UNIVERSIDADE FEDERAL DE JUIZ DE FORA
FACULDADE DE ENGENHARIA ELÉTRICA
GRADUAÇÃO EM ENGENHARIA ELÉTRICA - HABILITAÇÃO EM SISTEMAS ELETRÔNICOS

Igor Daher Cabral

Implementação em FPGA de Método de Detecção de Novidades Adaptativo baseado em DTW para Sinais de Qualidade de Energia Utilizando Processadores *Soft-Core*

Juiz de Fora
2025

Igor Daher Cabral

Implementação em FPGA de Método de Detecção de Novidades Adaptativo baseado em DTW para Sinais de Qualidade de Energia Utilizando Processadores *Soft-Core*

Trabalho de conclusão de curso apresentada ao Faculdade de Engenharia Elétrica da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do grau de bacharel em Engenharia Elétrica. Área de concentração: Sistemas Eletrônicos

Orientador: Prof. Dr. Eng. Eder Barboza Kapisch

Juiz de Fora

2025

Ficha catalográfica elaborada através do Modelo Latex do CDC da UFJF
com os dados fornecidos pelo(a) autor(a)

Daher Cabral, Igor.
Implementação em FPGA de Método de Detecção de Novidades Adaptativo baseado em DTW para Sinais de Qualidade de Energia Utilizando Processadores *Soft-Core* / Igor Daher Cabral. – 2025.
66 f. : il.

Orientador: Eder Barboza Kapisch
Trabalho de Conclusão de Curso (Graduação) – Universidade Federal de Juiz de Fora, Faculdade de Engenharia Elétrica. Graduação em Engenharia Elétrica - Habilitação em Sistemas Eletrônicos, 2025.

1. Palavra-chave. 2. Palavra-chave. 3. Palavra-chave. I. Sobrenome, Nome do orientador, orient. II. Título.

Igor Daher Cabral

Implementação em FPGA de Método de Detecção de Novidades Adaptativo baseado em DTW para Sinais de Qualidade de Energia Utilizando Processadores *Soft-Core*

Trabalho de conclusão de curso apresentada ao Faculdade de Engenharia Elétrica da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do grau de bacharel em Engenharia Elétrica. Área de concentração: Sistemas Eletrônicos

Aprovada em 31 de julho de 2025

BANCA EXAMINADORA

Prof. Dr. Eng. Eder Barboza Kapisch - Orientador
Universidade Federal de Juiz de Fora

Prof. Dr. Eng. Leandro Rodrigues Manso Silva
Universidade Federal de Juiz de Fora (UFJF)

Prof. D. Sc. Luciano Manhães de Andrade Filho
Universidade Federal de Juiz de Fora (UFJF)

Dedico este trabalho à minha família e meu amor, pelo amor incondicional, apoio constante e por sempre acreditarem em mim, mesmo nos momentos mais difíceis.

Aos meus amigos, que estiveram ao meu lado ao longo dessa caminhada, oferecendo palavras de incentivo, companhia nas horas difíceis e celebrações nas conquistas.

A todos vocês, minha eterna gratidão.

AGRADECIMENTOS

Agradeço primeiramente à minha família e ao meu amor, pelo amor, paciência e por sempre acreditarem no meu potencial. Sem vocês, essa conquista não teria sido possível.

Aos meus amigos, que estiveram presentes em cada etapa dessa jornada, oferecendo apoio, conselhos e, acima de tudo, amizade verdadeira.

Aos colegas de faculdade, com quem compartilhei aprendizados, desafios e momentos marcantes ao longo do curso.

Aos professores, pelo conhecimento transmitido, pela dedicação em sala de aula e pelo incentivo à busca contínua por aprendizado.

Aos membros do NIPS (Núcleo de Instrumentação e Processamento de Sinais), pela oportunidade de participar de projetos desafiadores, pelas discussões enriquecedoras e pelo ambiente de constante evolução técnica e profissional.

Agradeço também ao órgão de fomento CNPQ pelo suporte financeiro concedido por meio de bolsa, que foi fundamental para a realização deste trabalho.

A todos que, de alguma forma, contribuíram para a construção deste percurso, o meu muito obrigado.

RESUMO

Este trabalho apresenta a implementação de um protótipo de um detector de novidades em uma plataforma *Field Gate Programmable Array* (FPGA) baseado na métrica de similaridade dinâmica *Dynamic Time Warp* (DTW). O método implementado se destaca por possuir capacidade de segmentação adaptativa e análise de *frames* de tamanhos distintos, evitando falsos alarmes de detecção, como em cenários de ruído, harmônicos de variações de frequência, sendo voltado para a aplicação em detecção de distúrbios de Qualidade de Energia Elétrica (QEE) presentes nos sinais de tensão e corrente do Sistema Elétrico de Potência (SEP). O trabalho é atual, pois distúrbios desconhecidos podem surgir no novo contexto de consolidação das Redes Inteligentes (RI). O protótipo funcional do detector foi desenvolvido utilizando processadores *Soft-Core* (SAPHO), para que fosse possível obter a otimização do consumo de hardware. Para os testes experimentais, foi utilizado um limiar de Detecção de Novidades (DN) obtido a partir de testes emíricos de simulação, considerando um nível de ruído de 35 dB para a *Signal-to-Noise Ratio* (SNR). Os testes de bancada confirmam que o projeto satisfaz os requisitos para operar em tempo real e se mostra eficiente em detectar novidades. Além disso, foi ocupada cerca de 4% da memória do FPGA, e menos de 3% dos outros recursos de hardware.

Palavras-chave: Sistema Elétrico de Potência; DTW; FPGA; Processador Soft-Core; Qualidade de Energia; Métrica de Similaridade Dinâmica.

ABSTRACT

This work presents the implementation of a novelty detector on a Field Programmable Gate Array (FPGA) platform based on the Dynamic Time Warping (DTW) similarity metric. The prototype stands out by avoiding false detection alarms in the presence of frequency variations. It also features adaptive segmentation capabilities and analysis of segments with different lengths, targeting the application of detecting Power Quality Disturbances (PQD) in voltage and current signals of the Electric Power System (EPS). The topic is current, as unknown disturbances may emerge in the new context of Smart Grid (SG) consolidation. The functional prototype was developed using Soft-Core processors to optimize hardware resource usage. For the experimental tests, the Novelty Detection (ND) threshold was determined through empirical simulations, considering a Signal-to-Noise Ratio (SNR) of 35 dB. Bench tests confirm that the design meets the requirements for real-time operation and proves effective in detecting novelties. Additionally, the implementation used approximately 4% of the FPGA memory and less than 3% of other available resources.

Keywords: Electric Power System; DTW; FPGA; Soft-Core Processor; Power Quality; Dynamic Similarity Metric.

LISTA DE ILUSTRAÇÕES

Figura 1: 1º e 2º passos para a construção da matriz D. 21
Figura 2: 3º passo para a construção da matriz D. Valor da DTW: 11. 22
Figura 3: Sinais utilizados. 22
Figura 4: Mapa de calor do custo DTW com o caminho ótimo. 23
Figura 5: Metodologia do Detector de Novidades. 25
Figura 6: Estrutura do Hardware. 26
Figura 6: Imagem da versão da IDE do SAPHO utilizada neste trabalho. 27
Figura 8: Atraso de grupo do filtro 28
Figura 9: Módulo da resposta em frequência do filtro Nutall utilizado. 29
Figura 10: Fase da resposta em frequência do filtro Nutall utilizado. 29
Figura 11: Resposta ao impulso do filtro Nutall utilizado. 29
Figura 12: Amostras no instante do cruzamento por zero: correção por interpolação linear. 31
Figura 13: Sinal seccionado sem atrasar de 20 amostras. 32
Figura 14: *Mega Wizard* da FIFO. 33
Figura 15: Histograma dos valores DTW 35
Figura 16: Consumo de lógica. 36
Figura 17: Parte principal do RTL Viewer do Projeto. 37
Figura 18: Teste em Matlab® de detecção com transitório. 39
Figura 19: Teste ModelSim® de detecção com transitório. 40
Figura 20: Teste em Matlab® de detecção com *sag*. 40
Figura 21: Teste no ModelSim® de detecção com *sag*. 41
Figura 22: Teste em Matlab® de detecção com harmônico variante no tempo. 41
Figura 23: Teste no ModelSim® de detecção com harmônico variante no tempo. . . 42
Figura 24: Teste em Matlab® com variação de frequência. 42
Figura 25: Teste no ModelSim® de detecção com variação linear de frequência. . . . 43
Figura 26: Teste no ModelSim® de tempo de processamento. 43
Figura 27: Bancada montada para captura de sinal no osciloscópio. 44
Figura 28: Sinal senoidal com ruído de fundo entrando no Signal Tap e sendo processado. 45
Figura 29: Signal Tap com os sinais senoidais começando a variar a frequência. . . . 46
Figura 30: Signal Tap com os sinais senoidais com grande variação na frequência. . 46
Figura 31: Signal Tap com os sinais senoidais com frequência acima do limiar onde não é detectada tal variação. 47
Figura 32: Teste com detecção de harmônico para medição de ciclos de *clock*. 47
Figura 33: Teste com frequência off-nominal para medição de ciclos de *clock*. 48
Figura 34: Relatório de compilação do Quartus mostrando a frequência máxima de operação para o chip selecionado. 49

Figura 35: Teste com detecção de transitório. 50
Figura 36: Teste de detecção com *sag*. 51
Figura 37: Teste de detecção com harmônico variante no tempo 52
Figura 38: Teste de detecção com variação de frequência. 52

LISTA DE TABELAS

Tabela 1 – Coeficientes do vetor $h[n]$ 30
Tabela 2 – Consumo de recursos por módulo (sem Signal Tap) 36

LISTA DE ABREVIATURAS E SIGLAS

DTW	Dynamic Time Warping
DFE	Diferença das Energias de Quadros
DN	Detecção de Novidade
ED	Euclidean Distance
EDR	Edit Distance on Real signal
FIFO	First In First Out
FIR	Finite Impulse Response
FPGA	Field Programmable Gate Array
IDE	Integrated Development Environment
MD	Manhattan Distance
MM	Morfologia Matemática
MU	Merging Units
NIPS	Núcleo de Instrumentação e Processamento de Sinais
PMU	Medição Fasorial
QEE	Qualidade de Energia Elétrica
RMS	Root Mean Square
RI	Redes Inteligentes
SAPHO	Scalable Architecture Processor for Hardware Optimization
SEP	Sistema Elétrico de Potência
SNR	Signal to Noise Ratio
SV	Sampled Values
TWED	Time Warp Edit Distance
ZC	Zero Crossing

SUMÁRIO

1 INTRODUÇÃO 13
1.1 Contextualização 13
1.2 Objetivos 15
1.3 Produção Bibliográfica 15
1.4 Estrutura do Trabalho 16
2 REVISÃO BIBLIOGRÁFICA 17
2.1 Detecção de Novidades em Sistemas Elétricos de Potência 17
2.2 *Dynamic Time Warping* (DTW) 19
2.2.1 Vantagens da DTW 19
2.2.2 Funcionamento do Algoritmo DTW 19
2.3 Segmentação de Sinais com Variação de Frequência 24
3 IMPLEMENTAÇÃO DO DETECTOR DE NOVIDADES . . 25
3.1 Metodologia do Detector de Novidades 25
3.2 Metodologia de Implementação do Detector de Novidades em FPGA . . 25
3.2.1 SAPHO 26
3.2.2 Processador Zero-Crossing (SAPHO 1) 28
3.2.2.1 Filtro Passa-Baixas Nutall 28
3.2.2.2 Parâmetros do Processador SAPHO 1 31
3.2.3 Atrasador 32
3.2.4 Memória FIFO 32
3.2.5 Registrador de posições ocupadas pela FIFO (Reg) 33
3.2.6 Processador DTW (SAPHO 2) 33
3.2.7 Máquina de Estados 34
3.3 Definição do Limiar de Novidade 34
4 RESULTADOS 36
4.1 Descrição do Hardware no Quartus e Consumo de Recursos 36
4.2 Resultados Simulação MatLab® x ModelSim® 37
4.2.1 Sinal com distúrbio transitório 38
4.2.2 Sinal com distúrbio *sag* 40
4.2.3 Sinal com harmônico variante no tempo 41
4.2.4 Sinal com variação de frequência 42
4.2.5 Tempo de processamento na simulação funcional 43
4.3 Resultados Experimentais 44
4.3.1 Resultados Signal Tap 45
4.3.1.1 Sinal senoidal sem distúrbios 45
4.3.1.2 Sinal com Variação de Frequência 45
4.3.2 Tempo de processamento 47

4.3.3 Resultados de Bancada no osciloscópio 49
4.3.3.1 Sinal com Distúrbio Transitório 49
4.3.3.2 Sinal com Distúrbio *Sag* 50
4.3.3.3 Sinal com Distúrbio Harmônico Variante no Tempo 51
4.3.3.4 Sinal com Variação de Frequência 51
5 CONCLUSÃO 54
APÊNDICE A – Códigos em Verilog desenvolvidos no trabalho 55
REFERÊNCIAS 66

1 INTRODUÇÃO

1.1 Contextualização

Os Sistemas Elétricos de Potência estão passando por um processo de transformação profunda, caracterizado modernização e digitalização. Esse avanço tecnológico tem possibilitado a coleta, o processamento e a análise de dados em larga escala, permitindo uma operação mais eficiente, segura e inteligente dos sistemas elétricos.

