados para geração de PPS, após serão contados a quantidade de bordas de subida de forma a prever a borda de subida de um segundo código P consecutivo. A máquina de estados que rege esse módulo se encontra na Figura 21.

Figura 21 – Máquina de estados do Frame Start



Fonte: do autor

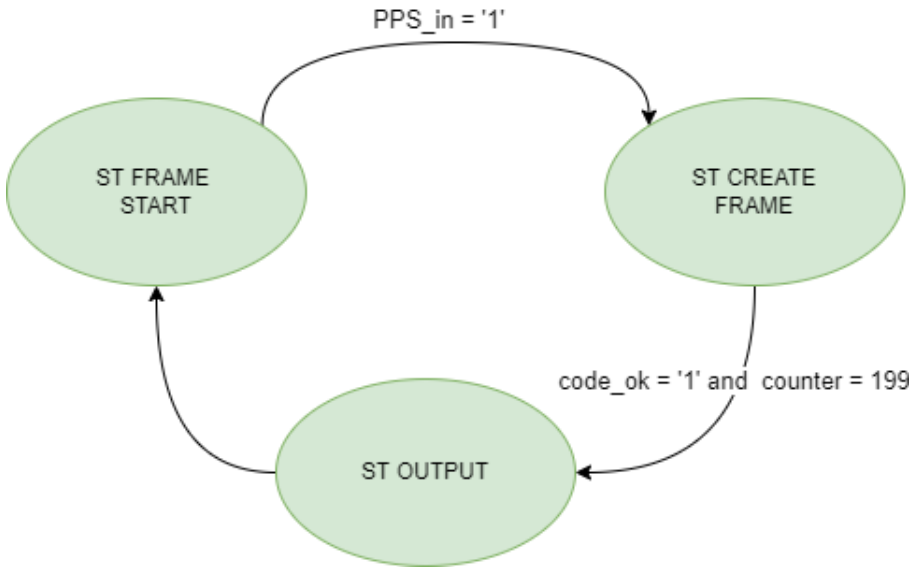
A máquina de estados se inicia em ST FIRST WAIT, a qual irá esperar uma borda de subida após ser detectado um código P, code = “10”, nessa condição a máquina de estados irá para ST SECOND WAIT. Nesse estado será verificado se o próximo código a ser recebido também é de referência, ou ele está no meio do *Frame*. Dessa forma quando receber uma borda de subida ele irá verificar se code = “10” para

ir para ST WAIT START, se code for qualquer outro valor ele irá retornar para ST FIRST WAIT. Esses dois estados iniciais irão garantir que ST WAIT START funcione em um começo de *Frame*. Em ST WAIT START um contador incrementará de 0 a 97 toda vez que houver uma borda de subida. Quando houver uma borda de subida e esse contador for maior que 97 será verificado o código entregue pelo detector de código. Se code = “10” será gerado um pulso de saída e a máquina de estados irá para ST SECOND WAIT para verificar se o próximo código realmente é um código P. Caso code seja qualquer outro valor a máquina de estados irá retornar para ST FIRST WAIT pois provavelmente ocorreu algum erro.

3.3.1.2.5 Frame Generator

Esse módulo tem como objetivo armazenar todos os 200 *bits* de um *frame* em um registrador para que a informação do *frame* possa ser posteriormente utilizada por outros processos. Esse módulo recebera o sinal PPS de *Frame Start*, *code* e *code ok* de *Code Detector*. Na Figura 22 se encontra a máquina de estados que rege esse módulo.

Figura 22 – Máquina de estados do Frame Generator



Fonte: do autor

O estado inicial é o ST FRAME START que irá esperar um sinal de PPS vindo de FRAME START para prosseguir para o estado ST CREATE FRAME. Nesse estado dois registradores, *frame* de 200 *bits* e counter que vai de 1 a 199, irão gerar o frame IRIG-B. Toda vez que receber um sinal code_ok de Code Detector o contador irá incrementar em 2 e o registrador *frame* irá atualizar 2 de seus 200 *bits* de acordo com (5)

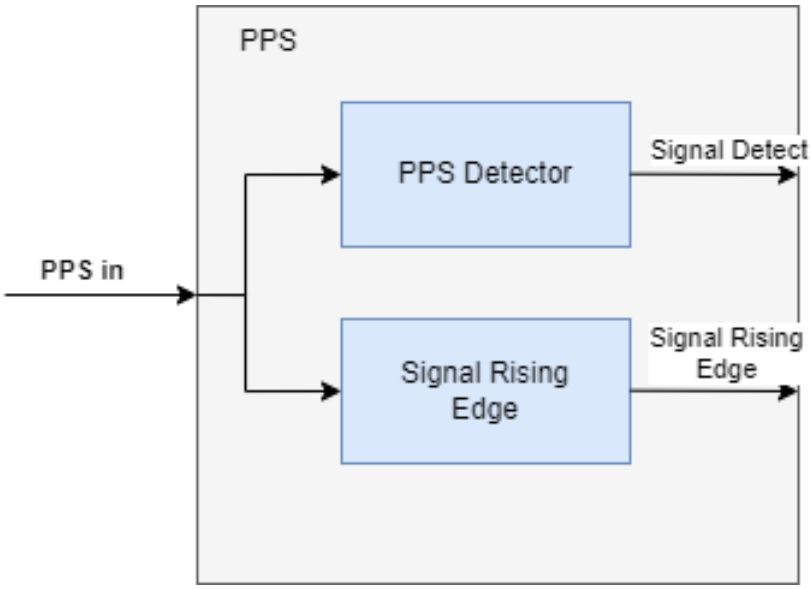
$$frame(counter\ downto\ counter - 1) <= code \tag{5}$$

Quando o contador chegar em 199, o valor do *frame* será atualizado normalmente, contudo o contador irá receber o valor 1 e irá para o estado ST OUTPUT. Nesse estado um registrador irá armazenar esse valor de *frame* até que volte novamente para esse estado de forma a atualizar o *frame*, também é enviado um pulso frame_ok informando que o valor do *frame* foi atualizado.

3.3.1.3 PPS

Esse bloco é constituído por um detector de PPS que irá detectar se o sinal de entrada é um sinal de PPS válido e um detector de borda de subida. O sinal de borda de subida tem por finalidade identificar o começo do sinal do PPS para sincronizar e o detector irá solicitar que o NCO funcione apenas quando o PPS estiver ativo e válido. Abaixo um diagrama de blocos simplificado, sem os sinais de *clock* e *reset*, é apresentado na Figura 23.

Figura 23 – PPS



Fonte: do autor

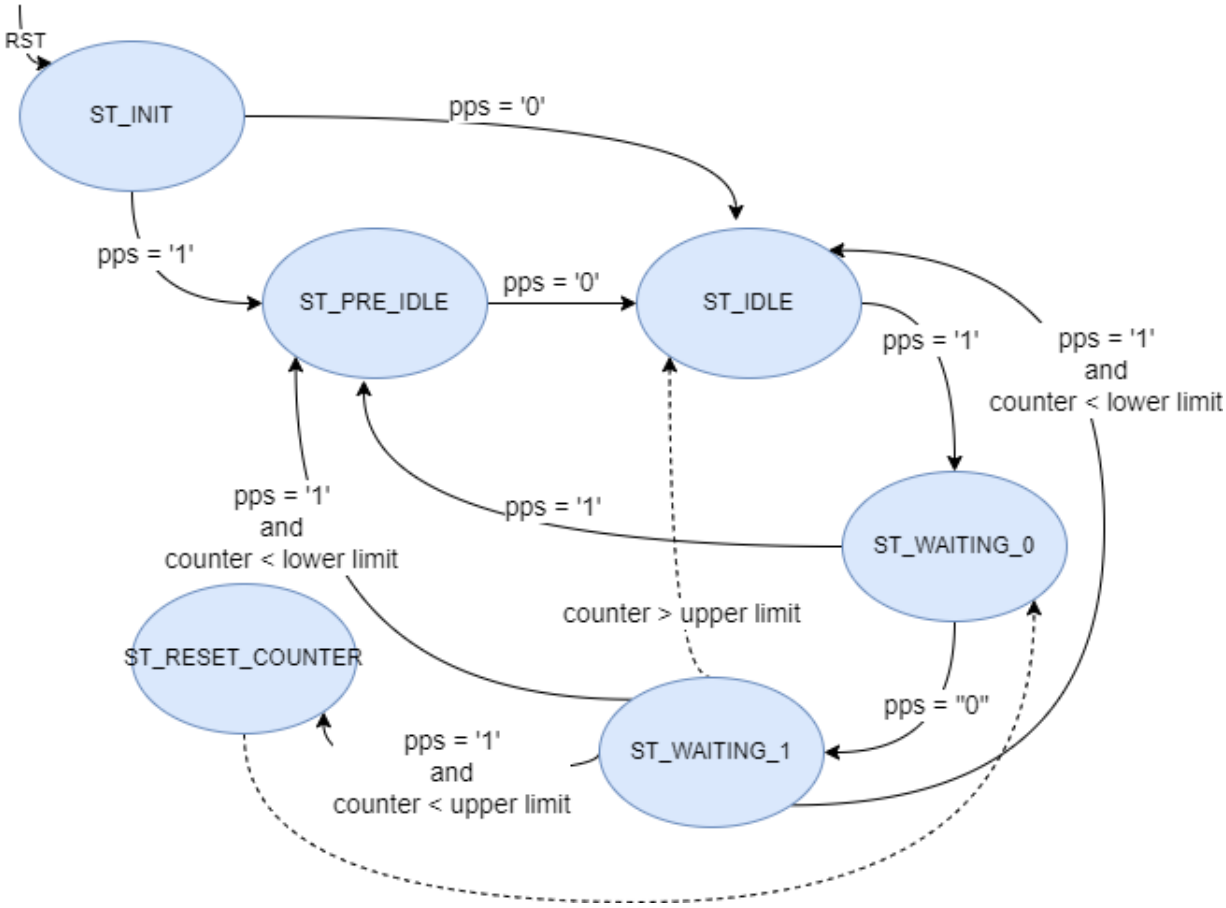
3.3.1.3.1 Signal Rising Edge

Esse módulo é o mesmo utilizado no modulo IRIG-B e por isso não será explicado novamente.

3.3.1.3.2 PPS Detector

O modulo PPS detector irá externar um sinal de ativo enquanto o número de ciclos registrados durante um ciclo de PPS estiver dentro de uma faixa aceitável. A máquina de estados do modulo se encontra na Figura 24.

Figura 24 – PPS Detector



Fonte: do autor

Um contador é incrementado a cada ciclo de *clock* de forma a registrar a quantidade de ciclos de *clock* contido em um PPS. A máquina de estados inicia no estado ST_INIT e irá para ST_PRE_IDLE caso o PPS esteja em nível lógico alto e irá para ST_IDLE caso esteja em nível lógico baixo. Quando estiver em ST_PRE_IDLE ele ficará nesse estado até que PPS esteja em nível lógico baixo. Em ST_IDLE ele irá para ST_WAITING_0 quando PPS estiver em nível lógico alto, o começo dessa máquina de estados tem por finalidade garantir uma borda de subida de PPS válida. Quando em ST_WAITING_0 o contador começará a contar e irá para ST_WAITING_1 quando tiver um sinal de PPS com nível lógico baixo. Caso enquanto estiver nesse estado o contador extrapolar o limite superior ele retornará para o estado ST_PRE_IDLE para reiniciar o processo de contagem. Em ST_WAITING_1 ele irá para ST_RESET_COUNTER

caso o PPS tenha um nível lógico alto e nenhum limite tenha sido extrapolado ele irá para o estado ST_RESET_COUNTER que irá reiniciar o contador e irá voltar para o estado ST_WAITING_0. Caso algum dos limites tenha sido extrapolado enquanto em ST_WAITING_1 a máquina de estados irá para ST_PRE_IDLE ou ST_IDLE.

Os limites são estipulados por constantes, podendo se escolher qual é a precisão desejada. Nesse caso será utilizado um limite de 1,1(55000000 ciclos) e 0.9 (45000000 ciclos). O PPS estará ativo enquanto os estados estiverem em ST_WAITING_0, ST_WAITING_1 ou ST_RESET_COUNTER.

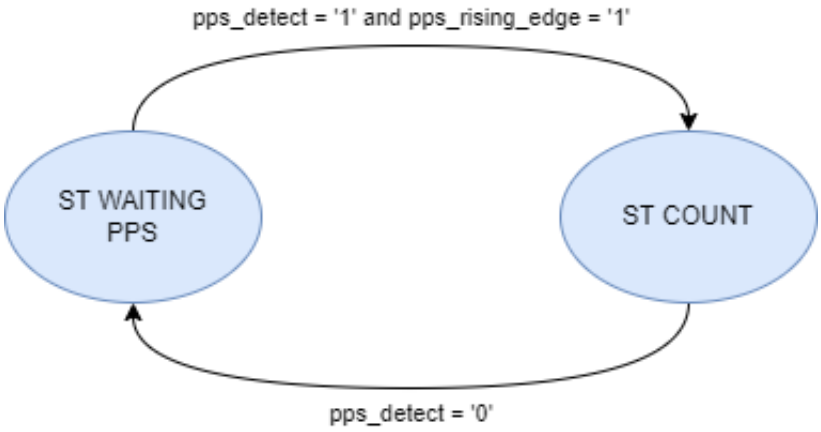
3.3.2 Disciplinador de Clock

O Bloco de disciplinador de *clock* é o mais importante do sistema, é quem irá gerar um *clock* com frequência mais baixa para aquisição a partir de um sinal de sincronismo. Sua arquitetura é similar a de (MACHADO MONTEIRO; SOUZA; DALMAS, 2016) tendo seus elementos principais um Comparador de fase, um controlador, um NCO e um divisor.