A evolução necessária para acompanhar esse crescimento requer a inclusão de novos equipamentos de monitoramento em tempo real para garantir uma boa Qualidade de Energia (QE). Tal contexto faz parte da consolidação das Redes Inteligentes (RI). Uma RI (ou *smart grid*) é uma rede de eletricidade caracterizada por fluxos bidirecionais de energia e dados, além da integração de tecnologias avançadas de informação/comunicação, sensoriamento, medição e controle. Essas funcionalidades possibilitam à RI uma operação flexível, confiável, resiliente, estável e sustentável (BABAYOMI *et al.*, 2023). O estabelecimento das RI no âmbito dos sistemas de transmissão e distribuição de energia elétrica surgiu da necessidade de se atender a alguns fatores, dentre os quais pode-se destacar:

1. aprimoramento da eficiência na produção e distribuição de eletricidade;
2. aumento da confiabilidade do sistema elétrico;
3. fornecimento aos usuários de informações que lhes permitam acompanhar e controlar o consumo e os custos de energia elétrica;
4. e estimação de parâmetros da rede a fim de contribuir com o desenvolvimento de soluções para mitigação dos problemas de QE no Sistemas Elétricos de Potência (SEP).

Essas motivações impulsionaram ações nos âmbitos industrial, regulatório e de pesquisa para o desenvolvimento contínuo das RI.

Considerando esse novo cenário que vem sendo formado, novos distúrbios e problemas de QE ainda não conhecidos podem surgir, ou seja, as novidades. Por isso, os sistemas de medição e monitoramento devem estar preparados para preservar qualquer informação que tenha relação com qualquer evento presente no sinal, pois a análise desses dados pode contribuir para a resolução de problemas de QE, como a classificação de distúrbios, identificação de assinaturas elétricas, estimação de contribuição harmônica, entre outros (MOSAYEBI; AZAD; ANVARI-MOGHADDAM, 2024).

Além disso, há a necessidade de se manipular e armazenar os dados de uma forma eficiente. Através do registro das novidades, é possível otimizar o espaço ocupado pelas gravações dos sinais de tensão e corrente obtidos das medições do SEP, preservando-se

apenas as informações relevantes presentes nos mesmos. Caso necessário, o sinal completo pode ser reconstruído a partir das amostras dos trechos contendo novidade, considerando a estimação de algum parâmetro como a frequência fundamental ao longo do período onde não há novidades (SILVA *et al.*, 2017). Nesse contexto, a aplicação de detectores de novidade pode ser útil.

Os detectores convencionais, projetados para identificar apenas os distúrbios clássicos de QEE (afundamentos e elevações de tensão, harmônicos, transitórios), podem não apresentar desempenho satisfatório. O conceito de Detecção de Novidade (DN) mostra-se adequado como base para o algoritmo de detecção, uma vez que busca identificar qualquer nova "anormalidade" presente no sinal (SILVA *et al.*, 2016).

A detecção de novidades em sinais do Sistema Elétrico de Potência (SEP) tem como objetivo identificar trechos dos sinais nos quais tenham ocorrido eventos potencialmente relevantes para análises, as quais podem ser feitas para aplicações tanto em tempo real quanto em modo off-line. Uma estratégia comumente empregada para essa finalidade é a segmentação dos sinais, que consiste em particioná-los em janelas ou *frames*. Após essa etapa, cada *frame* é analisado por meio de métricas específicas — sejam elas individuais ou comparativas — com o intuito de detectar aqueles *frames* que apresentam características de novidade, ou quebras de estacionariedade (KAPISCH, 2019). Ressalta-se que existem diferentes abordagens de segmentação, e a escolha do tipo de segmentação pode influenciar diretamente na seleção da técnica de detecção de novidades mais apropriada.

A segmentação pode ser feita de duas formas principais: fixa ou adaptativa. A segmentação fixa divide o sinal em *frames* de mesmo tamanho. O processo de comparação usa técnicas matemáticas mais simples, porém pode sofrer limitações de alto índice de falsos alarmes, principalmente em cenários de variação de frequência fundamental. Já a segmentação adaptativa, é capaz de dividir o sinal em *frames* de tamanho variado de acordo com a dinâmica presente no mesmo. Essa abordagem é mais otimizada para evitar detecções desnecessárias em caso de variação de frequência, porém pode requerer o uso de técnicas matemáticas mais complexas.

Nesse contexto, a segmentação adaptativa enfrenta como principal desafio a variação de frequência presente nos sinais analisados. Tais sinais, oriundos do Sistema Elétrico de Potência, possuem natureza senoidal, o que justifica a adoção da técnica de cruzamento por zero como critério para a delimitação do início e do fim de cada *frame*. Como resultado, os *frames* gerados podem apresentar diferentes quantidades de amostras, refletindo variações na frequência.

Dessa forma, o presente estudo tem como finalidade apresentar uma descrição da implementação, em *Field Programmable Gate Array* (FPGA), de um detector de novidades capaz de segmentar e analisar *frames* de tamanhos distintos baseado na métrica de similaridade dinâmica *Dynamic Time Warping* (DTW), aplicado a sinais de Qualidade

de Energia (QE). A implementação foi concebida de maneira a otimizar a utilização dos recursos de hardware através da utilização de processadores customizados embarcados em FPGA (SAPHO), apresentando uma abordagem eficiente, incorporando segmentação adaptativa, robustez à presença de ruído e capacidade de lidar com eventos caracterizados por variações de frequência, minimizando, assim, a ocorrência de falsos positivos.

1.2 Objetivos

O objetivo principal do trabalho é a implementação de um detector de novidades em FPGA baseado na técnica de similaridade dinâmica DTW. Para isso, objetivos secundários foram estabelecidos, conferindo ao sistemas algumas características de interesse, tais como:

- Capacidade de segmentação adaptativa de *frames*.
- Evitar detecção de novidades (falso alarme) em cenários de variação de frequência.
- Otimização de recursos de hardware através da utilização de processadores *soft-core* (SAPHO).
- Garantir que o sistema tenha robustez em relação ao ruído.

1.3 Produção Bibliográfica

Durante a realização das pesquisas desenvolvidas para este trabalho, os seguintes artigos foram publicados dentro do tema:

- CASTRO, Ana L. B.; RODRIGUES, Matheus S.; **CABRAL, Igor Daher**; KAPISCH, Eder B.; SILVA, Leandro R. M.; DUQUE, Carlos A.; RIBEIRO, Moisés V. Implementação em módulo FPGA de detector de novidades adaptativo baseado na DTW para sinais de qualidade de energia utilizando processadores soft-core em tempo real. In: SIMPÓSIO BRASILEIRO DE SISTEMAS ELÉTRICOS (SBSE), 9., 2025, São João del-Rei. Anais [...]. São João del-Rei:SBSE, 2025.
- **CABRAL, Igor Daher**; RODRIGUES, Matheus da Silva; KAPISCH, Eder Barboza; RIBEIRO, Victor Mendes; SILVA, Leandro Rodrigues Manso; DUQUE, Carlos Augusto. Implementação em FPGA de detector de novidades adaptativo baseado na DTW para sinais de qualidade de energia utilizando processadores soft-core. In: CONGRESSO BRASILEIRO DE AUTOMÁTICA (CBA), 25., 2024, Rio de Janeiro. Anais [...]. Rio de Janeiro: Sociedade Brasileira de Automática, 2024.

1.4 Estrutura do Trabalho

O trabalho foi dividido conforme segue. No capítulo I, é feita uma breve introdução contextualizando o tema de sistemas elétricos e detecção de novidades, apresentando os objetivos, a produção bibliográfica e a estrutura do trabalho. No Capítulo II, é feita uma revisão bibliográfica que passa por trabalhos que serviram de base para este, apresentando conceitos importantes e explicando a metodologia utilizada para detectar novidades. No Capítulo III, é apresentada a metodologia utilizada para implementação do detector de novidades baseado na DTW. No Capítulo IV, são apresentados os resultados com o intuito de demonstrar e comprovar o funcionamento do protótipo. Para isso, são apresentados resultados de consumo de hardware, simulações funcionais e testes experimentais. Por fim, no Capítulo V, é feita uma breve conclusão do trabalho. Como complemento, os códigos de linguagem descritiva de hardware do protótipo desenvolvido estão disponíveis no Apêndice A ao final do trabalho.

2 REVISÃO BIBLIOGRÁFICA

2.1 Detecção de Novidades em Sistemas Elétricos de Potência

A Detecção de Novidades (DN) é uma tarefa fundamental para o monitoramento e proteção de Sistemas Elétricos de Potência (SEP). Esses sistemas são suscetíveis a diversas perturbações, como falhas, distúrbios na qualidade de energia e variações nas condições operacionais. A detecção automática de eventos anômalos ou não previamente conhecidos permite respostas rápidas e eficazes, minimizando danos e interrupções (SILVA *et al.*, 2016).

No âmbito dos SEP, o uso de unidades de medição fasorial (PMUs) e *Merging Units* (MUs) tem possibilitado a obtenção de dados digitalizados em tempo real, transmitidos por protocolos como o *Sampled Values* (SV) (International Electrotechnical Commission, 2011). A análise contínua desses dados demanda técnicas capazes de identificar mudanças súbitas nos sinais elétricos, o que reforça a importância da detecção de novidades.

Para a detecção dessas novidades, podem ser utilizados diversos métodos de Detecção de Novidades (DN) e, dentro dos explorados em diversos trabalhos, temos:

- **Diferença das Energias de Quadros (DFE)** (SILVA *et al.*, 2017): é calculada como o valor absoluto da subtração da energia do quadro atual pela energia do quadro de referência, conforme mostrado na Equação (2.1).

$$\mathrm{DFE}(n) = |E(n) - E_{\mathrm{ref}}| \tag{2.1}$$

  Se a DFE for maior que um determinado limiar (*threshold*), um *quadro de novidade* é detectado, e esse quadro é então rotulado como o novo quadro de referência.

- **Morfologia Matemática (MM)** (FERNANDES *et al.*,): as operações são utilizadas no processamento de sinais com base na forma do sinal. A saída de uma operação morfológica depende da amostra atual do sinal e das amostras em sua vizinhança. O tamanho e a forma dessa vizinhança são determinados por uma função s, chamada de elemento estruturante. Essas operações, como dilatação, erosão, abertura e fechamento, são aplicadas com o objetivo de realçar ou suprimir determinadas características morfológicas do sinal.

  Para fins de Detecção de Novidades (DN), a MM é útil para identificar mudanças súbitas na morfologia do sinal. Quando um novo padrão ou forma não compatível com o contexto local (definido pelo elemento estruturante) aparece, ele gera um resíduo morfológico que pode ser usado como indicativo de um evento novo ou anômalo.

- **Energia do Resíduo** (ER): é calculada conforme a Equação (2.2), sendo o resíduo definido como a diferença ponto a ponto entre as amostras do *frame* atual e as do *frame* de referência.

$$\mathrm{ER} = \sum_{n=0}^{M-1} |x_c[n] - x_r[n]|^2 \tag{2.2}$$

  onde:

  - $x_c[n]$: amostra de índice n do quadro atual,
  - $x_r[n]$: amostra correspondente do quadro de referência,
  - M: número total de amostras por quadro.

  Essa métrica é utilizada na detecção de novidades ao comparar quadros consecutivos de um sinal. Um aumento significativo na energia do resíduo indica que o quadro atual difere substancialmente do quadro de referência, sugerindo a presença de um novo evento ou mudança no padrão do sinal.

- **Métricas de similaridade Dinâmica** (PIRES *et al.*, 2022): diversos tipos de métricas foram apresentadas e comparadas, tais como *Edit Distance on Real signal* (EDR), *Time Warp Edit Distance* (TWED) e, por fim, a *Dynamic Time Warping* (DTW). As técnicas foram testadas e comparadas entre si com diversos tipos de distúrbios elétricos diferentes e a que se destacou pela simples implementação e desempenho foi a DTW. Por isso, tal técnica foi escolhida como métrica de detecção de novidades para o presente trabalho.

Uma das maiores vantagens da DTW para aplicações que requerem segmentação adaptativa é que ela permite a comparação e análise de *frames* de tamanhos distintos, possibilitando uma segmentação adaptada aos distúrbios que podem ocorrer nos sinais de QE, além de ser robusta para detectar alterações significativas no comportamento do sistema (PIRES *et al.*, 2022). A DTW é especialmente eficaz para identificar eventos transitórios, falhas ou partidas de grandes cargas, pois além de permitir a comparação de segmentos com diferentes comprimentos, é capaz de fornecer o melhor alinhamento entre esses, mesmo quando ocorrem distorções e deslocamentos temporais.

Neste trabalho, a detecção de novidades foi implementada com base na comparação de *frames* pela DTW considerando-se a comparação entre o *frame* de referência e o *frame* atual, sendo o *frame* de referência o último detectado. Quando a distância DTW entre os dois *frames* comparados ultrapassa um limiar definido, é sinalizada uma novidade, indicando possível alteração operacional ou anomalia no SEP. Essa abordagem se mostra eficiente e adequada para aplicações em tempo real, compatível com as exigências atuais de sistemas de proteção e automação.

2.2 *Dynamic Time Warping* (DTW)

A comparação de séries temporais é uma tarefa essencial em diversas aplicações, como reconhecimento de padrões, diagnóstico de falhas e monitoramento de sistemas elétricos de potência. Dentre as técnicas disponíveis, destacam-se as métricas tradicionais de distância como a *Euclidean Distance* (ED) e a *Manhattan Distance* (MD), amplamente utilizadas devido à sua simplicidade computacional (KEOGH; RATANAMAHATANA, 2005). Contudo, tais métricas apresentam limitações significativas em contextos onde há variações não lineares na escala temporal ou distorções locais de tempo, fenômenos comuns em sinais reais, especialmente sob a presença de ruído ou eventos transitórios.

Em comparação, a *Dynamic Time Warping* (DTW) destaca-se por sua robustez na análise de séries temporais com variações na velocidade ou pequenas deformações no tempo. Enquanto a ED exige que as sequências sejam comparadas ponto a ponto, a DTW permite alinhamentos não-lineares entre os elementos das séries, resultando em uma correspondência mais flexível e precisa, visto que ela possui a capacidade de relacionar um ponto de um segmento a múltiplos pontos de outro ao mesmo tempo que relaciona múltiplos pontos a um ponto. Esta característica torna a DTW particularmente vantajosa em sistemas onde a sincronia temporal não pode ser garantida, como no monitoramento em tempo real de sinais elétricos com ruídos e variações transitórias.

2.2.1 Vantagens da DTW

As principais vantagens da DTW em relação às técnicas tradicionais incluem:

- **Robustez a Desvios Temporais**: A DTW é capaz de identificar padrões semelhantes mesmo que ocorram em diferentes escalas ou posições temporais.
- **Adaptação a Variações Locais**: A flexibilidade no alinhamento permite a DTW lidar eficientemente com distorções locais no tempo, mantendo alta precisão na comparação.
- **Aplicabilidade em Cenários Reais**: Ideal para aplicações de detecção de eventos em sinais ruidosos, como falhas ou anomalias em sistemas elétricos.

Por outro lado, ressalta-se que a DTW possui um custo computacional mais elevado, com complexidade quadrática em relação ao comprimento das séries ($O(N^2)$).

2.2.2 Funcionamento do Algoritmo DTW

O algoritmo DTW busca encontrar o caminho ótimo de alinhamento entre duas sequências temporais $X = \{x_1, x_2, ..., x_N\}$ e $Y = \{y_1, y_2, ..., y_M\}$ minimizando o custo acumulado ao longo do alinhamento.

1. **Matriz de Custo Local**: Inicialmente, é construída uma matriz de custo D de dimensões $N \times M$, onde cada elemento $D(i, j)$ representa o custo local de alinhamento entre os pontos x_i e y_j, geralmente calculado como $|x_i - y_j|^2$.

2. **Matriz de Custo Acumulado**: A seguir, calcula-se a matriz de custo acumulado C, onde cada elemento $C(i, j)$ representa o custo mínimo necessário para alcançar a posição (i, j) a partir do início $(1, 1)$. A recorrência é definida por:

$$C(i,j) = D(i,j) + \min \begin{cases} C(i-1,j) \\ C(i,j-1) \\ C(i-1,j-1) \end{cases} \tag{2.3}$$

   com as condições de fronteira $C(1, 1) = D(1, 1)$ e $C(i, 0) = C(0, j) = \infty$.

3. **Caminho Ótimo**: A partir do elemento $C(N, M)$, realiza-se um retrocesso até $C(1, 1)$, determinando o caminho ótimo de alinhamento que minimiza o custo acumulado.

4. **Distância DTW**: O valor final da distância DTW entre as sequências é dado por $C(N, M)$.

Para melhor compreensão da técnica, considere o seguinte exemplo utilizando os vetores $x = [3, 4, 0, 7, 1]$ e $y = [2, 5, 8, 4]$. Neste caso, temos $L = 5$ e $C = 4$. Além disso, vamos utilizar um método mais linear, que é o aplicado no trabalho, no qual não é calculado o custo local em uma matriz separada, mas sim na matriz de custo acumulado diretamente. Iremos chamar essa matriz de Matriz DTW.

1º passo: Montagem da Matriz DTW

Inicialmente, constrói-se uma matriz vazia D de ordem $L \times C = 5 \times 4$, com 20 posições. Os elementos do vetor x são dispostos à esquerda, correspondendo às linhas da matriz D, enquanto os elementos de y são dispostos na parte superior, correspondendo às colunas. Vale ressaltar que a escolha da disposição dos vetores (linhas ou colunas) não altera o valor final da DTW, apenas a matriz resultante será a transposta.

2º passo: Preenchimento da Primeira Linha e Primeira Coluna

Calcula-se inicialmente $D(1, 1)$ como a distância euclidiana entre $x(1)$ e $y(1)$:

$$D(1,1) = \sqrt{(3-2)^2} = 1$$

Em seguida, preenche-se a primeira linha somando sempre a distância ao valor anterior. Por exemplo:

$$D(1,2) = \sqrt{(3-5)^2} + D(1,1) = 2 + 1 = 3$$
$$D(1,3) = \sqrt{(3-8)^2} + D(1,2) = 5 + 3 = 8$$

Esse procedimento se repete até completar a linha.

Para a primeira coluna, soma-se a distância com o valor acima:

$$D(2,1) = \sqrt{(4-2)^2} + D(1,1) = 2 + 1 = 3$$
$$D(3,1) = \sqrt{(0-2)^2} + D(2,1) = 2 + 3 = 5$$

Esse procedimento é repetido para os demais elementos da primeira coluna.

A Figura 1 ilustra a matriz D com os vetores x e y, além do preenchimento da primeira linha (em azul) e da primeira coluna (em amarelo).

Calcular a primeira linha da matriz, $D(1,2 \rightarrow 4)$
Calcular a primeira coluna da matriz, $D(2 \rightarrow 5,1)$

	2	**5**	**8**	**4**
3	1	3	8	9
4	3			
0	5			
7	10			
1	11			

Figura 1: 1º e 2º passos para a construção da matriz D.

3º passo: Preenchimento da Matriz Restante

Para preencher as demais posições da matriz, utiliza-se a equação da DTW:

$$D(n,m) = \sqrt{(x(n) - y(m))^2} + \min\{D(n-1,m),\ D(n,m-1),\ D(n-1,m-1)\}$$

Esse processo é repetido até o preenchimento completo da matriz D. O valor da DTW é encontrado em $D(L,C) = D(5,4) = 11$.

A Figura 2 mostra o preenchimento completo da matriz com o valor final destacado.

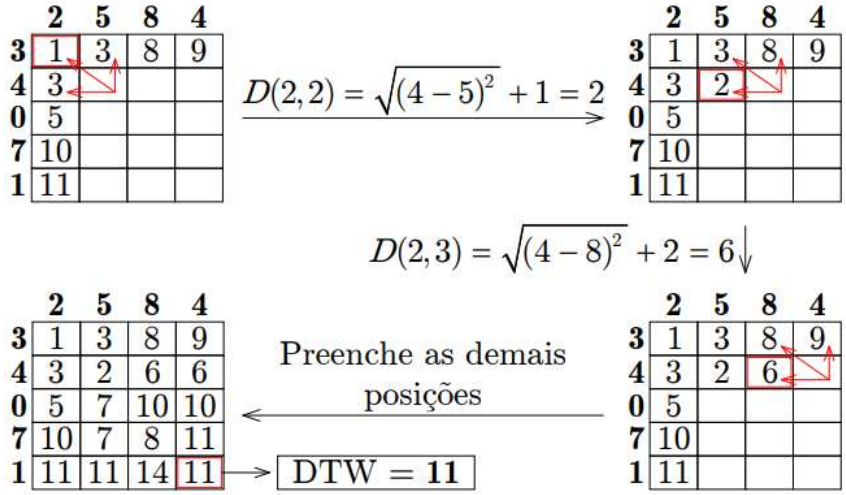


Figura 2: 3º passo para a construção da matriz D. Valor da DTW: 11.

Esse valor final é comparado com um limiar previamente definido para se dizer se o sinal é uma novidade ou não. O limiar depende da quantização e do grau de ruído do sistema, por isso, deve ser definido especificamente para cada aplicação.

A Figura 3 apresenta a comparação dos dois sinais que foram utilizados para a análise da similaridade por meio do algoritmo DTW. O sinal x é um sinal senoidal com 100 pontos, enquanto o sinal y contém 80 pontos e está deslocado em fase. Para que ambos os sinais pudessem ser comparados no mesmo plano temporal, o sinal y foi interpolado linearmente para o intervalo de amostragem do sinal x. Isso foi feito apenas para visualização da figura.

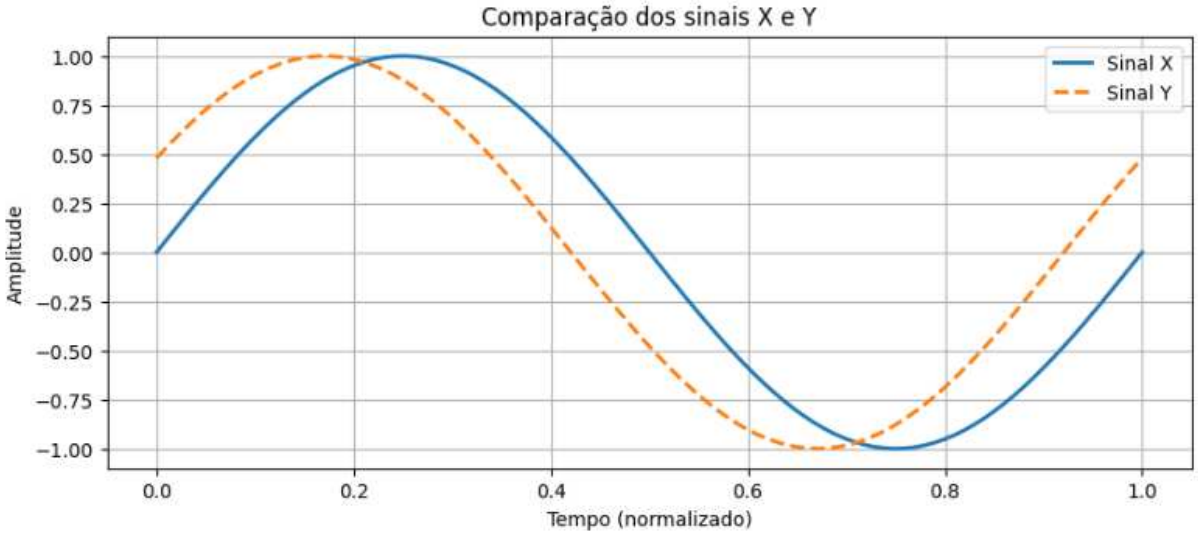


Figura 3: Sinais utilizados.