3.3.2.1 Frequencímetro

O frequencímetro é o bloco responsável por contar a quantidade de pulsos de *clock* dentro do intervalo de pulsos de sincronização de um segundo, desta forma fornecendo a frequência do oscilador. o diagrama da máquina de estados é apresentado na Figura 25

Figura 25 – Máquina de estados do Frequencímetro



Fonte: do autor

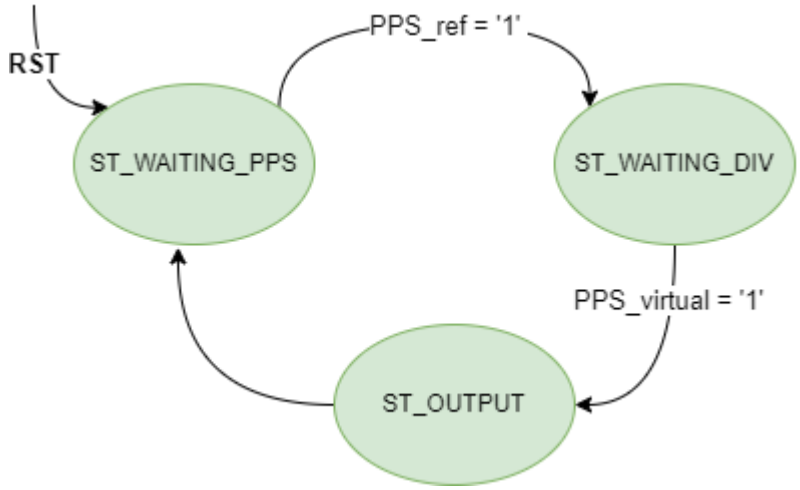
Sua máquina de estados possui apenas dois estados, começando em ST WAITING PPS. Nesse estado ele espera o sinal pps_detect do PPS Detector, indicando que o PPS está correto, e também uma borda de subida para ir para ST COUNT. Esse

estado é necessário para verificar o estado do PPS bem como iniciar a contagem em uma borda de subida. A seguir, a máquina de estados se manterá em ST COUNT até que por algum motivo o sinal pps_detect vá para nível lógico baixo. Enquanto estiver nesse estado um contador será incrementado a cada ciclo de clock, caso haja um sinal de borda de subida de PPS o contador será reiniciado para 1 e um registrador contendo a frequência medida será atualizado com o valor do contador.

3.3.2.2 Comparador de Fase

Esse módulo irá calcular a diferença de fase entre o PPS virtual e o PPS referência. Ele é regido por uma máquina de estados, apresentada na Figura 26 e um processo que irá calcular se a diferença será um valor positivo ou negativo, ou seja ele irá se ajustar para o pulso mais próximo.

Figura 26 – Máquina de estados do Comparador de Fase



Fonte: do autor

A máquina de estados inicia em ST_WAITING_PPS o qual irá esperar uma borda de subida de PPS referência e irá para ST_WAITING_DIV. Nesse estado ele irá iniciar um contador que irá incrementar a cada ciclo de *clock*, quando houver um sinal de pps virutal ele irá para ST_OUTPUT. É importante ressaltar que o contador não irá reiniciar para 0 e sim como 2 de forma a compensar o atraso adicionado pelos 2 registradores no sincronizador. Em ST_OUTPUT o contador será reiniciado, será enviado um valor de diferença, um valor de *enable* será ativado e a máquina de estados irá voltar para o início.

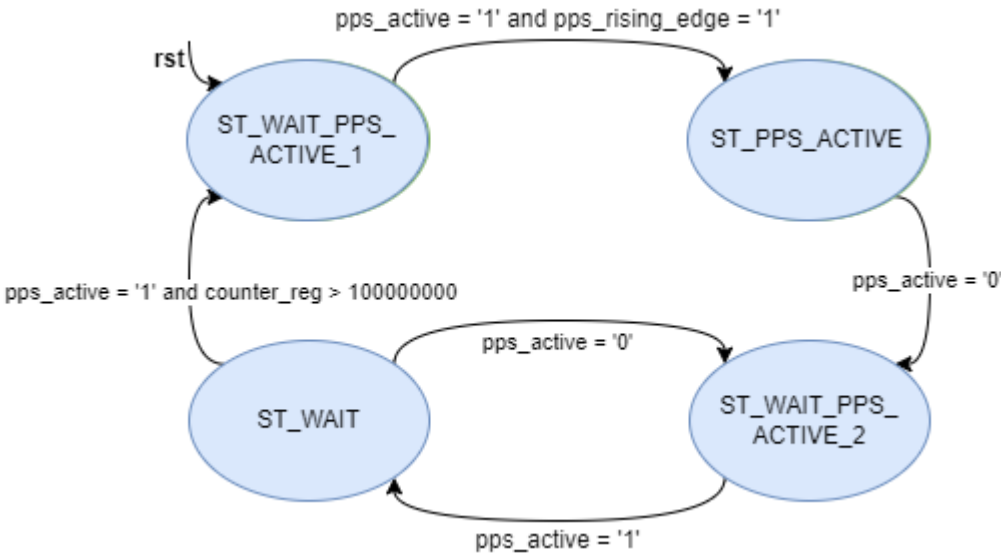
No processo de positivo e negativo, quando houver um sinal de *enable* ele irá verificar se o contador é maior que metade da frequência medida pelo frequencímetro, nesse caso o valor de diferença de saída será negativo. Caso o contrário, o valor de

saída de diferença será positivo. Essa distinção é necessária para decidir qual pulso é o mais perto para ele se ajustar rapidamente.

3.3.2.3 Phase Locked

Esse módulo tem como objetivo aumentar a robustez do sistema em casos de ruído no sinal sincronismo ou perda desse sinal. Na eventualidade do PPS não estar ativo, o valor de erro de fase se manterá constante por alguns segundos até que o sinal de PPS ativo retorne. Na Figura 27 se encontra a máquina de estados desse módulo.

Figura 27 – Máquina de estados do Phase Locked



Fonte: do autor

A máquina de estados se inicia em ST_WAIT_PPS_ACTIVE_1 e vai para o próximo estado quando houver um sinal de PPS ativo e uma borda de subida de PPS. Quando em ST_PPS_ACTIVE, o sinal de erro de fase de entrada é a saída do módulo, operando nesse estado até que o sinal PPS ativo entre nível lógico baixo, sinalizando uma perda de sinal. Fora desse estado, o valor de erro de fase de saída será sempre o mesmo valor registrado da última vez em que o PPS estava ativo. O estado ST_WAIT_PPS_ACTIVE_2 irá aguardar até que o sinal de PPS ativo retorne, indo para ST_WAIT. Nesse estado um contador irá ativar e esperar 2 segundos, 100000000 ciclos de *clock*, de PPS ativo para ir para ST_WAIT_PPS_ACTIVE_1 e prosseguir com seu funcionamento normal. Caso o sinal de PPS ativo ir para nível lógico baixo em algum momento enquanto estiver nesse estado, ou seja, o sinal de PPS não está adequado, ele irá retornar para ST_WAIT_PPS_ACTIVE_2 para identificar um novo sinal de PPS ativo válido.

3.3.2.4 Controlador PI

Esse bloco corresponde ao controlador PI que irá gerar o valor de K usado pelo NCO para gerar a frequência de amostragem. A parte principal deste módulo é dado pela Equação (6).

$$K\_next = K\_bias + k\_prop + int\_sum \tag{6}$$

O próximo valor de K que será externado ao NCO é o somatório da estimativa inicial de K que é o K_bias, o ganho proporcional e o ganho integral. O valor de k_bias é dado pela Equação (3), remanejando as variáveis temos a Equação (7).

$$K = \frac{Fout(2^{nbits})}{clockfreq} \tag{7}$$

Trocando as variáveis em (7) pelos valores utilizados temos (8):

$$K = \frac{14400(2^{42})}{50000000} = 1266637395 \tag{8}$$

O valor k_prop é o valor da diferença de fase, entregue pelo comparador de fase multiplicado pela constante de ganho proporcional. Para fazer a multiplicação, ou divisão pelo comparador de fase é feito simplesmente um *shift* de *bits* desse valor para esquerda ou para a direita, dependendo se a operação é de multiplicação ou divisão, acrescido de uma extensão de sinal. Como essa operação é realizada com *shift* de bits, a constante de ganho será sempre uma potência de 2.

Nessa equação int_sum, que é a parte integral da equação, começa zerado, sendo ativado somente após alguns segundos e quando tiver um erro constante entre pulsos consecutivos de PPS, essa integral condicional evita que o ganho integral tenha uma influência muito grande no início deixando o sistema instável. O valor do erro constante é dado pela diferença entre o valor do comparador de fase do ciclo anterior e do ciclo atual, se esse valor for 0, quer dizer que alcançou um erro constante. Um erro constante indica que o PPS virtual e PPS referência estão com a mesma frequência, porém seus pulsos ainda podem estar defasados, a atuação da parte integral irá corrigir essa defasagem. O mesmo contador utilizado para barrar o funcionamento da parte integral no começo dos tempos é reiniciado toda vez que houver um valor de erro constante. Essa condição foi adicionada porque em testes pode se observar algumas situações em que ocorria dois sinais de erro constante consecutivos, ocasionando em uma influência muito grande da parte integral e um *overshooting*. O valor de int_sum, quando atingido suas condições é dado pela soma do valor interior mais o valor atual dado pelo comparador de fase. Da mesma forma que na parte proporcional, o valor do comparador de fase é multiplicado ou dividido por meio de *shift* de *bits*. Essa parte do código é dada na Figura 28.

Onde int_sum_reg é o valor anterior, padding_int é a extensão de sinal para shift do comparador de phase, phase_comp_reg o valor do comparador de fase, er-

Figura 28 – Código do int sum

```
int_sum_next <= (int_sum_reg + signed(padding_int & std_logic_vector(phase_comp_reg(31 downto DIV_INT))))
                when ((error_const_reg = '1') and (counter_reg = 7)) else
                int_sum_reg;          -- Integral Gain is only activated
                                      -- after a certain time and when the
                                      -- error is constant
```

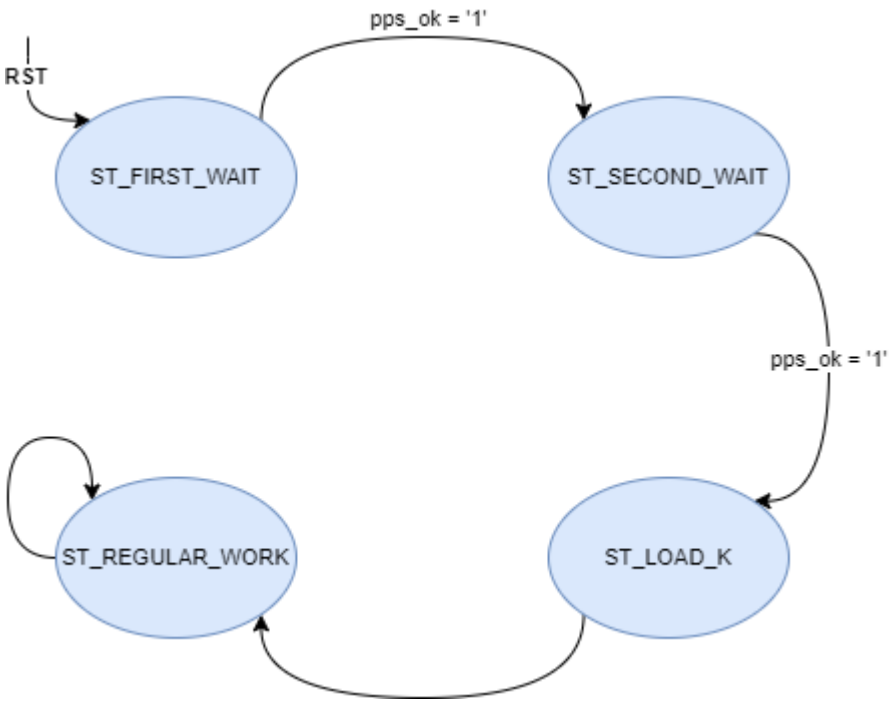
Fonte: do autor

ror_const_reg o sinalizador de erro constante e counter_reg o contador que barra a atualização no começo dos tempos ou logo depois que houver uma sinalização de erro constante. Nesse caso 7 sinaliza uma espera de 7 segundos.

3.3.2.5 NCO

Após gerado o valor de K, o módulo NCO é encarregado de gerar a frequência síncrona desejada. A cada ciclo de *clock* um registrador NCO irá incrementar com o valor de K até transbordar, gerando assim uma rampa. O bit mais significativo desse registrador irá gerar um *clock* síncrono da frequência desejada. A máquina de estados desse módulo se encontra na Figura 29

Figura 29 – Numerically Controlled Oscilator



Fonte: do autor

O sinal pps_ok é dado pela operação lógica *and* entre o sinal de pps_active e pps_rising_edge. Ou seja, enquanto o detector de PPS estiver ativo e ocorrer uma

borda de subida de PPS referência o sinal estará com nível lógico alto. A parte principal desse módulo é o acumulador dado na Equação (9).

$$nco\_next = nco\_reg + fcw\_reg \tag{9}$$

Os três sinais são de 42 bits, correspondendo a precisão desejada do NCO. O novo valor do registrador do NCO será dado pelo valor antigo mais o valor de K entregue pelo controlador PI. No início dos tempos e quando a máquina de estados é reiniciada esse valor de K é iniciado com uma constante. Essa constante é dada pela Equação (7)

Os estados iniciais ST_FIRST_WAIT e ST_SECOND_WAIT esperam um sinal de pps_ok para ir para o próximo estado de forma a garantir que a tenha ocorrido dois sinais de PPS válidos e estáveis para ir para o estado ST_LOAD_K. Nesse estado é registrado um novo valor de K dado pelo controlador, substituindo a constante que tinha sido usada para o início. No próximo ciclo de *clock* a máquina de estado vai para ST_LOAD_K e fica preso nesse estado a não ser que ocorra um sinal de *reset*. Nesse estado toda vez que houver sinal de pps_ok, ou seja, ocorreu um pulso de PPS válido, fcw será atualizado com o valor mais recente de K do controlador.