Essa representação visual permite observar diferenças e similaridades iniciais entre os sinais antes da aplicação do algoritmo DTW. O método é especialmente útil para comparação de sinais que possuem diferentes frequências e/ou que estejam desfasados

temporalmente, como no caso apresentado. Podendo assim, ser utilizado como uma métrica de detecção de novidades, já que o resultado se torna um parâmetro de semelhança entre os sinais.

Aplicada a métrica de comparação, podemos observar na Figura 4 o mapa de calor, que representa o custo acumulado. Pode-se ver que o método escolhe o caminho de menor custo para comparar os sinais. Dessa forma, temos o valor final da DTW como somatório dos elementos do caminho de menor custo. No exemplo mostrado, os sinais apresentaram bastante similaridade, já que são ciclos senoidais apenas ligeiramente defasados, resultando num valor de DTW que seria menor do que uma comparação de sinais mais diferentes, digamos, uma senoide e uma onda quadrada. Por ser um valor pequeno (para essa aplicação), os sinais são semelhantes de acordo com a métrica utilizada. Num caso ideal, por exemplo, comparando-se dois sinais exatamente iguais, o caminho de menor custo percorreria sobre e diagonal da matriz DTW e seu somatório resultaria em zero, revelando o máximo de similaridade possível. Por outro lado, quanto mais diferentes forem os sinais comparados, maior o valor resultante da métrica. A forma de cálculo apresentada pelas Figuras 1 e 2 já fornece o resultado da métrica ao fim do preenchimento da matriz, localizado na última posição.

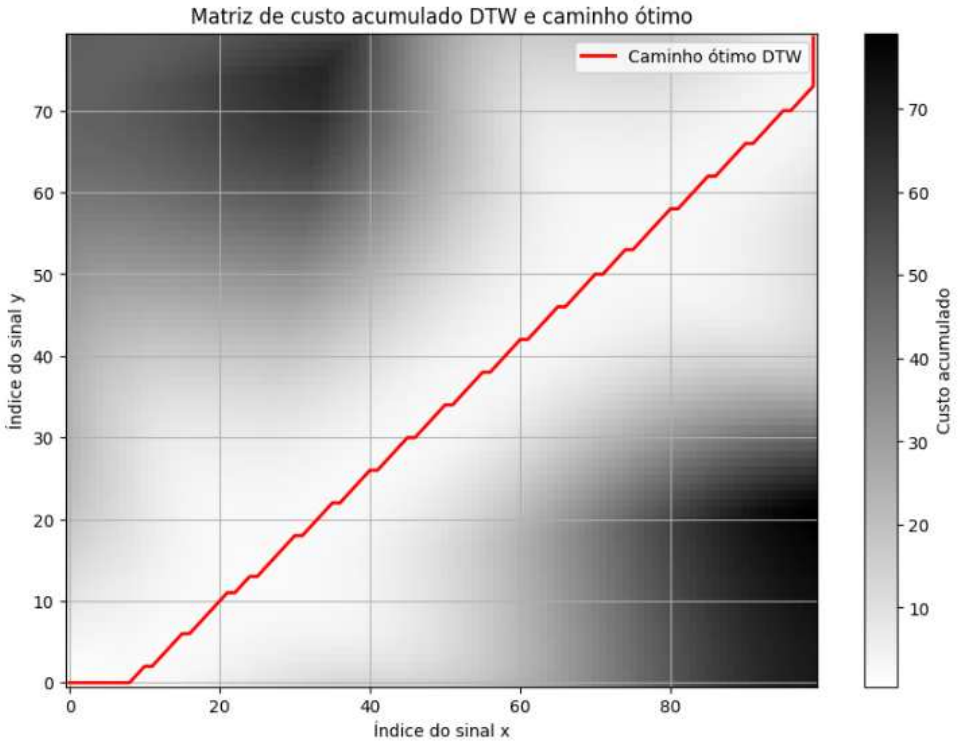


Figura 4: Mapa de calor do custo DTW com o caminho ótimo.

Devido às suas características, a DTW tem sido amplamente adotada em sistemas inteligentes de monitoramento de sinais elétricos, oferecendo uma solução robusta e eficiente para a detecção de eventos e análise de qualidade de energia. Neste trabalho, a DTW foi

empregada como base para a implementação de um detector de novidades embarcado em FPGA, com foco em aplicações de monitoramento em tempo real de Sistemas Elétricos de Potência.

2.3 Segmentação de Sinais com Variação de Frequência

A segmentação de sinais é uma etapa fundamental em diversos sistemas de processamento, especialmente quando os sinais apresentam variações de frequência ao longo do tempo, como ocorre frequentemente em sinais elétricos de potência. A separação por quantidade de pontos por ciclo é a abordagem mais simples, mas não é eficiente em separar *frames* em que haja variação de frequência. Logo, técnicas de segmentação adaptativa são utilizadas pois são capazes de demarcar adequadamente os trechos do sinal para posterior análise ou processamento, mesmo diante de variações dinâmicas e de ruídos.

Dentre os métodos de segmentação, destaca-se a técnica de detecção por cruzamento por zero (*zero crossing*), que identifica pontos em que o sinal cruza o eixo zero para delimitar segmentos. Esta abordagem é amplamente utilizada por sua simplicidade conceitual, baixo custo computacional e facilidade de implementação em sistemas embarcados e de tempo real. Apesar de sua sensibilidade ao ruído, o que pode resultar em falsos cruzamentos por zero, a segmentação por cruzamento por zero apresenta desempenho satisfatório para sinais com variações frequenciais moderadas e pode ser aprimorada com o uso de filtros passa-baixas para minimizar erros.

Em contrapartida, métodos mais sofisticados, como as técnicas baseadas em transformada *wavelet* (PROCHAZKA; KUKAL; VYSATA, 2008), oferecem maior robustez e precisão na segmentação de sinais não estacionários, especialmente em cenários com alta variabilidade de frequência e ruído intenso. No entanto, esses métodos demandam maior complexidade computacional e podem não ser viáveis para aplicações com restrições de hardware ou requisitos de latência muito rigorosos, como é o caso deste trabalho.

Assim, a escolha do método de segmentação deve considerar o equilíbrio entre complexidade, precisão e requisitos de hardware. Para aplicações em Sistemas Elétricos de Potência (SEP), onde o monitoramento em tempo real e a resposta rápida são críticos, a segmentação por cruzamento por zero combinada com um filtro representa uma solução eficaz e prática, permitindo uma segmentação eficiente e confiável dos sinais monitorados.

3 IMPLEMENTAÇÃO DO DETECTOR DE NOVIDADES

Este capítulo descreve as metodologias de detecção de novidades e de implementação em FPGA do sistema proposto.

3.1 Metodologia do Detector de Novidades

A Figura 5 mostra a metodologia adotada para o detector de novidades através de um fluxograma com as principais etapas da técnica de detecção.

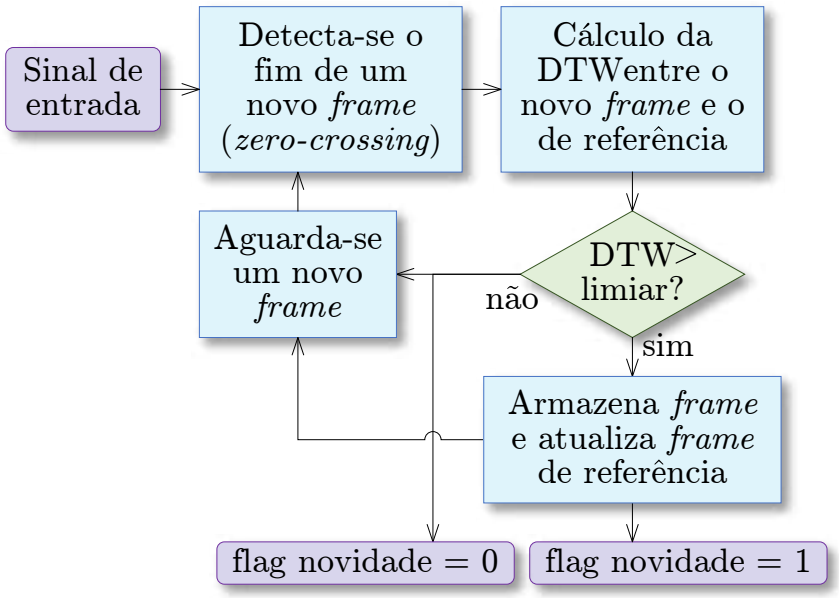


Figura 5: Metodologia do Detector de Novidades.

Primeiramente, o sinal de entrada é segmentado em *frames* utilizando-se o cruzamento por zero. Ao fim de um novo *frame*, calcula-se a métrica DTW entre o *frame* atual e o de referência. A partir do valor obtido da comparação, compara-se a métrica com um limiar pré determinado. Se o valor da métrica ultrapassar o limiar, é detectada uma novidade e o *frame* deve ser armazenado, pois é um *frame* de novidade. Essa detecção é indicada pela *flag* de novidade = 1 e o *frame* detectado passa a ser o novo *frame* de referência. Caso a métrica DTW não ultrapasse o limiar, significa que o *frame* não é de novidade e pode ser descartado (*flag* de novidade = 0), e aguarda-se um novo *frame* para ser analisado.

3.2 Metodologia de Implementação do Detector de Novidades em FPGA

A fim de implementar em FPGA a metodologia mostrada na Figura 5, foi utilizada a estrutura representada na Figura 6.

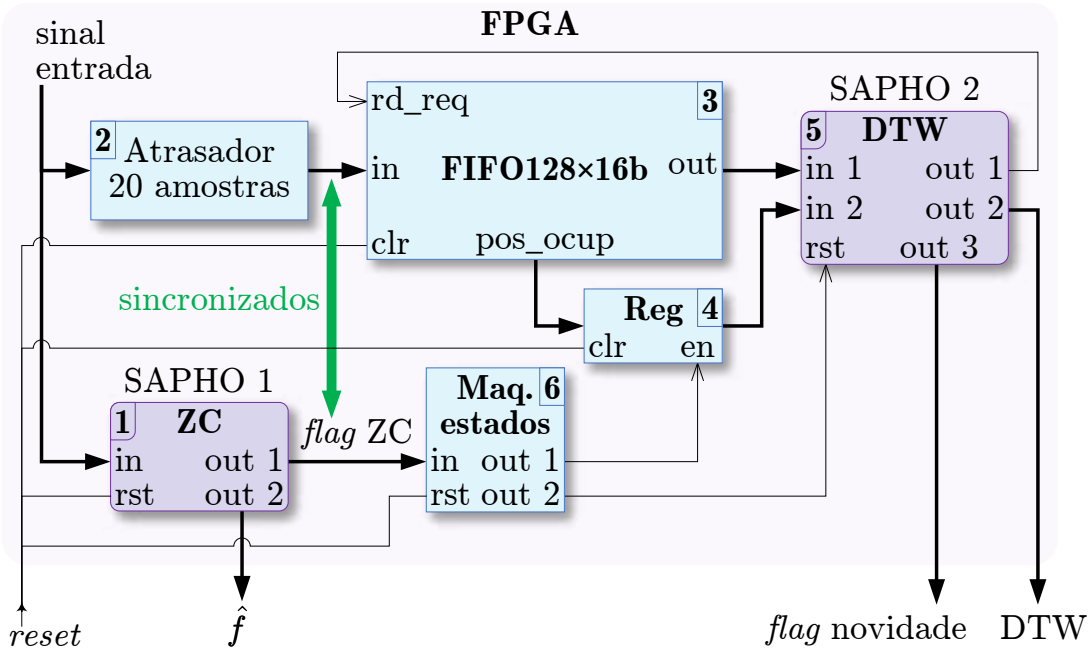


Figura 6: Estrutura do Hardware.

3.2.1 SAPHO

Os blocos de maior importância que fazem o processamento dos sinais neste trabalho são formados por processadores *soft-core Scalable Architecture Processor for Hardware Optimization* (SAPHO), projeto desenvolvido pelo Núcleo de Instrumentação e Processamento de Sinais (NIPS) da UFJF[1]. Por essa razão, será descrito brevemente a seguir o funcionamento básico da sua *Integrated Development Environment* (IDE).