A saída desse módulo será dada pela Equação (10)

$$output\_nco = nco\_reg(41) \tag{10}$$

Ou seja a saída será o bit mais significativo do acumulador do NCO, gerando assim uma onda quadrada.

3.3.2.6 Divisor

O bloco divisor é utilizado para retornar a frequência de 14,4 kHz para 1 Hz, criando uma referência virtual que irá realimentar o sistema, como dado pela Equação (11).

$$pps\_virtual(n) = f(n-1)/14400 \tag{11}$$

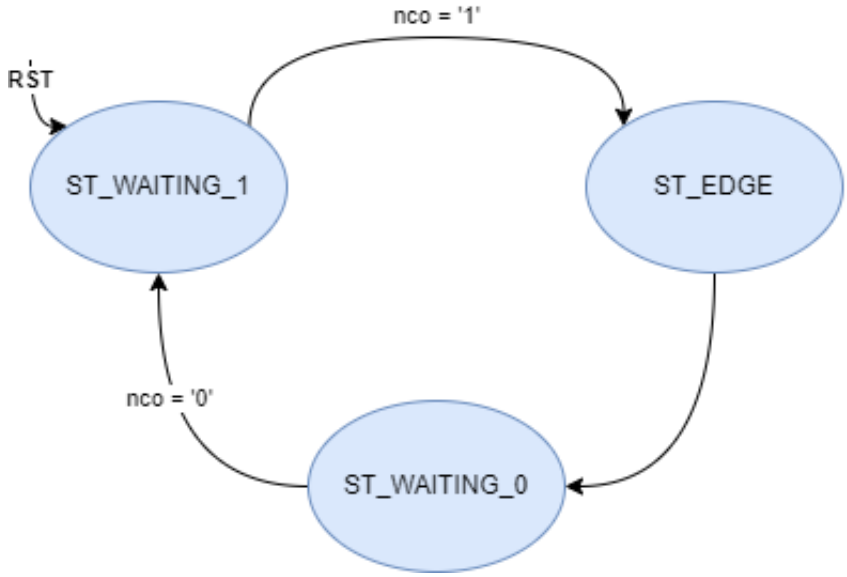
A parte principal desse módulo é um contador inteiro de 0 até a frequência desejada, nesse caso 14400. Quando o contador chegar em FREQ um pulso será enviado como saída como mostrado no código (12),

$$div\_pulse\_next <= '1' \ when \ counter\_reg = FREQ \ else \ '0'; \tag{12}$$

A máquina de estados que rege esse módulo se encontra na Figura 30 abaixo.

Iniciando em ST_WAITING_1, ele espera uma borda de subida positiva do *clock* de 14400 Hz gerado pelo NCO para ir para ST_EDGE. Nesse estado será incrementado o contador do divisor e no próximo estado irá para o estado ST_WAITING_0. O

Figura 30 – Máquina de estados do Divisor



Fonte: do autor

estado irá esperar uma borda de descida do clock de 14400 gerado pelo NCO para retornar para ST_WAITING_1 e reiniciar a máquina de estados.

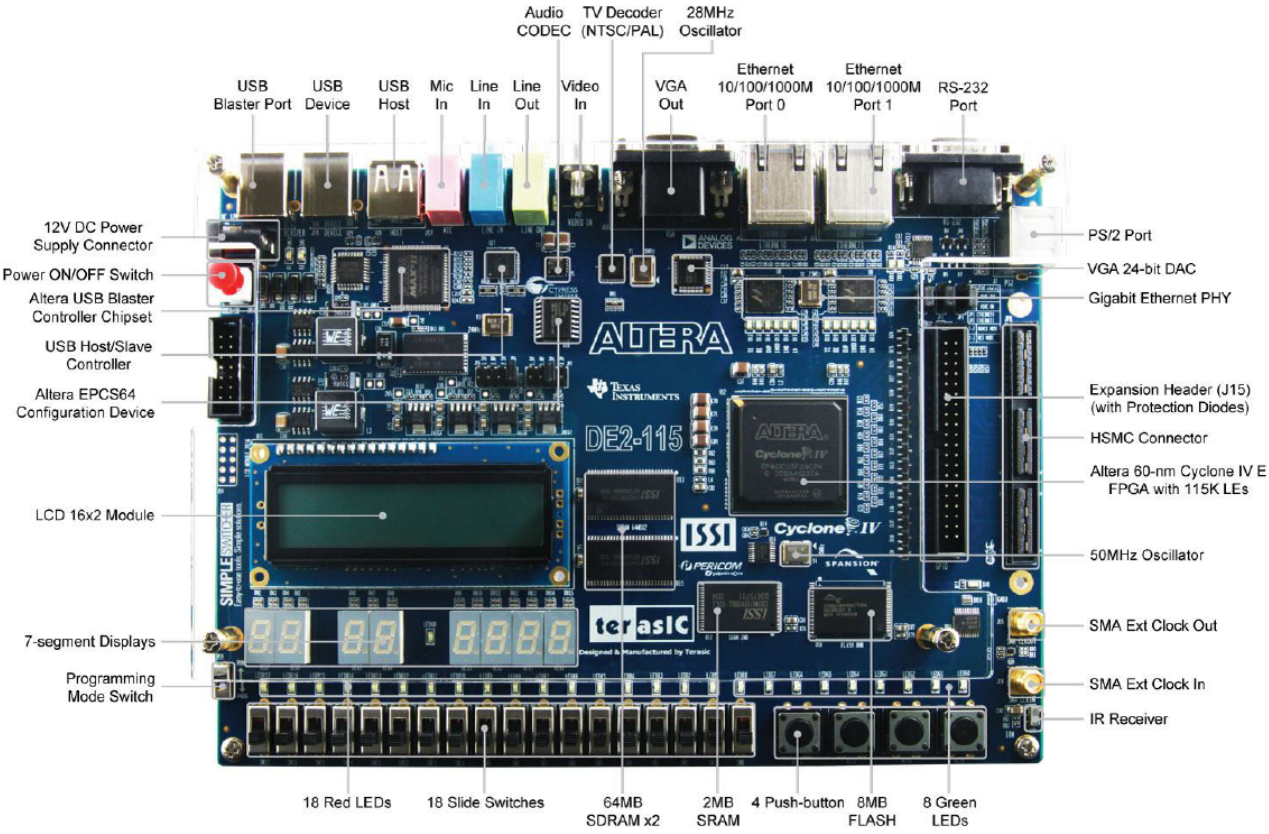
4 IMPLEMENTAÇÃO, TESTES E RESULTADOS

Nesta seção serão apresentados os testes feitos durante o decorrer do trabalho, bem como os resultados obtidos. Será apresentado também as ferramentas e softwares utilizados para a obtenção dos resultados

4.1 PLACA DE DESENVOLVIMENTO

Para os testes foi utilizado o Kit de desenvolvimento da Altera DE2-115(TERASIC, 2012). A placa contém um FPGA Altera Cyclone IV EP4CE115, um oscilador de cristal 50MHz para *clock*, além de diversos periféricos como *switches*, *LEDS*, *GPIO's* e muitos outros periféricos para *Debug* e validação de lógicas digitais.

Figura 31 – Kit de Desenvolvimento DE2-115



Fonte: (TERASIC, 2012)

Para programar a placa é utilizada uma *USB blaster*, usando ambos *JTAG* e *Active Serial* (AS). Mas para tanto é necessário a utilização da ferramenta *Programmer* contida no *software Intel Quartus Prime*.

4.2 RECEPTOR GNSS RT 430

Como fonte de sincronismo foram utilizados os receptores GNSS RT 430 da GE Grid Solutions. Tais relógios são referenciados aos satélites GPS e GLONASS e foram projetados para operações de grande porte como subestações elétricas para geração, transmissão, distribuição e sistemas de controle (GE, 2021). São suportados diversos tipos de sincronização como:

- PTP de acordo com IEEE 1588v2:2008;
- Opera como PTP *Master* ou *Ordinary Clock*;
- PTP *Power Profile*, de acordo com a mais nova norma IEEE C37.238:2017 e sua versão antecessora de 2011;
- PTP Perfil de *Power Utility Automation*, de acordo com a norma IEC 61850-9-3:2016;
- NTP/SNTP;
- IRIG-B004 (não modulado);
- IRIG-B124 (Modulado);
- DCF77;
- Datagrama serial;
- Pulsos de baixa frequência, como PPS, PPM e outras opções configuráveis.

Figura 32 – Vista Frontal RT430

Fonte: (GE, 2021)

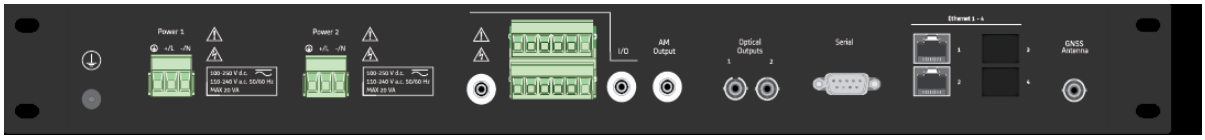
Os Relógios de precisão GNSS possuem uma precisão de 50 ns para IRIG-B e PPS, já para o protocolo IEEE 1588v2 PTP, sua precisão é melhor que 100 ns.

O equipamento possui para sincronização:

- Duas saídas elétricas TTL com conector BNC para sincronização, uma delas isolada;
- Duas saídas elétricas TTL com conector de parafuso de nível TTL para sincronização, uma delas isolada;
- Duas saídas de coletor aberto para sincronização;
- Uma saída modulada em amplitude IRIG-B124 para sincronização;
- Duas saídas ópticas com conectores ST para sincronização;

- Duas portas de comunicação de rede Ethernet

Figura 33 – Vista Traseira RT430

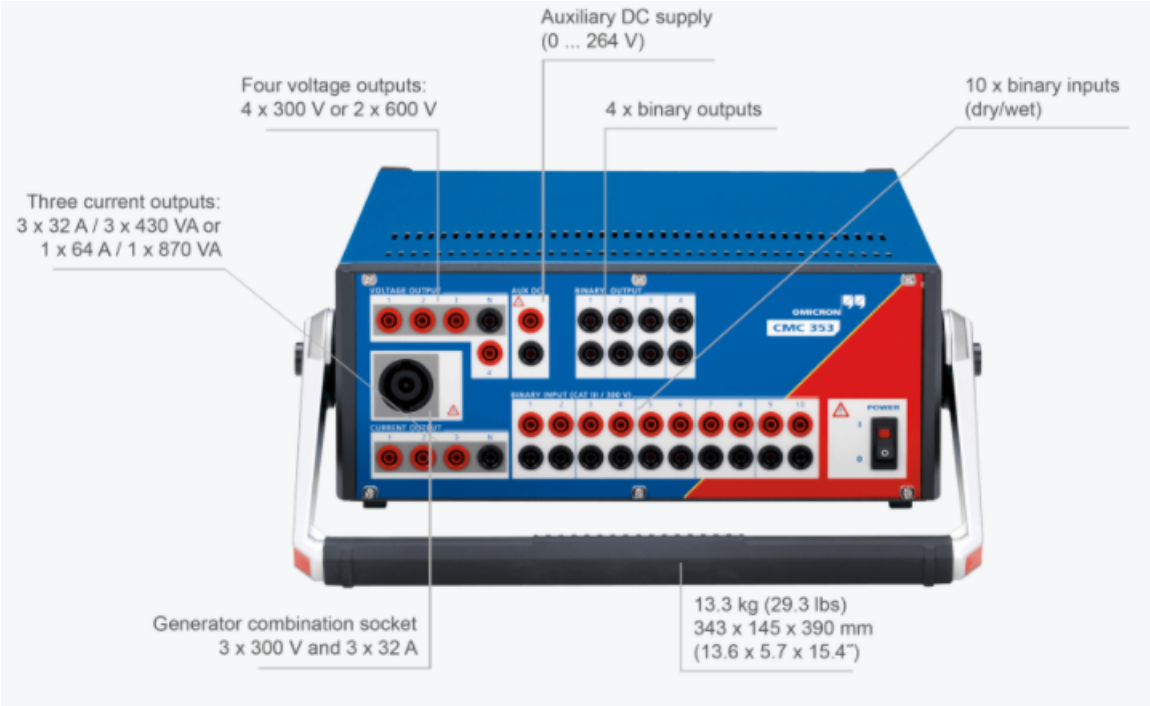


Fonte: (GE, 2021)

4.3 OMICRON CMC 353

O conjunto de testes trifásico CMC 353 é uma solução para testes de sistemas trifásicos, ideal para testes de proteção e para o comissionamento de sistemas SCADA (OMICRON, 2022a). O equipamento conta com diversas saídas de tensão, corrente e entradas e saídas binárias que podem ser utilizadas em uma vasta gama de aplicações. O conjunto de teste pode ser controlado por um computador ou por um tablet Android por um cabo Ethernet/USB ou por Wi-Fi.