O SAPHO é um ambiente de programação, compilação, sintetização de hardware, depuração e análise de processadores customizáveis em FPGA. Sua IDE utiliza compiladores C e Assembler para gerar automaticamente um códigos em HDL (Verilog) de processadores *soft-core* que podem ser utilizados em sistemas de processamento digitais complexos e interconectados, os quais podem ser importados para outros sintetizadores, como o Quartus II®. Cada processador *soft-core* pode ser construído de acordo com as configurações desejadas pelo usuário para uma determinada aplicação. Os comandos utilizados no código de programa que será carregado ao processador determinam quais recursos de hardware serão utilizados da FPGA. Dessa forma, o SAPHO é capaz de sintetizar apenas os blocos necessários para o funcionamento, otimizando o consumo de lógica do chip de FPGA. A IDE permite que se programe o processador a partir de uma linguagem formada por um subconjunto da linguagem C, denominada de C+-, ou até mesmo por Assembly.

Dessa forma, os processadores desenvolvidos, são descritos em hardware já são parametrizados automaticamente para receber a quantidade exata de memória e utilizar apenas a lógica necessária para o programa, consumindo o menor espaço possível na placa

[1] https://www.nipscern.com/

de desenvolvimento FPGA e otimizando o hardware.

A Figura 7 mostra uma imagem de uma das telas da versão da IDE do SAPHO utilizada para este trabalho. Na parte superior, há algumas ferramentas para criação de processadores, parametrização e compilação dos processadores desenvolvidos. Na parte esquerda, é possível ver uma árvore de projeto com instâncias dos processadores desenvolvidos. A tela principal mostra um código em C+- utilizado para o desenvolvimento de um dos processadores presentes neste trabalho.

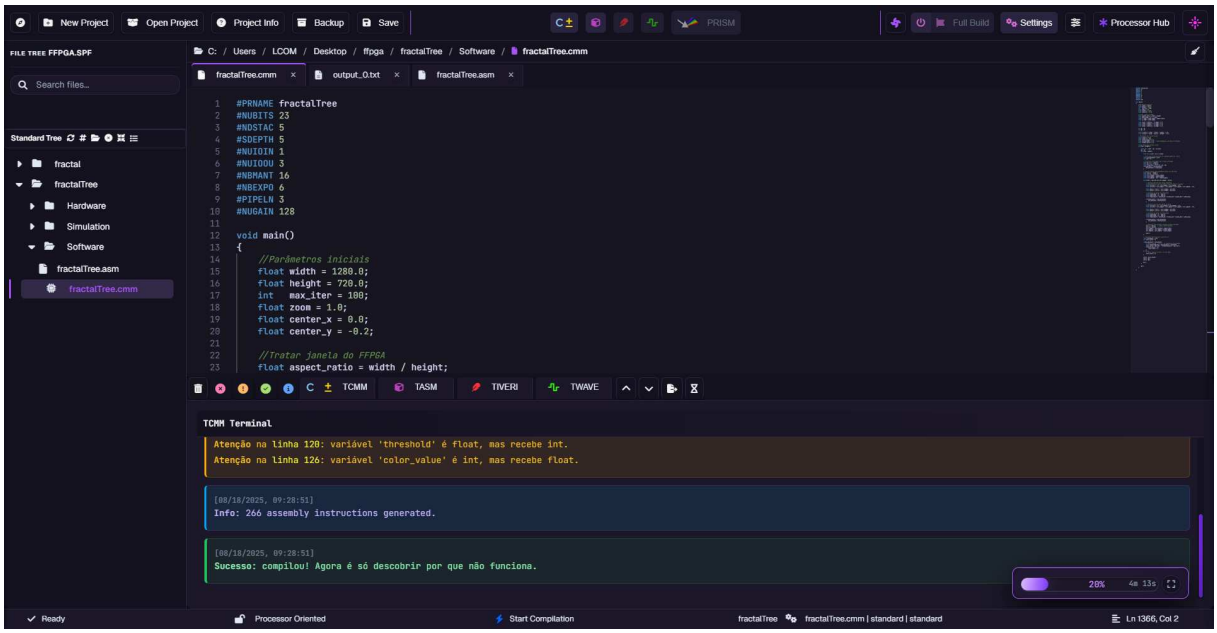


Figura 6: Imagem da versão da IDE do SAPHO utilizada neste trabalho.

É importante observar que esta IDE está em constante atualização, recebendo novos recursos e aprimoramentos dentro do projeto de desenvolvimento do SAPHO realizado no NIPS. A versão apresentada no presente trabalho pode diferir daquela encontrada no repositório do projeto.

No framework podemos escolher as seguintes configurações iniciais:

- Ponto flutuante ou fixo; Se flutuante:
  - Número de bits de Mantissa;
  - Número de bits de Expoente;
- Número de entradas;
- Número de saídas;
- Tamanho das *stacks* de memória.

Pode-se observar na Figura 6 os blocos de hardware do detector desenvolvido em FPGA, as subseções seguintes vão mostrar detalhadamente como cada módulo do sistema funciona.

3.2.2 Processador Zero-Crossing (SAPHO 1)

Este processador é responsável pela segmentação adaptativa do sinal de entrada, detectando os momentos de cruzamento por zero. Para evitar falsas detecções causadas por harmônicos ou ruídos de alta frequência, deve-se utiliza um filtro passa-baixas antes da detecção do cruzamento. Assim, apenas a frequência fundamental é aplicada ao detector do cruzamento por zero, mas o sinal segmentado é o sinal original.

3.2.2.1 Filtro Passa-Baixas Nutall

Como a proposta do trabalho é realizar o processamento em tempo real com a possibilidade de ocorrer variações de frequência, precisa-se utilizar um filtro com fase linear e, portanto, atraso de grupo constante. Para isso, optou-se pelo uso de um filtro FIR com fase linear de modo que o atraso não varie com a frequência fundamental do sinal. Para isso, foram testados vários filtros FIR de fase linear baseados no formato da janela, tais como Retangular, Blackman, Hamming, Kaiser, Nutall, dentre outros. Depois de testes empíricos, optou-se pelo filtro Nutall, visto que o mesmo promove alta rejeição na região acima da fundamental.

O projeto do filtro foi realizado no MatLab®, verificou-se que com ordem 40 o filtro já atendia as especificações desejadas. Assim, o atraso de grupo é de 20 amostras, como pode ser observado na Figura 8.

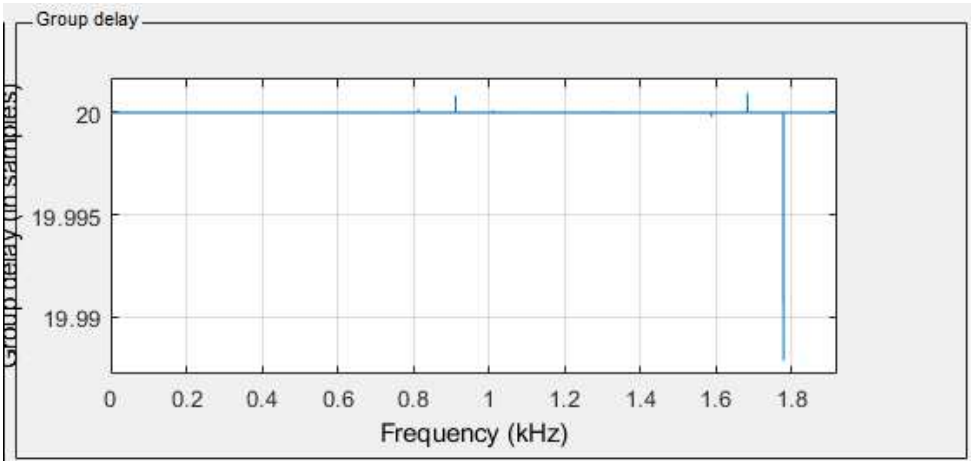


Figura 8: Atraso de grupo do filtro

As curvas de resposta em frequência de magnitude e fase podem ser vistas nas Figuras 9 e 10, respectivamente. Pode-se observar que a fase é linear, conforme requisitos do projeto.

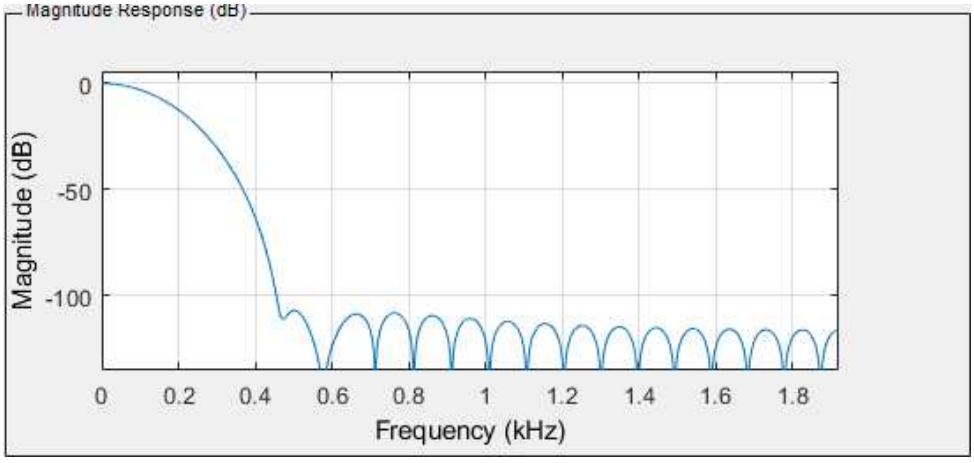


Figura 9: Módulo da resposta em frequência do filtro Nutall utilizado.

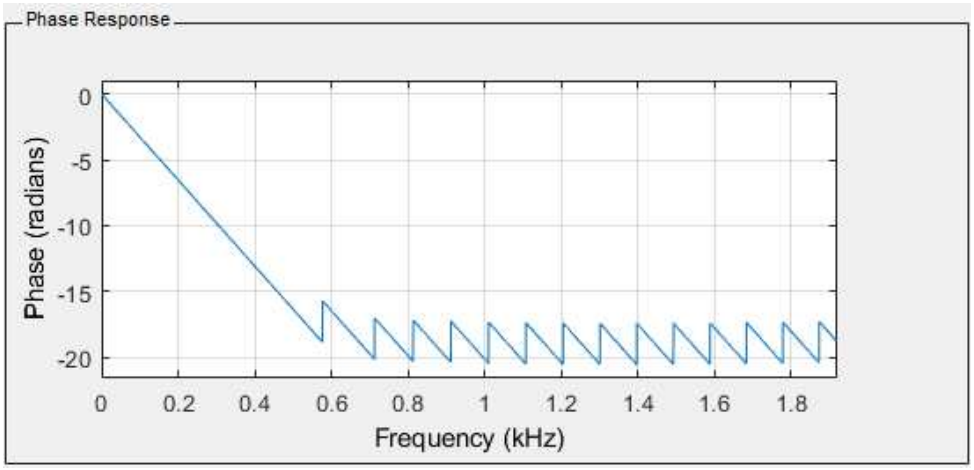


Figura 10: Fase da resposta em frequência do filtro Nutall utilizado.

Na Figura 11, pode-se ver a resposta ao impulso discreto $\delta[n]$ do filtro, revelando o formato da janela no domínio do tempo, composta pelos coeficientes do filtro Nutall.

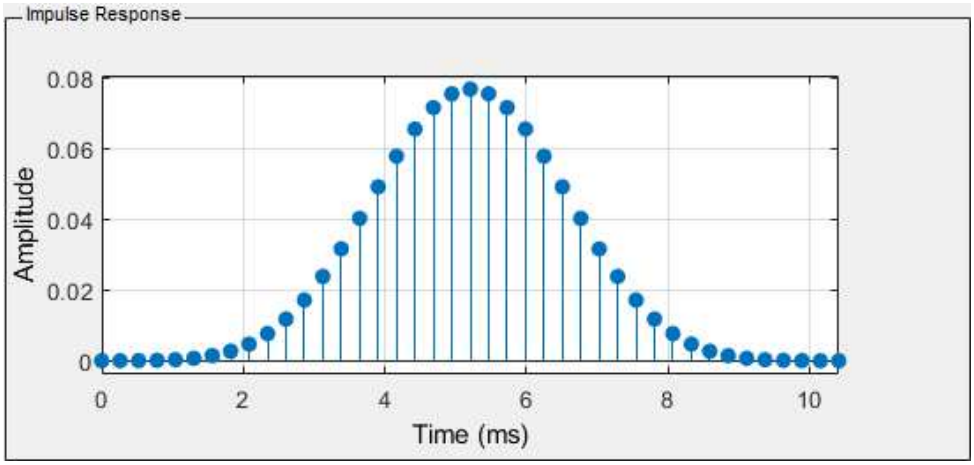


Figura 11: Resposta ao impulso do filtro Nutall utilizado.