Figura 34 – Vista Frontal CMC353



Fonte: (OMICRON, 2022a)

O equipamento também conta com diversos softwares, como o QuickCMC para controle manual de testes em relés de proteção, que podem ser utilizados e adqui-

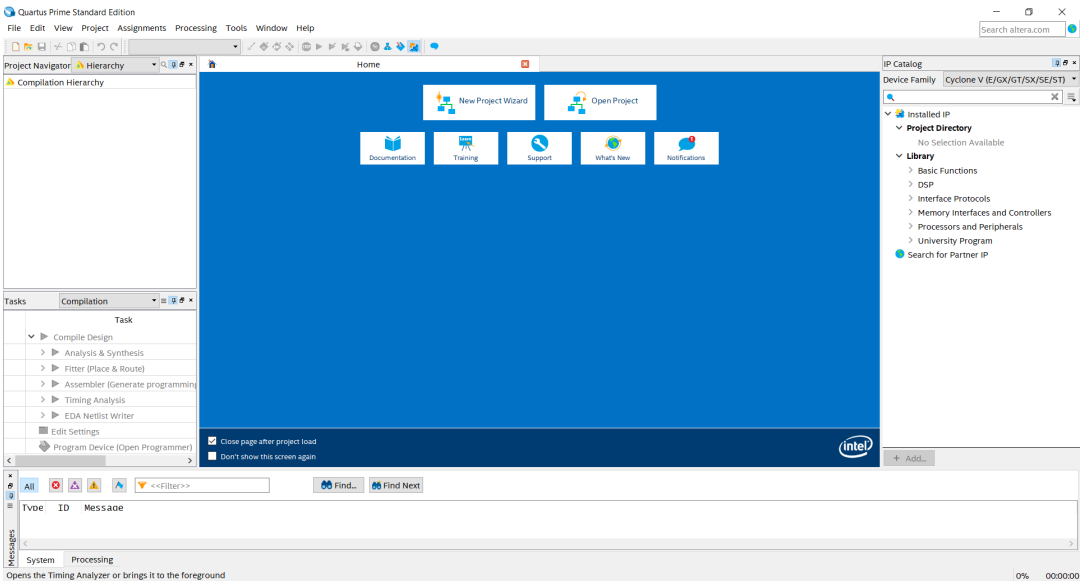
ridos individualmente para controlar o conjunto de testes, dependendo da aplicação desejada.

4.4 *SOFTWARES* UTILIZADOS

4.4.1 Intel Quartus Prime

Para o desenvolvimento de projetos em FPGA da marca Intel se fez necessário o uso da IDE Intel Quartus Prime(INTEL, 2021b). Intel Quartus Prime é um *software* de *design* produzindo pela Intel utilizado para desenvolvimento em FPGAs, SoCs e CPLDs produzidos pela Intel. após a recente aquisição da Altera pela Intel, o *software* foi renomeado de Altera Quartus II para o nome atual. Quartus proporciona o design, verificação, optimização e simulação de designs em HDL. O *software* disponibiliza a utilização de diferentes ferramentas e *features* como: SOPC builder, Qsys, DSP builder, uma vasta biblioteca de IPs e outros. Suporta VHDL, Verilog e System Verilog. Possui versões gratuitas e pagas para Windows e Linux.

Figura 35 – Ambiente do Intel Quartus Prime



Fonte: do autor

4.4.2 Modelsim

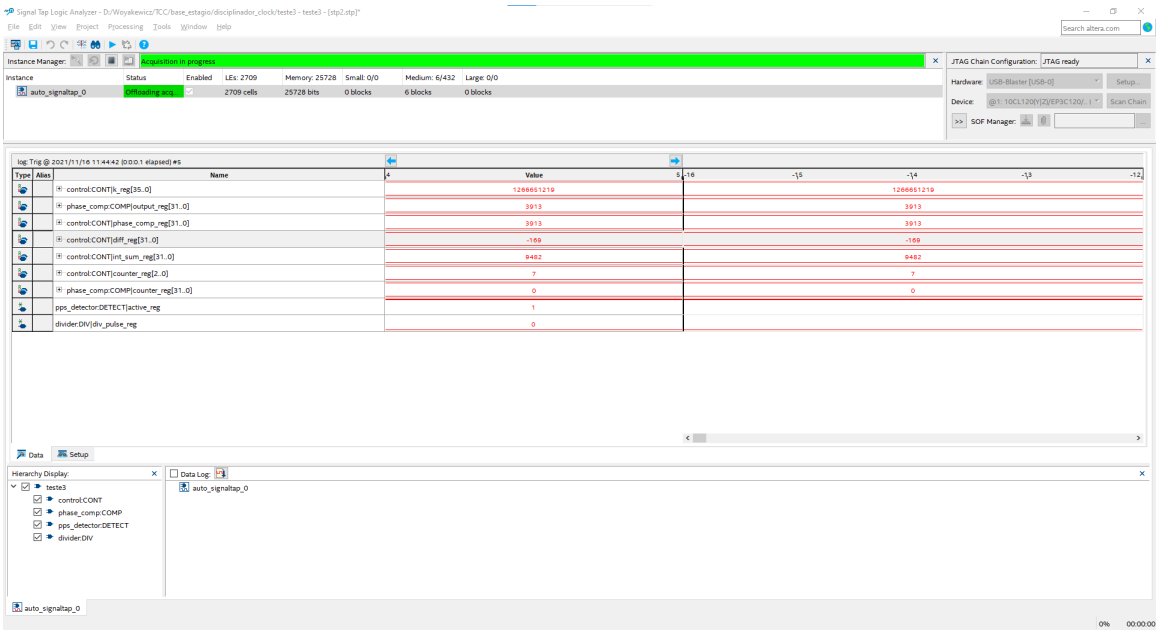
Modelsim(MENTOR, 2021) é um ambiente de simulação em HDL multi-linguagem desenvolvido Pela Mentor Graphics. Ele pode ser utilizado independentemente ou em conjunto com os softwares de desenvolvimento Intel Quartus Prime, Xilinx ISE ou Xilinx Vivado. O Software proporciona simulação comportamental, *test benches* em HDL e

TCL *scripting*. Possui suporte para as linguagens VHDL, Verilog, System Verilog, PSL e SystemC. Possui versões gratuitas e pagas.

4.4.3 Signaltap Logic Analyzer

Signaltap(INTEL, 2021a) é uma ferramenta da IDE Intel Quartus Prime que pode ser utilizada para visualizar formas de ondas do circuito digital em tempo real. Ela funciona conectando *probes* em diversas localidades do circuito para obter esses sinais, consequentemente aumentando a utilização de lógica do projeto. O editor do Signal Tap Logic Analyzer permite configurar os parâmetros de diversas formas, mudando a quantidade e os tipos de *triggers*, quantidade de amostras obtidas e quais *probes* são colocados.

Figura 36 – Signaltap Logic Analyzer



Fonte: do autor

Um arquivo Signaltap File (.stp) deve ser criado e configurado no Signaltap Logic Analyzer Editor e depois ter o projeto recompilado. Essa nova versão deve ser gravada na placa. Para observar as formas de onda em tempo real uma USB *Blaster* deve ser conectada na placa e no computador de forma a obter as aquisições.

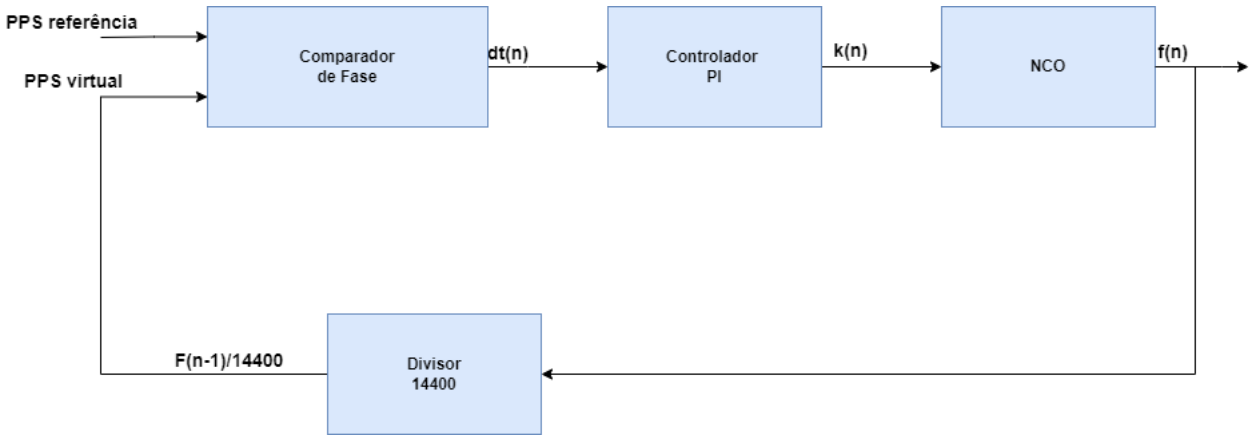
4.4.4 Omicron SVScout

O *software* SVScout da Omicron é uma ferramenta utilizada para a visualização de *Sampled Values* de acordo com a norma IEC 61860. O *software* recebe as *streams* de SVs de uma ou múltiplas *Merging Units* e exibe as formas de onda das tensões e correntes primárias de uma maneira similar a um osciloscópio (OMICRON, 2022b). O *software* conta com diversas funcionalidades, como visualizador de fasores, relatórios que resumem a informação de uma medida, gravação da captura dos *Sampled Values* e outros.

4.5 TESTES INICIAIS COM SIMPLES PPS

Primeiramente foram feitos testes iniciais para testar uma primeira versão do disciplinador de *clock* recebendo como entrada um pulso 1-PPS conforme mostrado na Figura 37 abaixo.

Figura 37 – Circuito digital inicial

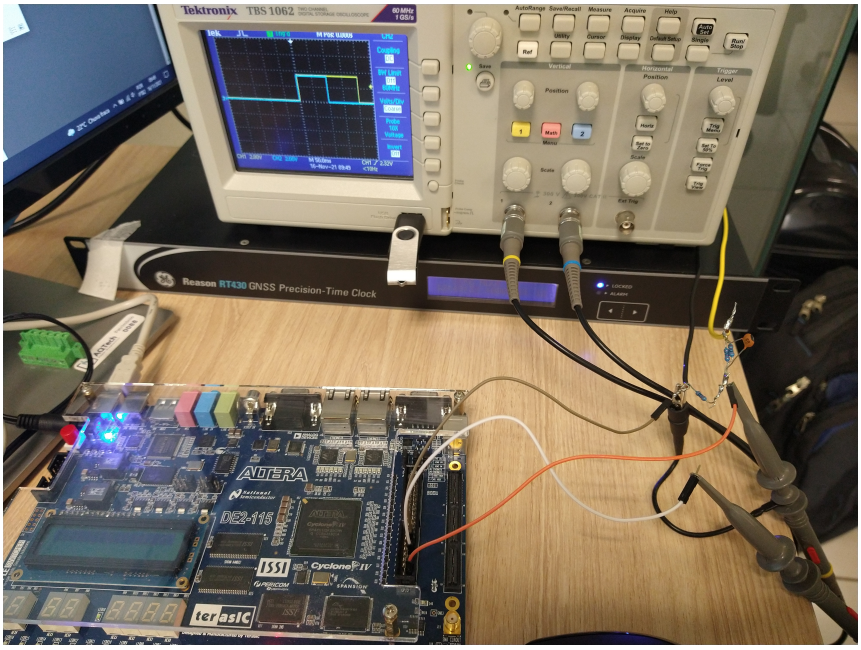


Fonte: do autor

O sistema recebe um sinal 1-PPS gerado pelo RT430 de 200 ms de largura como referência. Como a saída do GPS é uma saída elétrica TTL 5V CC, foi necessário um divisor resistivo para abaixar a tensão para 3,3V CC, que é a tensão aceita pelas GPIOS da placa DE2-115. O sinal de PPS virtual foi designado a um dos pinos do GPIO da placa DE2-115 de forma a comparar a diferença de fase e frequência entre o PPS virtual e a referência. A configuração para testes pode ser vista na Figura 38.

Apesar de ambos os sinais convergirem com uma média de 8 a 10 minutos após o ligamento da placa. O *jitter* médio observado foi de aproximadamente 10 μs, ainda considerado alto visando a aplicação. Foi possível presenciar algumas variações de fase inseperadas que aconteciam intermitentemente, a suspeita seria algum ruído que o sistema entendia como borda de subida do sinal. O mesmo comportamento era visto

Figura 38 – Configuração para testes



Fonte: do autor

quando acontecia a retirada e volta do sinal. Em certas ocasiões o PPS virtual ficava inconsistente e se distanciava do PPS referência não conseguindo se recuperar.

Após adicionado a condição no controlador na qual o contador que barra a ação da parte integral é reiniciado a cada poucos segundos pode se observar que o *jitter* melhorou para mais ou menos 500 ns. Isso se deve ao fato de que algumas vezes ocorria dois pulsos em que o erro constante estava em estado lógico alto, dessa forma o disciplinador não se ajustava corretamente.