A resposta ao impulso do filtro $H(x)$ é dada por:

$$H(x) = \sum_{n=0}^{40} h[n] \cdot \delta(x - n)$$

onde os coeficientes $h[n]$ são:

Tabela 1 – Coeficientes do vetor $h[n]$

n	$h[n]$	n	$h[n]$
0	0.001844615	11	17.154970056
1	0.007943906	12	23.814573858
2	0.035533285	13	31.605975101
3	0.115945571	14	40.210096838
4	0.307036495	15	49.142600955
5	0.703502592	16	57.790311524
6	1.443395263	17	65.475020225
7	2.707469230	18	71.536330294
8	4.708121280	19	75.420268900
9	7.665915589	20	76.758040491
10	11.774124177		

Os demais coeficientes seguem simetricamente: $h[21] = h[19]$, $h[22] = h[18]$, ..., $h[40] = h[0]$.

Após a pré-filtragem passa-baixas pelo filtro Nutall, a lógica do processador analisa amostra por amostra que chega na entrada e compara com a anterior para saber se houve troca do sinal positivo para o negativo, assim caracterizando um cruzamento por zero descendente. É importante destacar que o cruzamento foi padronizado para ser somente detectado quando o sinal está decrescendo, assim, uma senoide é dividida em um ciclo completo. Quando acontece o cruzamento, uma *flag* é acionada que está ligada à uma máquina de estados que controla o sistema. Esta máquina será explicada mais à frente.

Além da detecção da amostra em que ocorreu o cruzamento por zero, o processador SAPHO 1 também realiza a estimação da frequência fundamental de forma precisa através de uma interpolação linear entre as amostras imediatamente antes e logo após o cruzamento. Para isso, são definidos dois parâmetros, N_a e N_b, que determinam a distância entre as amostras e o cruzamento pelo zero. Por meio da interpolação linear, esses parâmetros podem ser estimados através de uma análise geométrica, semelhante ao gráfico apresentado na Figura 12. Os parâmetros são expressos da seguinte forma:

$$N_a = \frac{x[n]}{x[n] - x[n-1]}, \quad N_b = \frac{x[n-1]}{x[n-1] - x[n]} \tag{3.1}$$

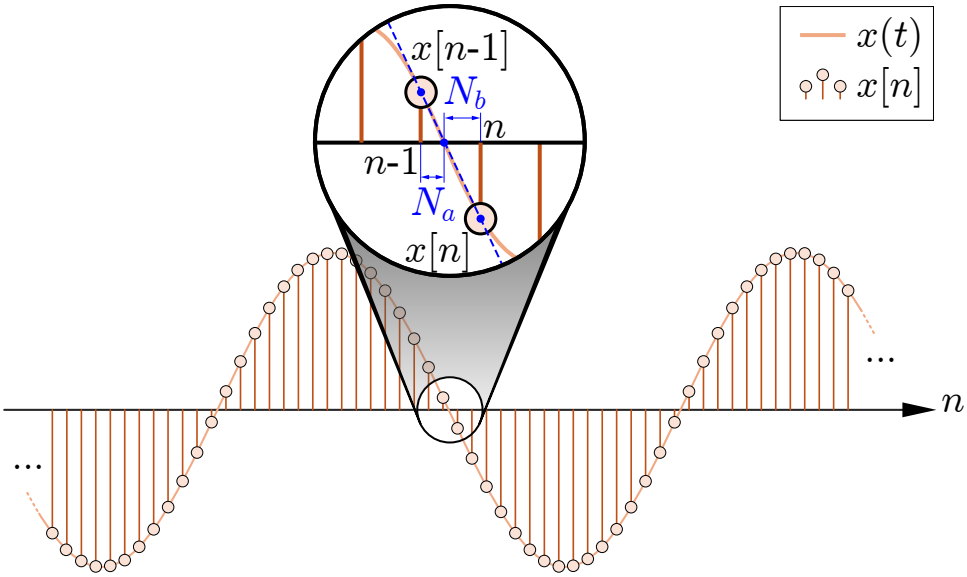


Figura 12: Amostras no instante do cruzamento por zero: correção por interpolação linear.

Este procedimento determina uma aproximação muito mais exata do momento real de cruzamento por zero do sinal avaliado $x(t)$, o qual ocorre entre as amostras atual $x[n]$ e anterior $x[n-1]$.

Dessa forma, é possível extrair a frequência estimada pelo método Zero-Crossing. Para isso, leva-se em conta quantos pontos tem em um ciclo, ou seja, quantas amostras foram identificadas entre um ZC e outro, e a frequência de operação do conversor AD, incluindo-se na estimação a interpolação linear. Assim, temos como segmentar o sinal dinamicamente pelos momento de ZC além de acompanhar uma estimativa da frequência do sinal com um custo computacional otimizado pelo processador SAPHO.

3.2.2.2 Parâmetros do Processador SAPHO 1

Para o processador Zero-Crossing (SAPHO 1), foram descritas as seguintes características:

- 23 bits de Mantissa;
- 8 bits de Expoente;
- 1 bit de sinal;
- Ponto flutuante;
- 1 entrada;
- 2 saídas.

3.2.3 Atrasador

O atrasador em FPGA é dado por uma série de registradores que, a cada *clock*, comutam para o valor do registrador anterior, assim, atrasando o sinal na quantidade desejada. Nesse trabalho, o atrasador é de 20 amostras para compensar o filtro passa-baixas utilizado.

O objetivo do atrasador nesse projeto é garantir que o sinal de entrada seja sincronizado com o sinal filtrado, utilizado para identificar os momentos de cruzamento, e portanto, as divisões dos *frames*. Assim, o sinal é seccionado corretamente.

Caso não houvesse um atrasador e o sinal tivesse a frequência variante, ele seria dividido de maneira errada, pois a variação de frequência resultaria em segmentos com tamanhos diferentes que acabariam resultando em *frames* com pontos a mais ou a menos dependendo da variação que seriam segmentados de maneira errada. A fim de ilustrar tal erro, a Figura 13 mostra um sinal senoidal puro que foi filtrado pelo filtro passa-baixas Nutall de ordem 40, mas segmentando sem atrasar de 20 amostras. As divisões entre *frames* são representadas por linhas verticais tracejadas em verde.

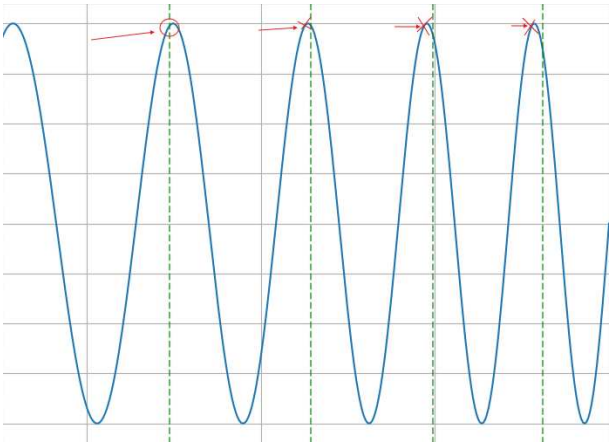


Figura 13: Sinal seccionado sem atrasar de 20 amostras.

Pode-se observar na Figura 13 que o sinal não é segmentado com um ciclo exato, pois o tamanho do ciclo muda a cada *frame* por conta da variação de frequência. Por isso, foi utilizado o atrasador. Com ele, o sinal original cruza pelo zero simultaneamente com a saída do filtro passa-baixas.

3.2.4 Memória FIFO

Foi utilizada uma memória *First In First Out* (FIFO) para servir de *buffer* em formato de fila. A quantidade de posições foi definida levando em conta que os sinais amostrados teriam 64 pontos por ciclo de 60 Hz em uma amostragem de 3840 Hz (amostragem configurada no ADC utilizado). Além disso, adotou-se um comprimento máximo de 75 amostras para os vetores utilizado na comparação da DTW a fim de não tornar o processo

de cálculo da métrica muito custoso. Sendo assim, a FIFO utilizada foi configurada com comprimento sendo potência de 2 acima de 75, resultando em 128 posições. A cada amostra, a FIFO é preenchida com uma nova posição. Ao ocorrer um cruzamento por zero, a FIFO possui um *frame* e pode ser descarregada.

Foi utilizado o recurso de *Mega Wizard Plug-in* do Quartus II para fazer o módulo da FIFO com as configurações mostrada na Figura 14.

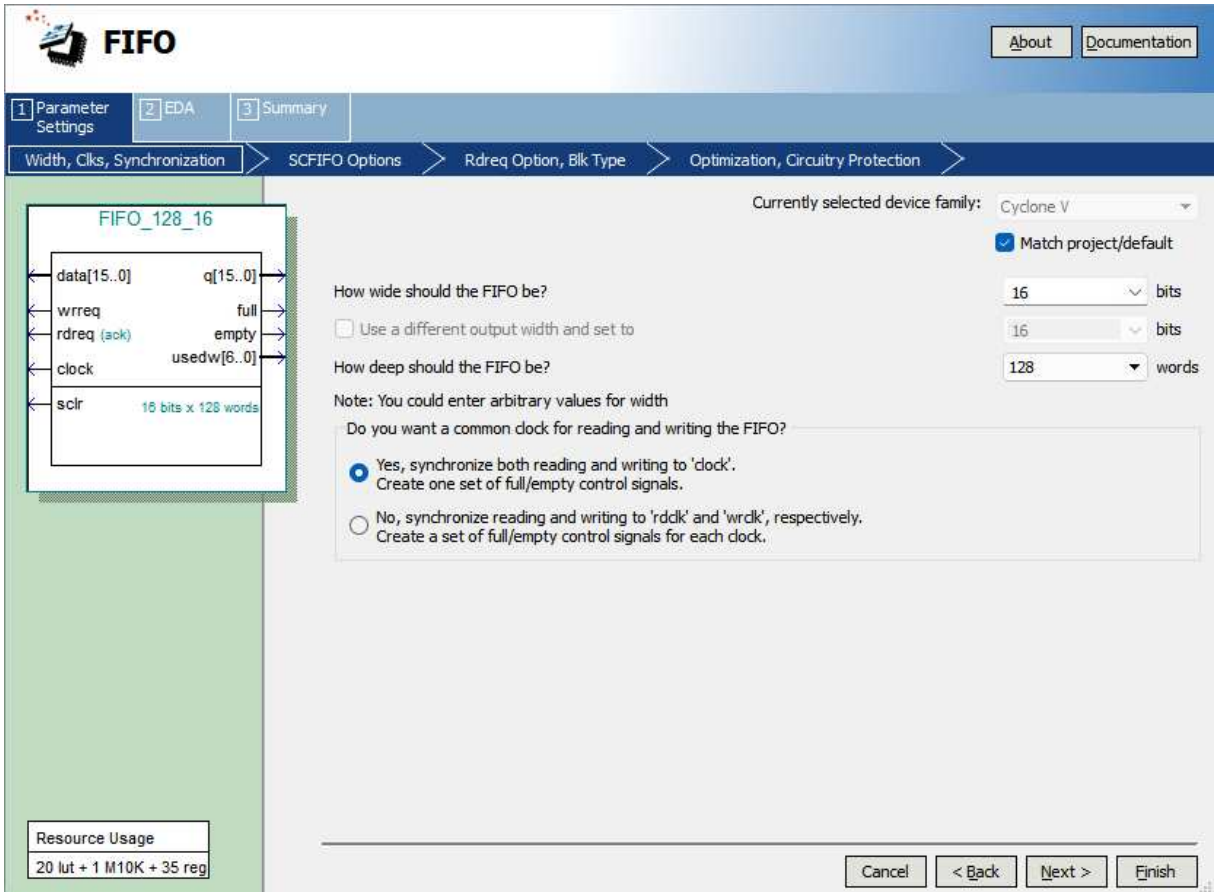


Figura 14: *Mega Wizard* da FIFO.