4.6 FREQUENCÍMETRO E MÓDULOS PARA AUMENTAR ROBUSTEZ

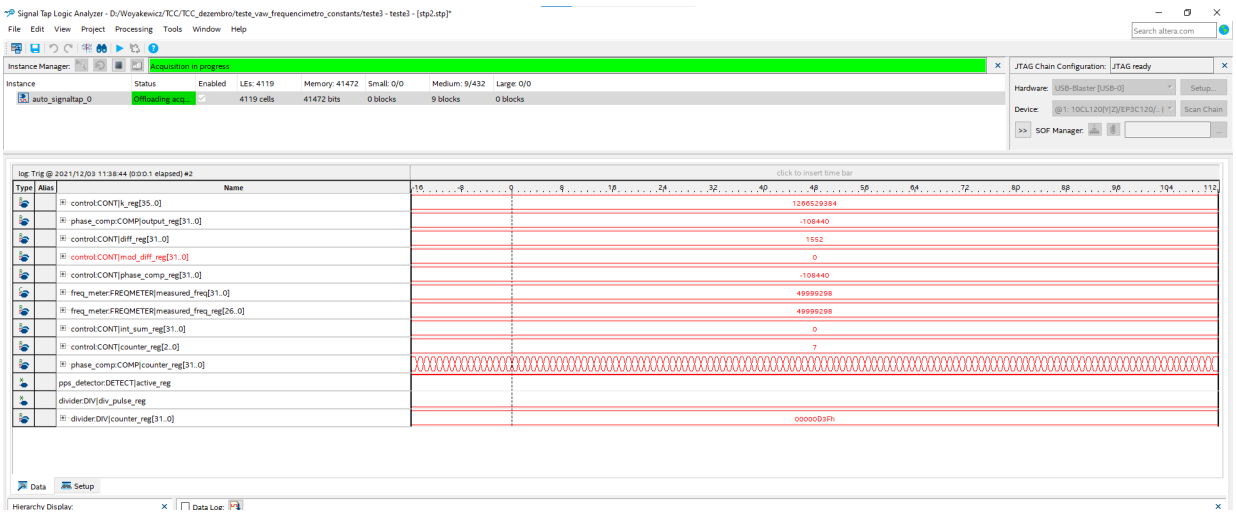
Em seguida foram feitos teste com módulos com finalidade de melhorar a robustez e precisão do sistema, evitando inconsistências e permitir que o sistema se recupere de uma perda de sinal.

4.6.1 Frequencímetro

O módulo foi adicionado ao sistema de forma a contar os ciclos de *clock* em um ciclo de sincronismo entregue pelo GNSS. Para verificar a quantidade de ciclos de *clock* foi utilizado a ferramenta signaltap logic analyzer como mostrado na figura abaixo.

Nos nódulos measured_freq e measured_freq_reg é possível se observar que a frequência não é exatamente 50 Mhz e sua precisão pode variar de acordo com

Figura 39 – Signaltap com frequencímetro



Fonte: do autor

o oscilador e as variações de temperatura. Um teste simples que foi feito é variar a temperatura do oscilador e observar a frequência medida alterar conforme essa variação.

Esse bloco tem como objetivo principal entregar ao valor do comparador de fase o valor exato da frequência *clock*, para melhor se ajustar. Nos testes realizados não pode se observar um ganho visível com a inserção do frequencímetro, contudo ele foi mantido para aumentar a robustez do sistema.

4.6.2 Limitador de K

Em seguida foram testados módulos que evitassem os saltos vistos devido a perda de sinal, ruído etc. Inicialmente foi feito um bloco simples que tem o propósito de limitar o valor de K gerado pelo controlador e que seria entregue ao NCO. A implementação é simples, sendo o K de saída o valor de K de entrada se ele estiver entre os limites superiores e inferiores de K estabelecidos por constantes. Caso contrário o sinal de saída será o próprio valor do limite.

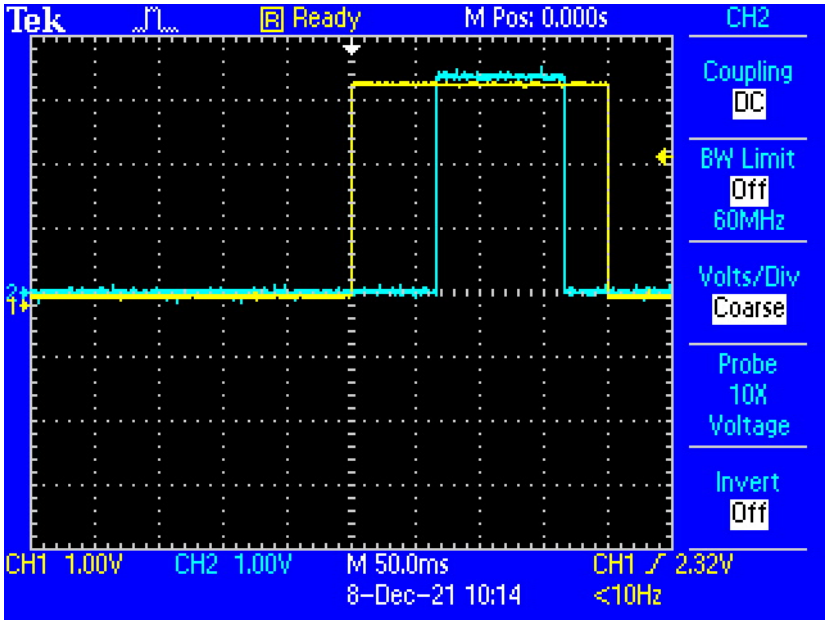
Figura 40 – Limitador de K

```
k_out <= K_SIG_UP when (k_in > K_SIG_UP) else
         K_SIG_LOW when (k_in < K_SIG_LOW) else
         k_in;
```

Fonte: do autor

Era esperado que esse bloco pudesse bloquear valores fora do comum, contudo não teve muito sucesso, sempre travando o sistema, como pode se ver pela figura 41 retirada do osciloscópio utilizado.

Figura 41 – Resultado Limitador de K



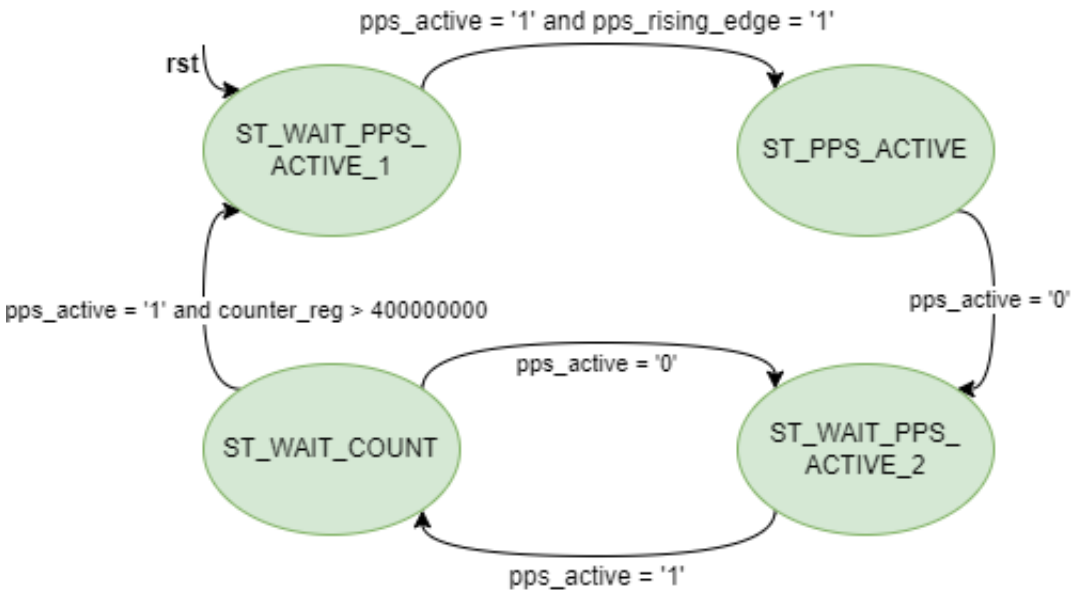
Fonte: do autor

4.6.3 K locked

Em seguida o módulo limitador de K foi substituído pelo módulo K Locked. Ele tem por objetivo ignorar qualquer valor de K gerado pelo controlador quando o pps active, que é o sinal que indica se tem um valor de PPS ativo, estiver em nível lógico baixo. Desta forma ele vai guardar o último valor de K válido para entregar para o NCO, esperando alguns segundos o controle se recuperar. O módulo é regido pela máquina de estados que se encontra na Figura 42.

A máquina de estados se inicia em ST_WAIT_PPS_ACTVE, a qual espera simultaneamente que tenha um sinal de PPS ativo e uma borda de subida de PPS para ir para ST_PPS_ACTIVE. Esse estado é o normal, no qual o K de entrada é o K de saída e continuará nesse estado até o sinal PPS ativo entre em nível lógico baixo, sinalizando uma perda de sinal, indo para o estado ST_WAIT_PPS_ACTIVE_2. Fora desse estado o valor do K de saída será sempre o valor do K do ciclo de clock anterior, ou seja, sempre o mesmo valor registrado da última vez que o PPS estava ativo. O estado ST_WAIT_PPS_ACTIVE_2 irá aguardar até que o sinal de PPS ativo retorne, indo para ST_WAIT_COUNT. Nesse estado, um contador irá ativar e esperar 8 segundos, 400000000 ciclos de *clock*, de PPS ativo para ir ST_WAIT_PPS_ACTIVE_1

Figura 42 – Máquina de estados do K Locked



Fonte: do autor

e seguir com o funcionamento normal. Caso o sinal de PPS ativo ir para nível lógico baixo em algum momento enquanto estiver nesse estado, ou seja, o sinal de PPS não está adequado, ele irá retornar para ST_WAIT_PPS_ACTIVE_2 para identificar um novo sinal de PPS ativo válido.

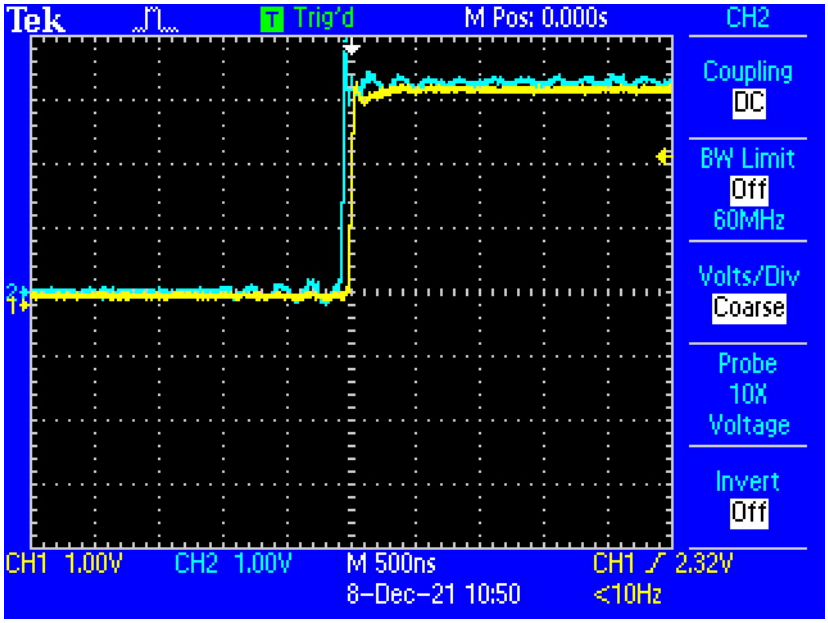
Esse módulo se mostrou efetivo no começo, contudo após algumas horas com o sistema ligado pode se observar um salto. Uma imagem da saída do osciloscópio se encontra na Figura 43

4.6.4 Phase locked

O módulo phase locked, diferentemente dos outros módulos é inserido entre o comparador de fase e o controlador, e irá manter o valor de erro de fase anterior quando o sinal de PPS não estiver ativo (quando ocorrer algum erro, ruído, etc) e irá esperar por 2 segundos de PPS ativo para garantir um sinal válido. Dessa forma garantimos que todo valor entregue ao controlador seja condizente e não vá deixar ele instável. Sua implementação pode ser encontrada no capítulo anterior. Diferentemente dos módulos limitadores de K e K locked esse módulo irá filtrar valores inconsistentes antes de chegar no controlador, evitando que ocorram variações bruscas no controlador.

Com esse bloco foi possível observar que as grandes variações de fase se extinguiram. Para validar isso foi criado um bloco chamado *error detection* que detectava quando o sinal saia de *locked*, ou seja quando perdesse o sincronismo. Quando isso acontece ele trava uma máquina de estados, acende um LED e registra diferente si-

Figura 43 – Resultado com K Locked

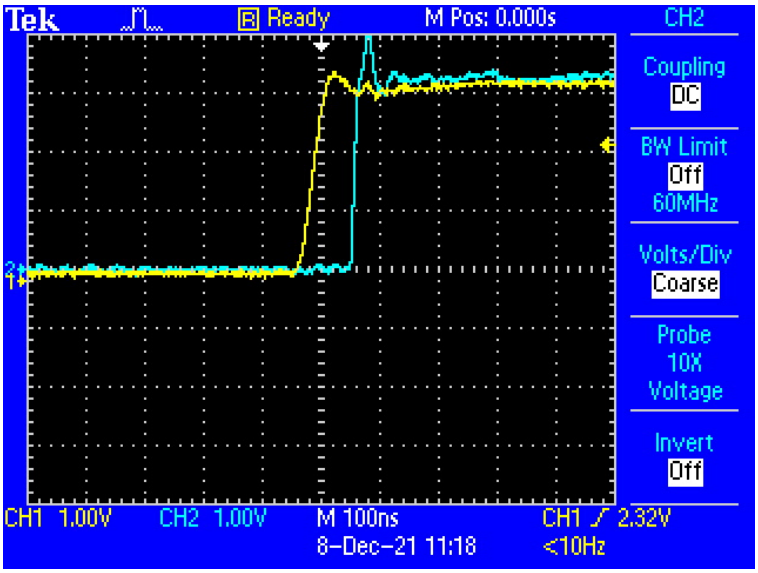


Fonte: do autor

nais de diferentes blocos. Foi feito um teste deixando os equipamentos ligados por um final de semana inteiro e não foi registrado nenhum erro, diferentemente do que se observava nas versões anteriores.

O módulo *error detection*, utilizado somente para testes, irá verificar se houve

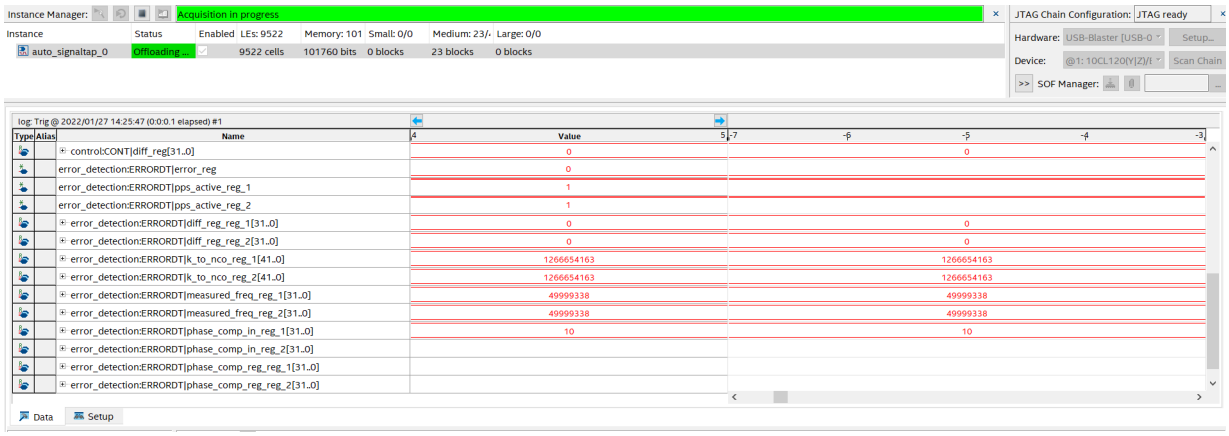
Figura 44 – Resultado com Phase Locked



Fonte: do autor

algum erro e é regido por uma máquina de estados de apenas dois estados e por isso não será mostrado um diagrama desses estados. Esse módulo é reiniciado por um botão no kit de desenvolvimento, começando no estado ST_WAITING_ERROR. Nesse estado, caso o sinal de locked do controle esteja em nível lógico baixo ele irá para o estado ST_ERROR_DETECTED, indicando que houve um erro, bem como deixando o sinal *error* em nível lógico alto. Como até o sistema entrar em regime permanente e entrar em *locked* a máquina de estados irá detectar erro, deve-se esperar que o sistema entre em *locked* e somente depois apertar o botão de *reset* do kit de desenvolvimento para começar a verificar se houve a falta de locked. Uma captura de tela do signaltap rodando o detector de erro se encontra na Figura 45.

Figura 45 – Captura de tela do Detector de erro no Signaltap



Fonte: do autor

Pode se observar pela figura diversos sinais que são analisados, mas o mais importante é o error_reg que é o registrador que irá sinalizar que houve um erro está em nível lógico baixo, indicando o correto funcionamento.

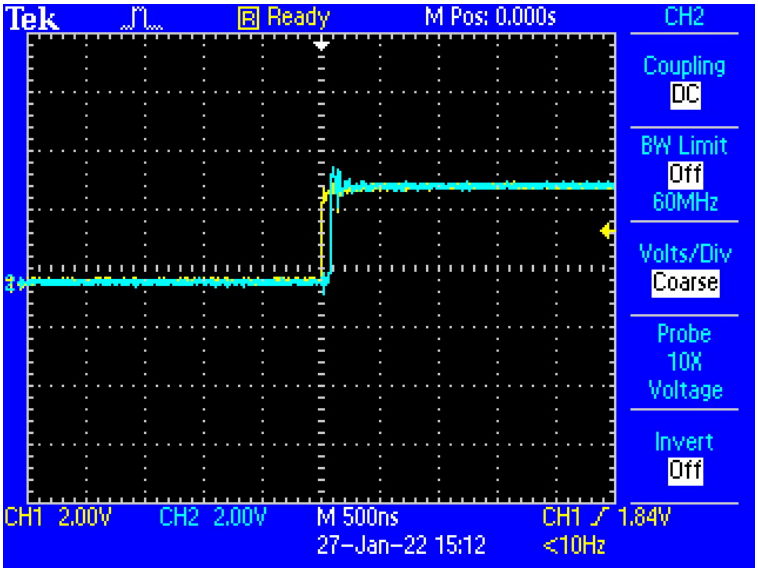
Outros testes foram feitos, removendo a entrada de sincronismo e retornando ela após alguns poucos minutos. Novamente não se observou nenhum salto e o sistema continuou comportado conforme esperado.

4.7 RESAMPLER E COMPENSADOR DE ATRASO DE REGISTRADORES

Foram feitos testes usando o modulo sincronizador, que contém a cadeia de registradores do sinal PPS de entrada e também foi modificado o módulo comparador de fase, de modo a compensar o atraso gerado pelos registradores do sincronizador. Seu resultado é apresentado na Figura 46.

Como esperado, não pode se observar uma diferença muito grande após a inserção dessas mudanças, já que é compensado apenas 40 ns e o jitter observado

Figura 46 – Resultado do Osciloscópio com a compensação de atraso



Fonte: do autor

está na casa dos 500 ns. Apesar disso essas alterações são importantes para robustez do sistema.

4.8 MELHORIAS NO CONTROLADOR

Em seguida foram feitos testes modificando os valores das constantes do controlador, de forma a analisar o comportamento do sistema e melhorar seu desempenho. Inicialmente foram testados alguns valores de $\frac{K_c}{\tau I}$, mantendo o valor de K_c em 1, observando a influência da parte integral sem alterar a parte proporcional do controlador. O resultado dos testes são apresentados na Tabela 2.

Tabela 2 – Resultado dos testes alterando a constante $\frac{K_c}{\tau I}$

$\frac{K_c}{\tau I}$	**tempo para regime permanente**	**Jitter**	**Observações**
1	8 a 10 minutos	250 ns +/-	normal
1/2	16 a 18 minutos	200 ns +/-	Melhorou um pouco o jitter, mas demora muito para se ajustar
1/4	+/- 25 minutos	200 ns +/-	Demorou muito para chegar em regime permanente e não teve nenhum ganho significativo
2	não convergiu	não convergiu	não convergiu

Fonte: do autor

Era esperado que quanto mais essa constante diminuísse, mais estável o con-

trole se tornasse, podendo diminuir o *jitter*. O *jitter* diminuiu levemente, contudo o aumento no tempo para o sistema entrar em regime permanente aumentou muito, portanto foi escolhido manter essa constante em 1.

Em seguida foram realizados os mesmos testes, agora modificando apenas K_c, mantendo o valor da constante $\frac{K_c}{\tau I}$ em 1, observando a influência da parte proporcional do controlador sem influenciar a parte integral. Os resultados dos testes se encontram na Tabela 3.

Tabela 3 – Resultado dos testes alterando a constante K_c

K_c	**tempo para regime permanente**	**Jitter**	**Observações**
1	8 a 10 minutos	250 ns +/-	normal
1/2	Não converge	Não converge	Demorou muito para chegar perto de convergir
2	7 a 9 minutos	200 ns +/-	Chegou a convergir um pouco mais rápido e teve um jitter um pouco menor
4	6 a 7 minutos	150 ns +/-	Teve o menor tempo de convergência e o menor jitter
8	6 a 7 minutos	100 ns +/-	Teve o mesmo tempo de convergência por causa que quem mais tem influencia no tempo de regime permanente é a parte integral
16	6 a 7 minutos	50 ns +/-	Melhor jitter de todos

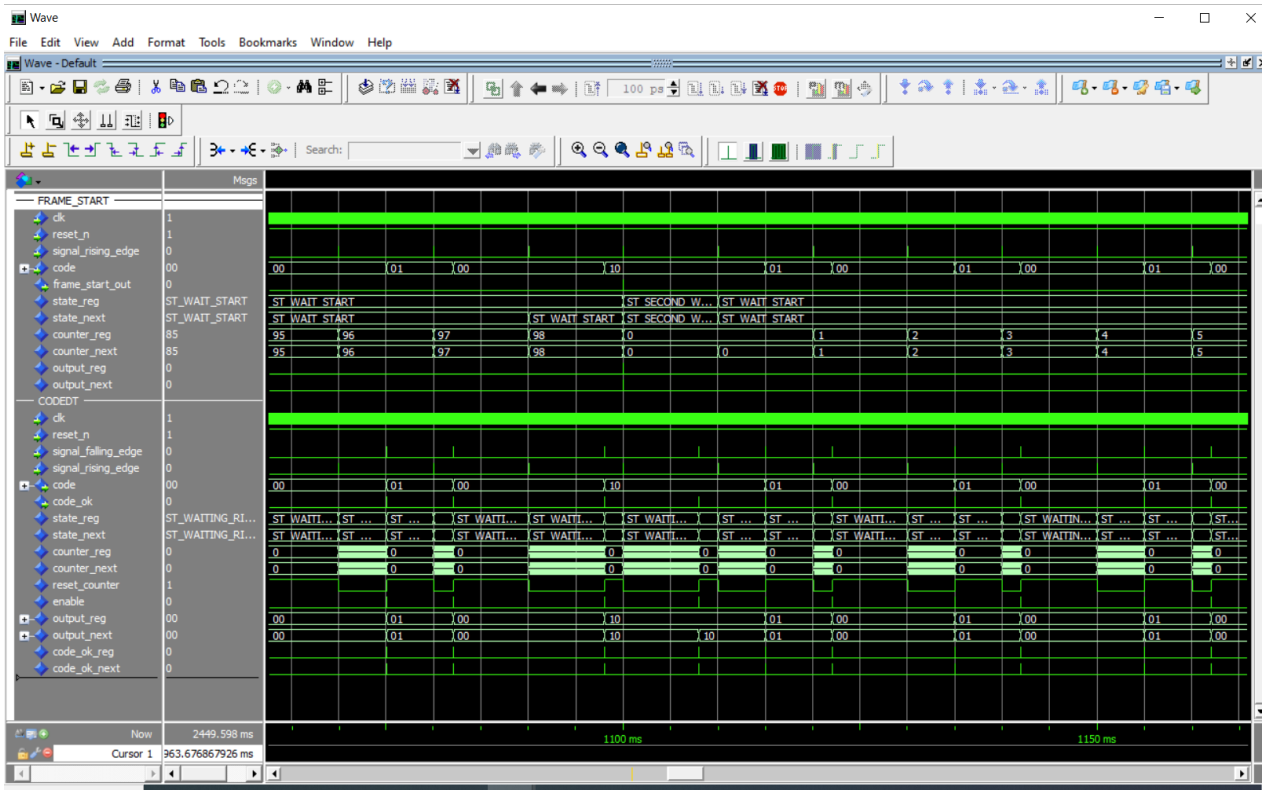
Fonte: do autor

Quanto maior fosse a constante K_c, mais rápido o sistema se ajustava até chegar em um regime permanente, sendo mais responsivo. O sistema também diminuiu o *jitter*, se ajustando mais rapidamente as variações de frequência do oscilador. Baseado nesses resultados foi escolhido utilizar um valor de K_c de 16. Adicionalmente foi feito um teste de remover o sinal PPS referência, cronometrando o tempo em que se mantinha no erro máximo de 1 μs. O sistema se manteve dentro desse erro por no mínimo 30 segundos obedecendo o tempo mínimo sem sinal de 5 segundos.

4.9 TESTES COM IRIG-B

Em seguida foram realizados testes para validar o sistema de aquisição IRIG-B. Dessa forma inicialmente foi validado o módulo pelo Modelsim, como pode se observar na Figura 47. Nessa imagem foram simulados os módulos Frame Start, Code Detector, Signal falling edge e Signal rising edge.

Figura 47 – Resultado do Modelsim dos módulos relacionados com IRIG-B



Fonte: do autor

Na Figura 47 Pode se observar as transições de códigos e uma geração de um sinal frame_start_out quando aparece um código “10”, após o contador chegar em 98, prevendo corretamente um sinal de PPS.

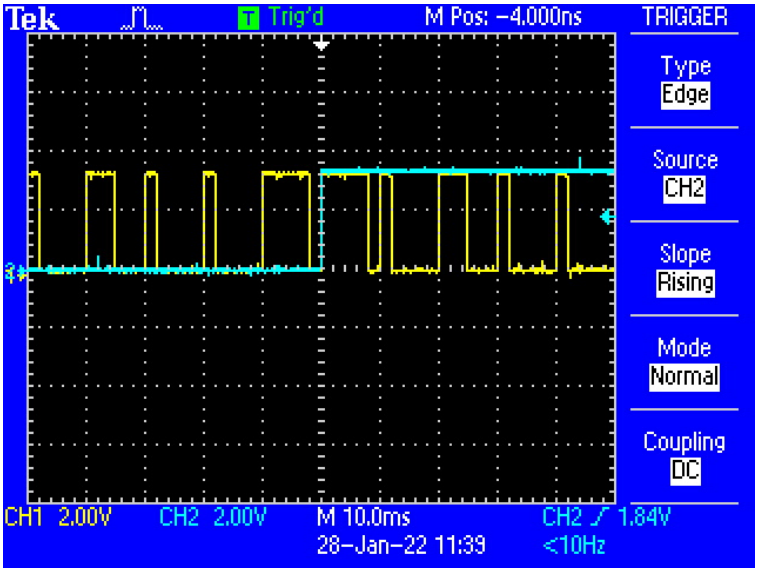
Em seguida foi externado o sinal de PPS extraído do pacote IRIG-B pelas GPIOs de forma a comparar com o sinal IRIG-B de entrada. A imagem retirada do osciloscópio se encontra na Figura 48.