3.2.5 Registrador de posições ocupadas pela FIFO (Reg)

O registrador Reg guarda quantas posições estão ocupadas pela memória FIFO pois o processador DTW precisa dessa informação do comprimento do vetor a ser descarregado da FIFO de maneira correta e também para montar a Matriz DTW que é NxM sendo N e M o tamanho dos *frames* analisados.

3.2.6 Processador DTW (SAPHO 2)

Para o processador DTW, o processador descrito no SAPHO possui as seguintes características:

- 32 bits;

- Ponto fixo;
- 2 entradas;
- 3 saídas,

O processador DTW tem o seguinte fluxo de trabalho:

⋆ **Passo 1**: o processador DTW recebe o valor do comprimento do vetor do *frame* que será recebido. Esse valor é o que está armazenado no Reg (bloco 4 da Figura 6);

⋆ **Passo 2**: requisita as amostras da memória FIFO até esvaziar completamente seu conteúdo, constituindo o *frame* atual do sinal;

⋆ **Passo 3**: faz o cálculo da DTW comparando o *frame* atual com o *frame* de referência previamente armazenado da detecção de novidade anterior (inicialmente este *frame* é nulo).;

⋆ **Passo 4**: compara com o limiar;

⋆ **Passo 5**: atualiza o *frame* de referência se necessário;

⋆ **Passo 6**: exporta o valor DTW e atualiza a *flag* de novidade.

3.2.7 Máquina de Estados

A máquina de estados coordena o funcionamento dos outros blocos, ela recebe a *flag* de Zero-Crossing e, quando este está na borda de subida, habilita o Reg que guarda quantas posições ocupadas na FIFO. Após isso, a máquina reseta o processador DTW (SAPHO 2) para que o mesmo requisite a leitura do valor do Reg em uma de suas entradas (in 1) e receba o *frame* atual na outra entrada (in 2). A máquina garante o sincronismo e coordenação dos demais blocos.

3.3 Definição do Limiar de Novidade

Para saber se o sinal é uma novidade ou não, é necessário definir um valor de DTW para que seja uma referência. Para tal, foi feita a DTW com vários sinais semelhantes com SNR de 35 dB e quantização de 16 bits, caracterizando um conjunto de *frames* sem novidade. Após isso, foi gerado um histograma. Foi identificado que se trata de uma distribuição normal, já que o ruído adicionado é um ruído Gaussiano, então, para se ter uma precisão de 99,7% utilizou-se o valor do limiar como:

$$LimiarDTW = \mu + 3 * \sigma, \tag{3.2}$$

em que μ é a média e σ é o desvio padrão da distribuição.

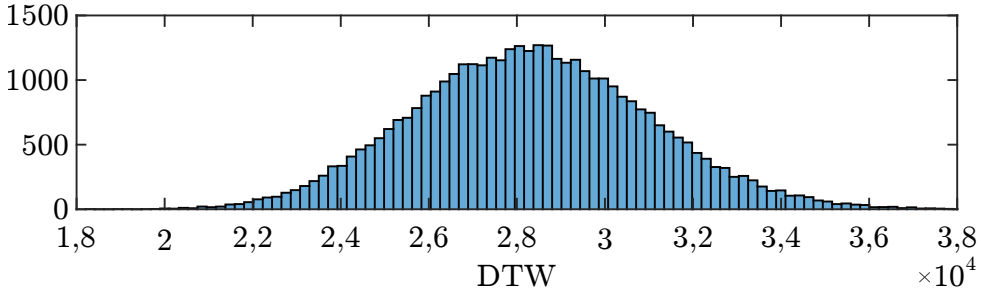


Figura 15: Histograma dos valores DTW

Após os testes de simulação para o limiar, este foi definido como 36000.

4 RESULTADOS

Com o intuito de comprovar o funcionamento do protótipo, foram feitas simulações em vários ambientes durante os testes. Conforme as simulações avançam, estas se mostram cada vez mais próximas de um ambiente real, incluindo testes em kit de desenvolvimento FPGA com leitura externa ao computador.

4.1 Descrição do Hardware no Quartus e Consumo de Recursos

Esta seção irá mostrar e comentar o desenvolvimento do protótipo no Quartus.

Foi utilizada a linguagem de descrição de hardware Verilog para desenvolvimento do projeto no framework Quartus Lite Versão 20.3. O kit de desenvolvimento utilizado foi o DE10NANO e o consumo de lógica dele é mostrado na Figura 16. Podemos observar que uma grande parte do hardware está sendo utilizada no Signal Tap (analisador lógico presente no Quartus), ou seja, o projeto é bem menor do que parece, como demonstrado na Tabela 2.

Entity:Instance	Combinational ALUTs	Dedicated Logic Registers	Block Memory Bits	DSP Blocks	Pins	Virtual Pins
Cyclone V: 5CSEBA6U23I7						
top_level	4800 (124)	4300 (8)	4383977	3	41	0
FIFO_128_16:FIFO_128_16_inst	51 (0)	33 (0)	2048	0	0	0
atrasador_20:atrasador_20_inst	26 (0)	16 (0)	320	0	0	0
sld_hub:auto_hub	90 (1)	90 (0)	0	0	0	0
sld_signaltap:auto_signaltap_0	958 (2)	3507 (508)	4161536	0	0	0
berco_DTW:berco_DTW_inst	376 (5)	218 (160)	206921	2	0	0
berco_FiltroPB_Zc:berco_FiltroPB_Zc_i...	2952 (8)	245 (192)	12640	1	0	0
interface_AD:interface_AD_inst	198 (0)	161 (0)	512	0	0	0
maq_estados:maq_estados_inst	5 (5)	4 (4)	0	0	0	0
maq_mais_cont_div_frame:maq_mais...	20 (0)	18 (0)	0	0	0	0
pll_entrada:pll_entrada_inst	0 (0)	0 (0)	0	0	0	0

Figura 16: Consumo de lógica.

Tabela 2 – Consumo de recursos por módulo (sem Signal Tap)

Módulo	**ALUTs**	**Regs**	**Memória (bits)**	**DSPs**
FIFO_128_16	51	33	2 048	0
atrasador_20	26	16	320	0
berco_DTW	376	218	206 921	2
berco_FiltroPB_Zc	2 952	245	12 640	1
interface_AD	198	161	512	0
maq_estados	5	4	0	0
maq_mais_cont_div_frame	20	18	0	0
pll_entrada	0	0	0	0
Total sem Signal Tap	**3 628**	**695**	**222 441**	**3**

O kit DE10-Nano utiliza o FPGA Intel Cyclone V SE 5CSEBA6U23I7, cujas capacidades totais são (Intel Corporation, 2019):

- 110000 elementos de lógica programável (LEs),
- Aproximadamente 220000 ALUTs,
- 5 570 000 bits de memória interna (5,57Mbits),
- 112 blocos DSP.

Considerando-se os valores base do kit, o consumo relativo do projeto é:

$$\text{ALUTs: } \frac{3\,628}{220\,000} \times 100\% \approx 1{,}65\%$$
$$\text{Registradores: } \frac{695}{110\,000} \times 100\% \approx 0{,}63\%$$
$$\text{Memória (bits): } \frac{222\,441}{5\,570\,000} \times 100\% \approx 4{,}00\%$$
$$\text{DSPs: } \frac{3}{112} \times 100\% \approx 2{,}68\%$$

Estas métricas indicam que o projeto consome apenas uma fração dos recursos disponíveis no FPGA, garantindo margem para expansão e complementação funcional.

A principal parte do RTL Viewer do projeto no Quartus é:

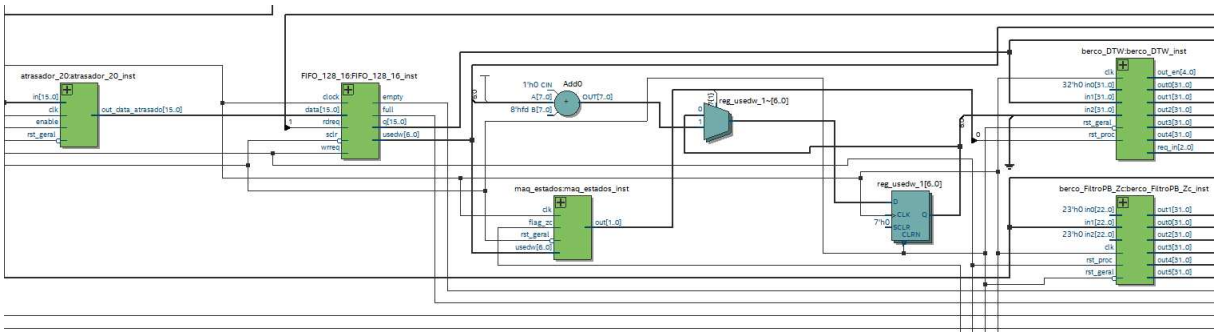


Figura 17: Parte principal do RTL Viewer do Projeto.

4.2 Resultados Simulação MatLab® x ModelSim®

Esta seção irá mostrar e comentar os resultados feitos com o MatLab® e comparar aos feitos com simulação funcional de hardware no ModelSim®.

As figuras de simulação no Matlab são mostradas em cores, enquanto as figuras do ModelSim foram extraídas em preto e branco. As figuras do ModelSim seguem o mesmo padrão e mostram resultados das simulações funcionais de hardware realizada. Cada figura é dividida em linhas, sendo elas:

- Na primeira linha (superior) está o *clock* do sistema.
- Na segunda está o *reset* do sistema.
- Na terceira linha está o sinal que está sendo analisado, sendo esse o mesmo analisado nas Figuras coloridas, que são de MatLab®.
- Na quarta linha estão os pulsos de *flag* ZC, dados pelo processador SAPHO 1, os quais ocorrem no momento do cruzamento descendente do sinal pelo zero. Tais pulsos dividem o sinal em *frames*. Foram inseridas linhas tracejadas para melhor visualização dos instantes dos cruzamentos. Além disso, uma numeração dos *frames* foi inserida sobre o sinal para facilitar a identificação de cada um.
- Na quinta linha estão os valores de DTW calculados em cada comparação de *frames* pelo processador SAPHO 2.
- Na sexta linha mostra sempre o mesmo valor, pois é o valor de limiar.
- Na sétima linha estão os valores de estimação de frequência realizada pelo processador SAPHO 1. É importante comentar que, por padrão, o ModelSim® só mostra valores inteiros arredondados para o inteiro inferior mais próximo. Sendo assim, mesmo que os valores de frequência estimada dentro do processador SAPHO 1 sejam em ponto flutuante, digamos 59,999 Hz ou 60,0001 Hz, os valores mostrados serão de 59 Hz e 60 Hz, respectivamente.
- Na oitava e última linha mostra a *flag* de novidade que é pulsada naqueles *frames* que contêm novidades.

É importante observar que, diferente dos testes em Matlab®, como o processador funciona em tempo real, não há informação prévia dos valores de DTW antes de um novo *frame* terminar. Por isso, a detecção de novidades só acontece após o fim de um novo *frame*, ou seja, durante o próximo *frame*. Por isso, a *flag* de novidade só sinaliza as detecções no *frame* seguinte ao qual foi detectada uma novidade. Para melhor entendimento, foi inserida uma numeração de *frames* também na última linha. Pela razão descrita acima, esta numeração sofre o atraso de um *frame* em relação à numeração de *frames* mostrada sobre o sinal.

4.2.1 Sinal com distúrbio transitório

A Figura 18 mostra em sua parte superior um gráfico do sinal em análise. O sinal foi segmentado em *frames* a cada cruzamento por zero descendente, resultando em 10 *frames*, os quais podem ser visualizados pelas linhas tracejadas que os separam. O sinal possui apenas o componente fundamental (60 Hz) e um transitório de alta frequência localizado no *frame* 6.