Como pode se observar, a borda de subida do PPS extraído, em azul no canal 2, está exatamente na borda de subida do segundo código P de referência, ou seja, bem onde se encontra o começo do *frame*, conforme era esperado.

Para os próximos testes será usado como referência o sinal PPS extraido no canal 1, já no canal 2 estará o PPS virtual. Conforme esperado, o sistema funcionando com IRIG-B teve o mesmo comportamento do que quando utilizado PPS. Resultado se encontra na Figura 49.

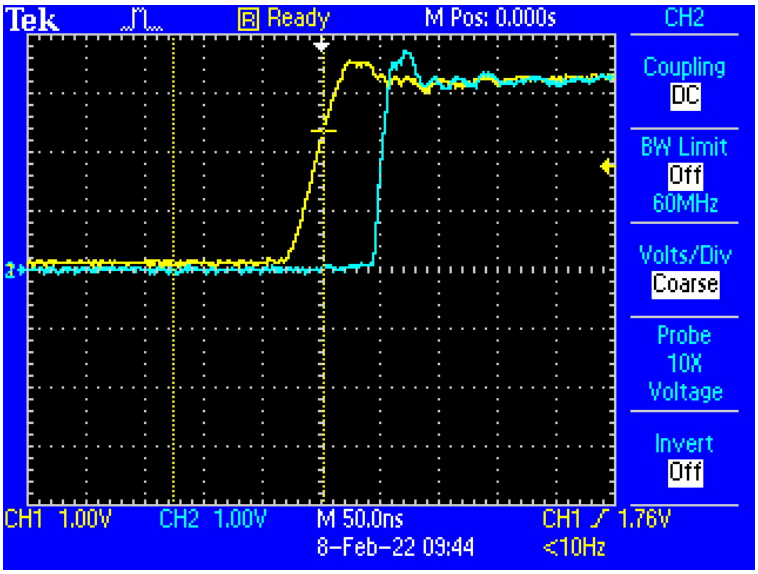
Por fim, foi externado a saída de 14,4 kHz dos NCOs para as GPIOs da placa DE2-115 conforme a figura 50. No canal 2 pode se observar que apesar da medida do osciloscópio ser inconclusiva ela está em aproximadamente 14,4 kHz, que é a

Figura 48 – Comparação do sinal IRIG-B e PPS



Fonte: do autor

Figura 49 – Resultado com sinal IRIG-B

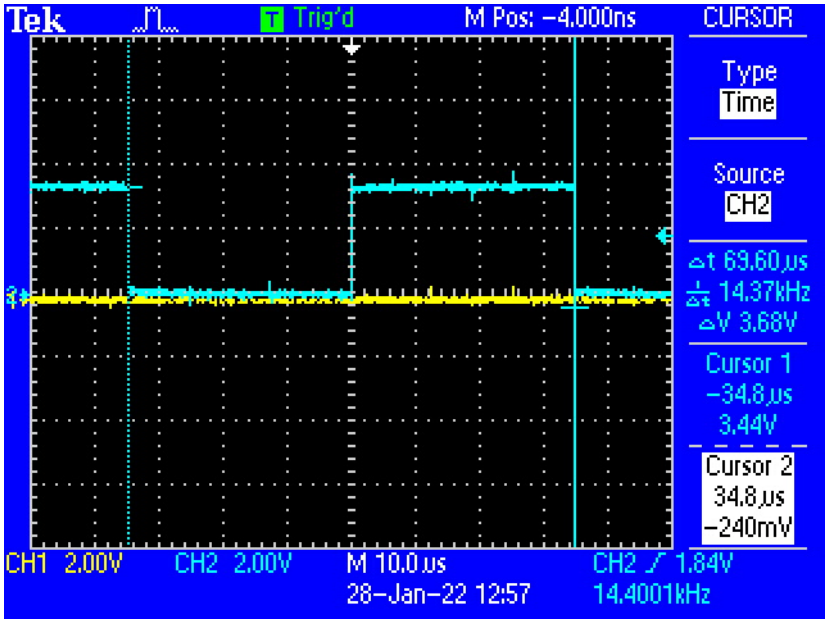


Fonte: do autor

frequência desejada.

Foi realizado um teste de leitura do frame IRIG-B criado pelo seu modulo em FPGA, sendo ele lido por um processador, utilizando um SoC, ou seja, um chip que continha um FPGA e um HPS no mesmo encapsulamento. Para uma aplicação específica a informação dos 200 bits do frame IRIG-B foi salva em um espaço de memória em palavras de 32 bits por meio de um DMA. Essas palavras foram então lidas pelo

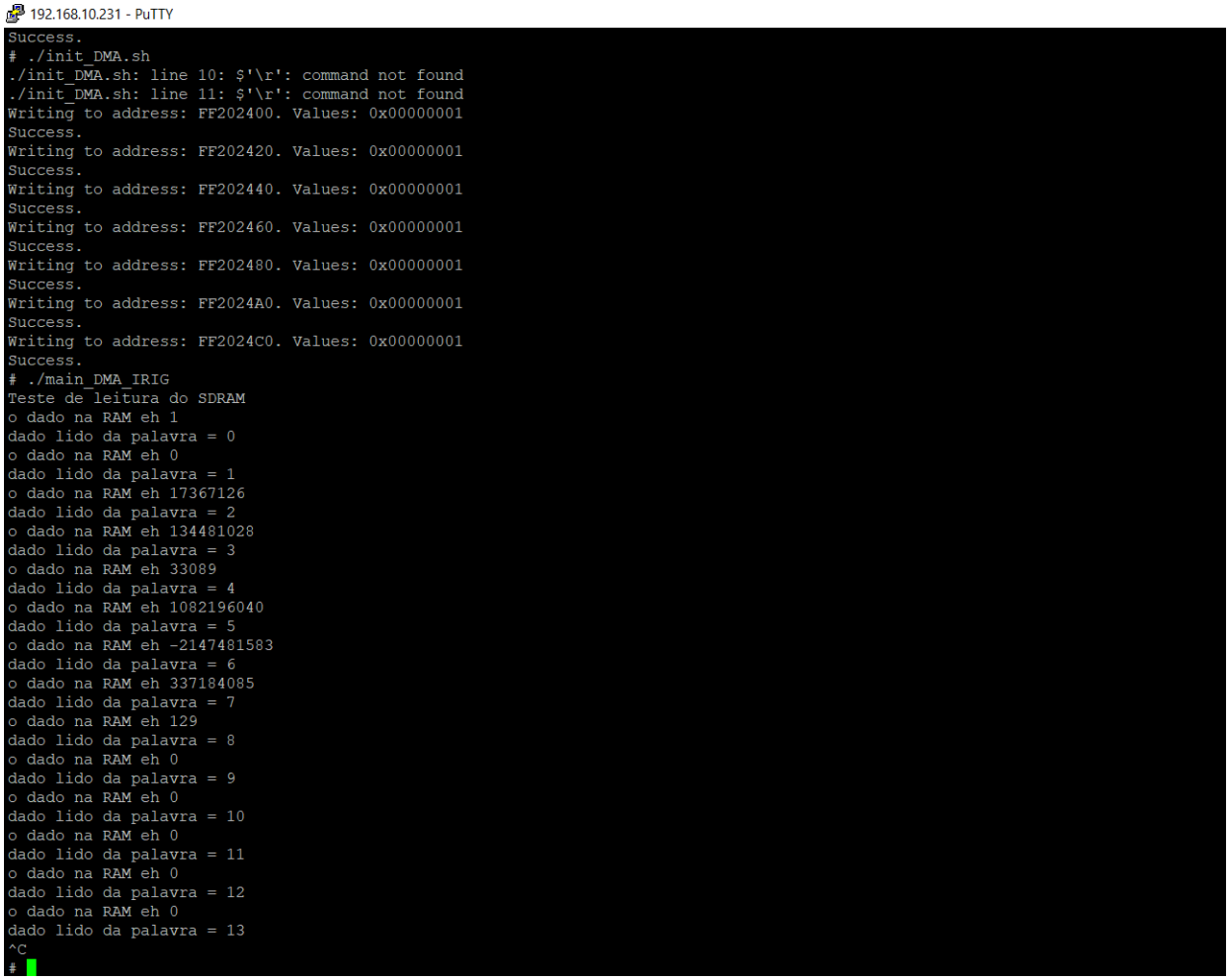
Figura 50 – Saída do NCO



Fonte: do autor

processador e apresentadas em um terminal serial.

Figura 51 – Teste de leitura pelo processador



Fonte: do autor

Na Figura 51 podemos observar cada palavra de 32 bits sendo escrita no espaço de memória. A palavra 0 contém apenas um flag de escrita “1”, já nas palavras 1 a 8 está contidas as informações do frame IRIG-B.

4.10 COMPARATIVO COM CMEMS

Em seguida foram realizados testes em uma placa na qual se utilizava um oscilador de tecnologia CMEMS de mesma frequência, de forma a comparar com o oscilador de cristal contido na placa DE2-115. Sistemas Osciladores Eletromecânicos CMOS (CMEMS) são osciladores extremamente estáveis, mais resistentes a variações de temperatura e mais confiáveis. Mais sobre osciladores CMEMS em (QUEVY, 2013). O objetivo desse teste é avaliar qual seria o ganho de um sistema utilizando osciladores mais precisos. Na Figura 52 temos a saída do osciloscópio.

Figura 52 – Teste utilizando um oscilador CMEMS

Fonte: do autor

No canal 1, em amarelo, temos o sinal referência, que está com sua polaridade invertida pela utilização de um conversor óptico-elétrico, sendo assim sua borda de subida de referência na verdade é a borda de descida. Já no canal 2 temos o PPS virtual criado pelo sistema.

Utilizando a versão final do projeto observou-se que o sistema demorava aproximadamente 2 minutos para entrar em regime permanente com um *jitter* de +/- 25 ns. Comparado com um oscilador de quartzo tradicional, que demorou aproximadamente 6 a 7 minutos para entrar em regime permanente com um *jitter* de +/- 50 ns, o sistema

teve um ganho significativo. Pode se observar, portanto, que o sistema pode obter resultados ainda melhores quando utilizado osciladores mais estáveis e precisos.

4.11 TESTES NA MERGING UNIT

Em seguida foi feito a inserção desse sistema em uma Merging Unit de forma a gerar a frequência sincronizada de 14,4 kHz para aquisição de amostras por um conversor analógico-digital. Essa etapa é de extrema importância para validar o conceito do projeto, mostrando sua utilização em um produto real.

Para esse teste foi utilizado o conjunto de testes CMC 353 da Omicron para gerar uma corrente eficaz de 20 amperes na frequência de 60 Hz por um condutor, esse condutor é então enrolado 20 vezes pela bobina que irá fazer a aquisição, sendo assim a corrente medida será de 400 amperes eficaz. A configuração do conjunto de testes gerando corrente se encontra na Figura 53.

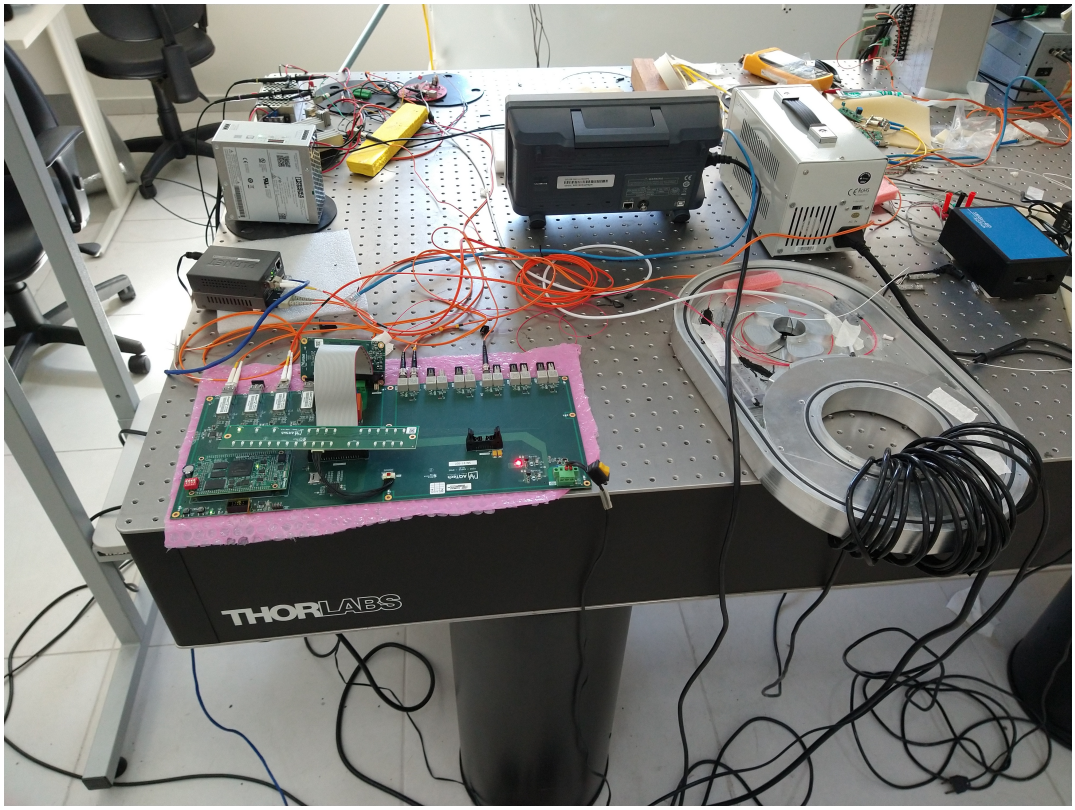
Figura 53 – Conjunto de testes CMC 353

Fonte: do autor

A aplicação do sistema para a geração de 14,4 kHz em FPGA foi então implementada juntamente com a implementação em FPGA que realiza a aquisição das amostras de corrente. Dessa forma é realizada a aquisição das amostras em sincronia com um sinal de referência gerado pelo receptor GNSS RT 430. Como o equipamento

é para aplicação em proteção e medições em geral, a taxa de saída de dados digital é de 4800 Hz. Dessa forma após a aquisição, as amostras passam por um filtro decimador. A configuração da placa utilizada para testes se encontra na Figura 54.

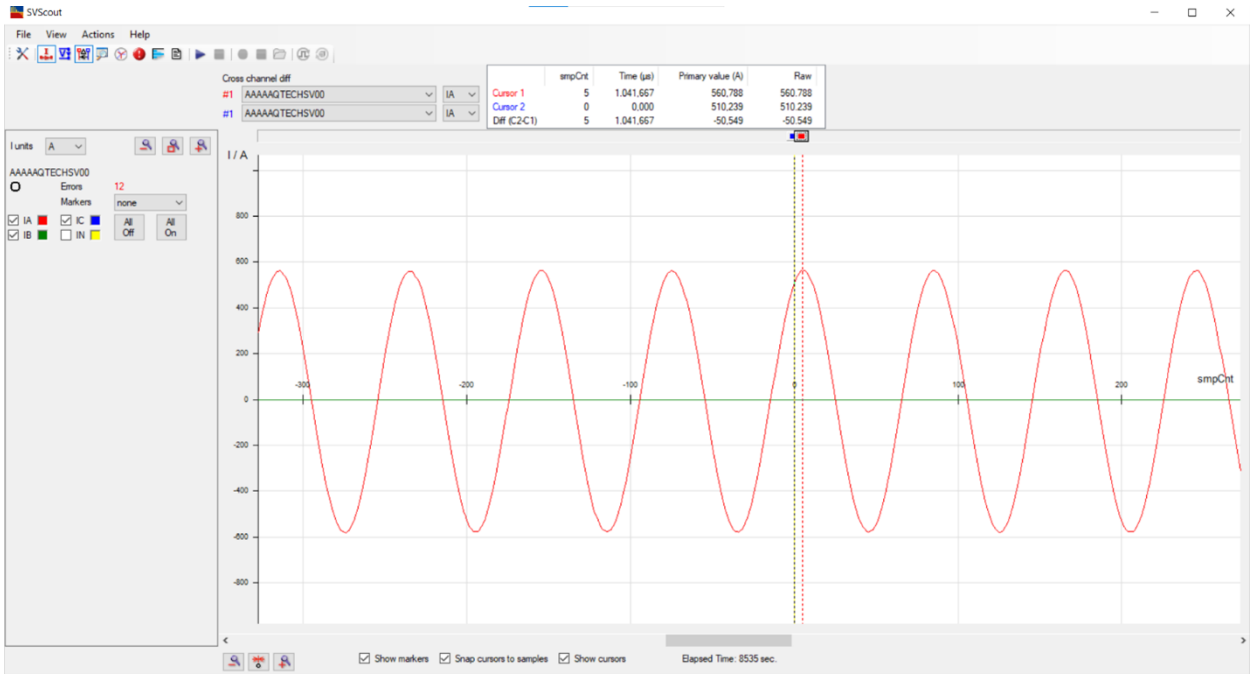
Figura 54 – Placa da Merging Unit para testes



Fonte: do autor

Após realizado as aquisições, são gerados Sampled Values a uma taxa de 4800 amostras por segundo, que são enviados pela rede e então recebidas por um notebook. Para a validação do conteúdo dos frames ethernet contendo Sampled Values foi utilizado o *software* visualizador IEC 61850 SVScout da Omicron. A Figura 55 apresenta as amostras pelo *software* SVScout.

Figura 55 – Visualização das amostras pelo SVScout



Fonte: do autor

Na Figura 55 podemos observar uma corrente eficaz de 400 amperes devidamente amostrada, com cada ciclo contendo 80 amostras. Dessa forma pode-se constatar que o resultado foi um sucesso, cumprindo com a proposta do projeto.

4.12 RECURSOS UTILIZADOS

A IDE Intel Quartus Prime possui o recurso de relatório de compilação, no qual é fornecida diversas informações de utilização de recursos do projeto. Para cada dispositivo é relatado a utilização de elementos lógicos, frequência máxima entre outros. Em seguida são apresentadas as principais informações do relatório gerado para o dispositivo EP4CE115F29C7 da família Cyclone IV E da Altera na Tabela 4.

Tabela 4 – Relatório de compilação

Descrição	**Utilizado**	**Total**	**Percentual**
Elementos lógicos totais	1137	114480	<1%
Total de registradores	611	-	-
Total de pinos	25	529	5%
Total de bits de memória	0	3981312	0%
Elementos multiplicadores de 9 bits embarcados	0	532	0%
Total de PLLs	0	4	0%
Frequência máxima	101,51 MHz	-	-

Fonte: do autor

5 CONCLUSÃO

Neste trabalho de conclusão de curso foi feito inicialmente uma breve introdução a subestações digitais, seus protocolos e equipamentos utilizados, bem como diversos conceitos e normas utilizados durante o trabalho. Em seguida foi proposto e implementado em FPGA um circuito digital, utilizando linguagem VHDL, capaz de gerar frequências sincronizadas com um pulso de referência.

Diversos testes executados durante a realização do trabalho foram apresentados, bem como seus resultados e dificuldades encontradas durante a prototipação. Dessa forma foi possível validar o cumprimento dos objetivos propostos nesse trabalho. Nesses testes foram comprovados que o sistema atende os requisitos de tempo previstos pela norma, bem como sua robustez a perda de sinais e comportamento no retorno desse sinal. O circuito também é capaz de receber duas das interfaces mais comumente utilizadas para comunicação de sincronismo, PPS e IRIG-B. O projeto atende também os requisitos estipulados para o circuito digital, comprovados por meio do relatório de compilação da IDE Intel Quartus Prime.

A eficácia do sistema foi comparada com dois osciladores de tecnologias diferentes, na qual pode se observar que osciladores mais estáveis e precisos fornecem resultados melhores. Foi apresentado um caso de uso do sistema em um protótipo de um equipamento, comprovando portanto a utilidade, eficácia e correto funcionamento do sistema em um equipamento real.

5.1 TRABALHOS FUTUROS

Uma proposta para trabalho futuro é a inserção de um módulo para extração da informação de sincronismo por meio do protocolo PTP, o qual é extremamente preciso. Desta forma o sistema ficará apto a receber os protocolos mais comumente usados nas subestações digitais.

Ainda é necessário realizar mais testes rigorosos com equipamentos, comparando sua oscilografia e principalmente verificando se ocorrem erros de fase e frequência das medidas em um cenário real. Verificando a taxa de distorção harmônica e verificando se ocorrem defasagens devido a algum erro do sistema. Ainda mais uma proposta seria avaliar o sistema para a geração de outras frequências além da de 14,4 kHz.

REFERÊNCIAS

ALMEIDA, Ezequiel Mendes de. **NORMA IEC 61850 – NOVO PADRÃO EM AUTOMAÇÃO DE SUBESTAÇÕES**. 2011. F. 58. Monografia (Graduação) – Universidade Federal do Ceará, Fortaleza.

ALTERA. **Undertanding Metastability in FPGAs**. [*S.l.*: *s.n.*]. Disponível em: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/wp/wp-01082-quartus-ii-metastability.pdf. Acesso em: 19 jan. 2022.

CHU, Pong P. **RTL Hardware Design Using VHDL: Coding for Efficiency, Portability and Scalability**. 1. ed. New Jersey: Wiley-Interscience, 2006.

GARG, Sachin. **Digital Substation Market with Covid-19 Impact Analysis by Module (Hardware, Fiber-Optic Communication Networks, and Scada Systems), Type (Transmission and Distribution), Voltage, Installation Type, Industry, and Region - Global Forecast to 2025**. [*S.l.*]. Disponível em: https://www.marketsandmarkets.com/Market-Reports/digital-substation-market-43227003.html. Acesso em: 10 jan. 2022.

GASPARINI, Leonardo; ZADEDYURINA, O.; FONTANA, Giorgio; MACII, D.; BONI, A.; OFEK, Y. A Digital Circuit for Jitter Reduction of GPS-disciplined 1-pps Synchronization Signals. *In*: p. 84–88.

GE GRID SOLUTIONS. **Manual Técnico Reason RT430/RT434**. [*S.l.*: *s.n.*]. Disponível em: https://www.gegridsolutions.com/app/DownloadFile.aspx?prod=RT430&type=3&file=25. Acesso em: 20 nov. 2021.

GISSELQUIST, Dan. **Building a Numerically Controlled Oscillator**. [*S.l.*: *s.n.*], 2017. Disponível em: https://zipcpu.com/dsp/2017/12/09/nco.html. Acesso em: 15 nov. 2021.

INTEL. **About the SignalTap II Logic Analyzer**. [*S.l.*: *s.n.*]. Disponível em: https://www.intel.com/content/www/us/en/programmable/quartushelp/13.0/mergedProjects/program/ela/ela_view_using.htm. Acesso em: 21 nov. 2021.

INTEL. **Intel Quartus Prime User Guide**. [*S.l.*: *s.n.*]. Disponível em: https://www.intel.com/content/www/us/en/programmable/products/design-

software/fpga-design/quartus-prime/user-guides.html. Acesso em: 21 nov. 2021.

INTERNATIONAL ELECTROTECHNICAL COMISSION. **IEC 61850-** Communication networks and systems for power utility automation – Part 5: Communication requirements for functions and device models. [*S.l.*], 2013.

INTERNATIONAL ELECTROTECHNICAL COMISSION. **IEC 61869-9**: Instrument transformers - Part 9: Digital interface for instrument transformers. [*S.l.*], 2019.

LATTICE. **Numerically Controlled Oscilator**. [*S.l.*: *s.n.*]. Disponível em: https://www.latticesemi.com/Products/DesignSoftwareAndIP/IntellectualProperty/IPCore/IPCores02/NumericallyControlledOscillator.aspx. Acesso em: 15 nov. 2021.

MACHADO MONTEIRO, Carlos Augusto; SOUZA, Celso Luis de; DALMAS, Marcelo. Global sample synchronization trigger. *In*: 2016 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS). [*S.l.*: *s.n.*], 2016. P. 1–4.

MENTOR GRAPHICS CORPORATION. **Modelsim User's Maual**. [*S.l.*: *s.n.*]. Disponível em: https://www.microsemi.com/document-portal/doc_view/131619-modelsim-user. Acesso em: 21 nov. 2021.

MILLS, D. L. **How NTP Works**. [*S.l.*: *s.n.*]. Disponível em: https://www.eecis.udel.edu/~mills/ntp/html/warp.html. Acesso em: 10 nov. 2021.

MORETO, Miguel; ROLIM, J. G. Análise automática de oscilografias em sistemas elétricos de potência. *In*.

NI. **FPGA Fundamentals**. [*S.l.*: *s.n.*]. Disponível em: https://www.ni.com/pt-br/innovations/white-papers/08/fpga-fundamentals.html. Acesso em: 15 nov. 2021.

OMICRON. **CMC 353 Brochure**. [*S.l.*: *s.n.*]. Disponível em: https://www.omicronenergy.com/download/document/4013D9DD-5F80-4BBE-AC8C-F8E0921B1EED/. Acesso em: 14 fev. 2022.

OMICRON. **SVScout Brochure**. [*S.l.*: *s.n.*]. Disponível em: https://www.omicronenergy.com/download/document/26A4CF2F-ED11-464F-BBCC-ABE14AFEF32C/. Acesso em: 14 fev. 2022.

ONS. **Glossário**. [*S.l.*]. Disponível em: http://www.ons.org.br/paginas/conhecimento/glossario. Acesso em: 10 nov. 2021.

PEDRONI, Volnei A. **Circuit Design with Vhdl**. 3. ed. Cambridge: The MIT Press, 2020.

QUEVY, Emmanuel. **CMEMS Technology: Leveraging High-Volume CMOS Manufacturing for MEMS-Based Frequency Control**. [*S.l.*], 2013.

RAJU, Febin; ALEX, Surya Susan. Fault direction detection and fault location identification in transmission lines using traveling waves. *In*: 2015 Online International Conference on Green Engineering and Technologies (IC-GET). [*S.l.*: *s.n.*], 2015. P. 1–6.

RANGE COMMANDERS COUNCIL, U.S. ARMY WHITE SANDS MISSILE RANGE. **IRIG STANDARD 200-04: "IRIG Serial Time Code Formats."** New Mexico: [*s.n.*], 2004.

TERASIC TECHNOLOGIES INC. **DE2-115 User Manual**. [*S.l.*: *s.n.*], 2012. Disponível em: https://www.intel.com/content/dam/altera-www/global/en_US/portal/dsn/42/doc-us-dsnbk-42-1404062209-de2-115-user-manual.pdf. Acesso em: 15 nov. 2021.

WALEED, Aashir; VIRK, Umar Siddique; RIAZ, Muhammad Tanveer; MEHMOOD, Shaikh; AHMAD, Saeed; JAVED, Muhammad; RAZA, Ali. Effectiveness and Comparison of Digital Substations Over Conventional Substations. **Advances in Science, Technology and Engineering Systems Journal**, v. 4, p. 431–439, ago. 2019.

XILINX. **What is an FPGA**. [*S.l.*: *s.n.*]. Disponível em: https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html. Acesso em: 15 nov. 2021.