Utilizando a comparação dos *frames*, pode-se chegar aos *frames* de novidade destacados em rosa. Inicialmente, todos os valores de saída do sistema detector são colocados em zero, visto que ainda não há informação de um *frame* inteiro para estimar informações e há um pulso de Reset. O primeiro *frame* sempre é considerado como *frame* de novidade, visto que anteriormente não havia outro *frame* a ser comparado. Pode-se ver que o *frame* 1 não possui um ciclo completo do sinal, pois a defasagem inicial do sinal não é nula. Sendo assim, ele é diferente do *frame* 2, o qual possui um ciclo completo. Dessa forma, o *frame* 2 passa a ser o *frame* de novidade. Como os *frames* 3, 4 e 5 são iguais ao 2, eles não detectam novidades. Já o *frame* 6, possui o distúrbio transitório, portanto, é um *frame* de novidade quando comparado ao 2. Isso faz com que o *frame* 6 seja a referência. Quando o *frame* 7 é comparado ao *frame* 6, novamente detecta-se uma novidade e assim o *frame* 7 torna-se o *frame* de referência. Após o *frame* 7, todos os *frames* subsequentes, 8, 9 e 10 são iguais. Portanto, não há mais novidades.

Na parte inferior da Figura 18 pode-se ver os valores de DTW de cada comparação de *frames*, juntamente com o limiar adotado. Pode-se notar que os valores de DTW que ultrapassaram o limiar ocorrem exatamente nos *frames* de novidade destacados na parte superior, mostrando que a DTW foi capaz de identificar corretamente os *frames* de novidade.

Pode-se ver que os valores calculados no FPGA mostrados na Figura 19 foram exatamente os mesmos da simulação em Matlab® mostrada na Figura 18 e que a *flag* de novidade foi ativada exatamente nos *frames* de novidade esperados.

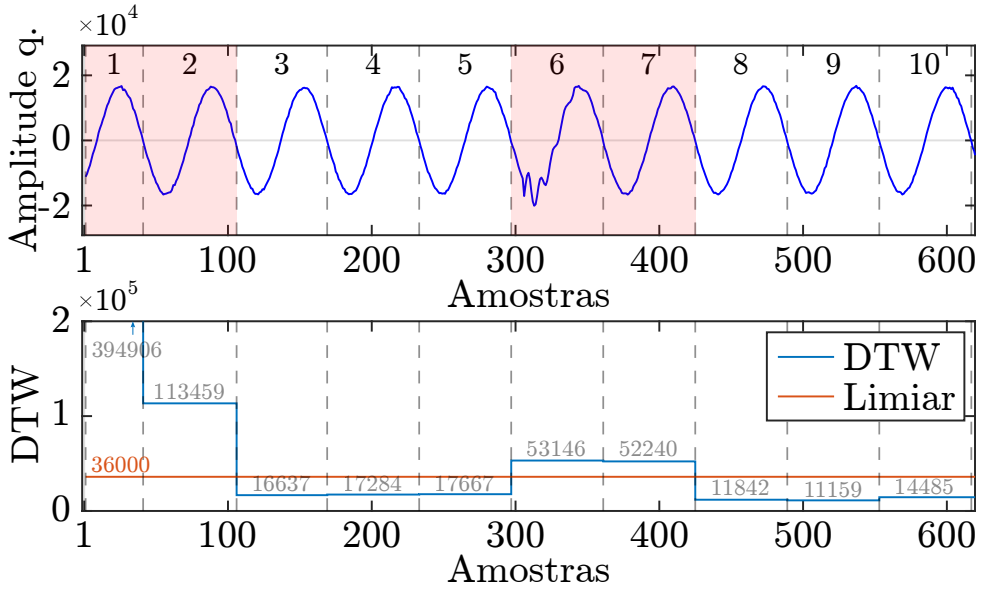


Figura 18: Teste em Matlab® de detecção com transitório.

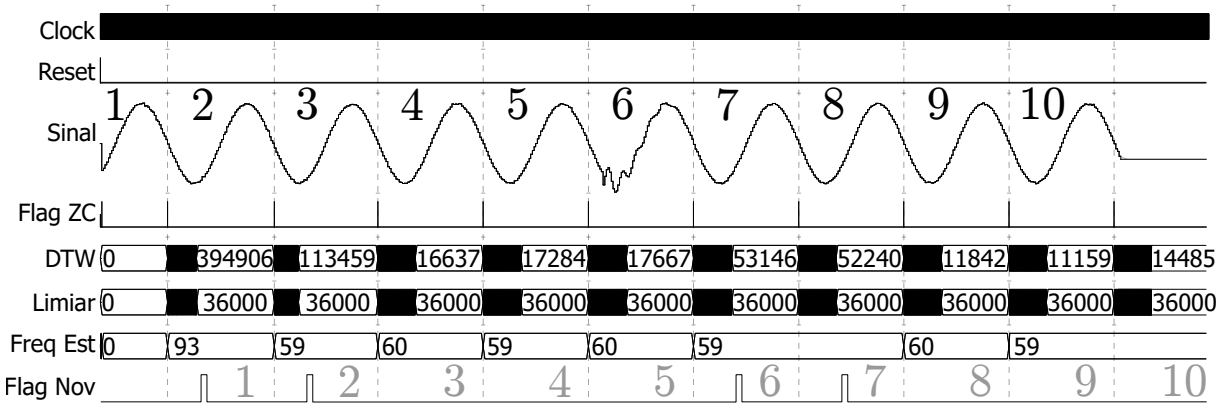


Figura 19: Teste ModelSim® de detecção com transitório.

4.2.2 Sinal com distúrbio *sag*

A Figura 20 mostra um sinal contendo um *sag* que se inicia no meio do *frame* 6 e finaliza no meio do *frame* 10. Pelo mesmo raciocínio utilizado no caso anterior, conclui-se que os *frames* de novidade são os *frames* 1, 2, 6, 7, 10 e 11, conforme destacado na Figura 20. É importante observar que durante a ocorrência do distúrbio *sag*, o sinal se mantém estacionário, resultando na não detecção de novidades. Isso contribui para o armazenamento eficiente apenas das informações relevantes e evita guardar redundâncias. Assim, otimizando um sistema de reconstrução de sinal que pode ser implementado no protótipo.

A Figura 21 mostra o resultado da simulação funcional de hardware. Pode-se ver que os valores de DTW calculados estão corretos e que os *frames* de novidade também foram detectados corretamente.

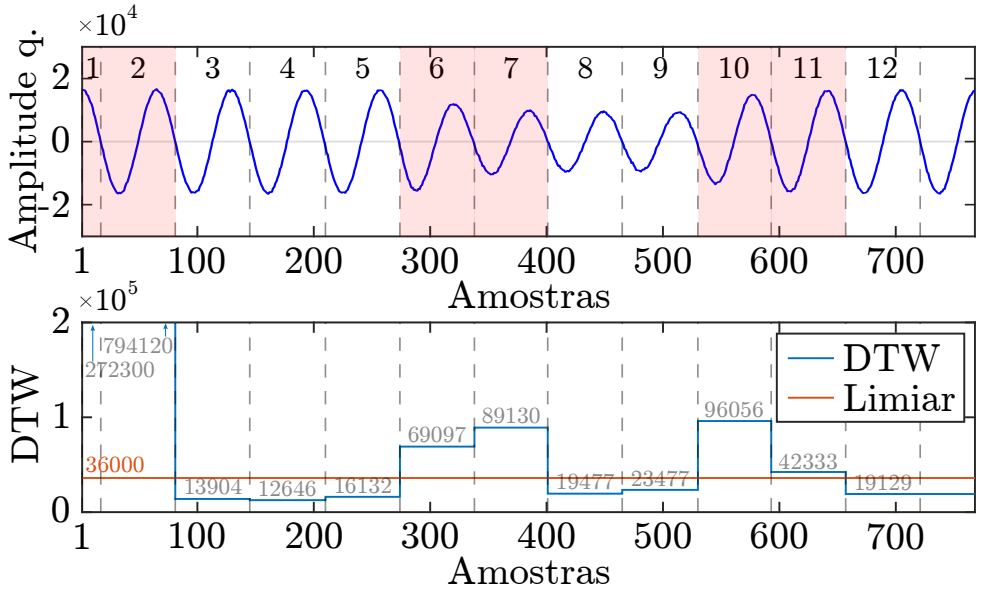


Figura 20: Teste em Matlab® de detecção com *sag*.

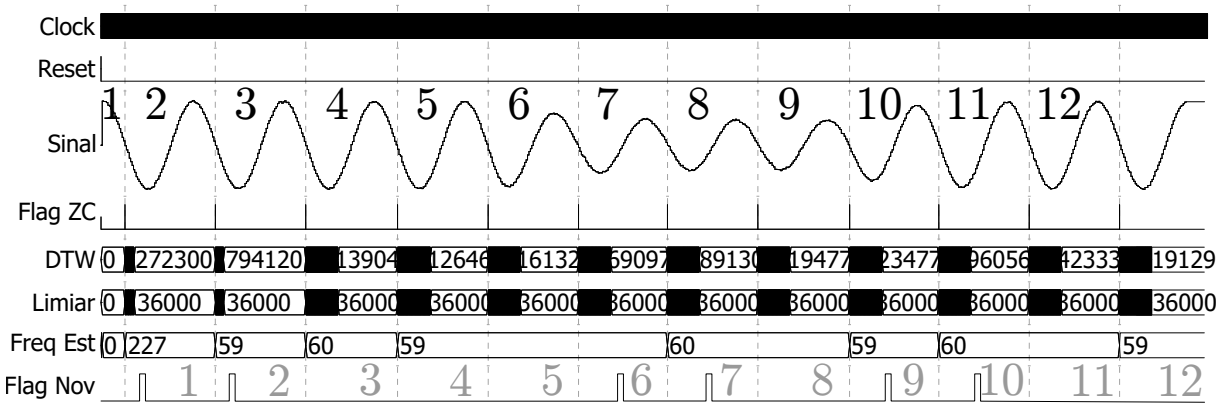


Figura 21: Teste no ModelSim® de detecção com *sag*.

4.2.3 Sinal com harmônico variante no tempo

A Figura 22 mostra um sinal contendo um conteúdo harmônico variante no tempo, o qual se inicia no meio do *frame* 5 e termina no meio do *frame* 9. Novidades são detectadas nos *frames* 1, 2, 5, 6, 9 e 10, conforme o esperado. A Figura 23 mostra o resultado da simulação funcional de hardware.

Como no caso anterior, este caso também mostra que, durante a ocorrência do distúrbio, mesmo com a presença do harmônico, o sinal é estacionário, sendo armazenados apenas os *frames* do início e fim do evento, não importando o tempo de duração, mesmo que haja variação na frequência. Além disso, fica evidente a utilidade do filtro passa-baixas presente no processador SAPHO 1 antes da detecção dos ZC, pois, caso não houvesse o filtro, o harmônico atrapalharia essas detecções de cruzamento.

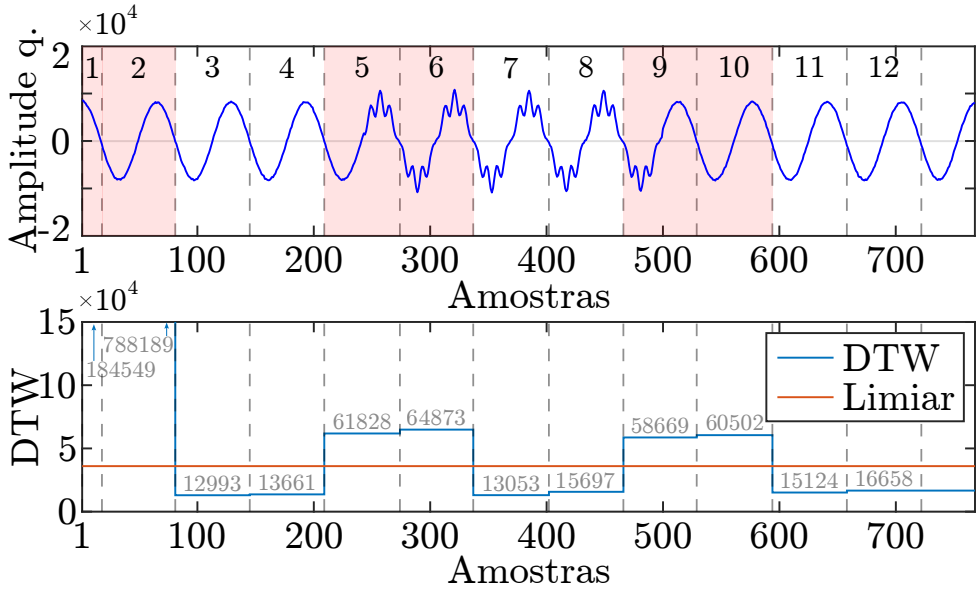


Figura 22: Teste em Matlab® de detecção com harmônico variante no tempo.

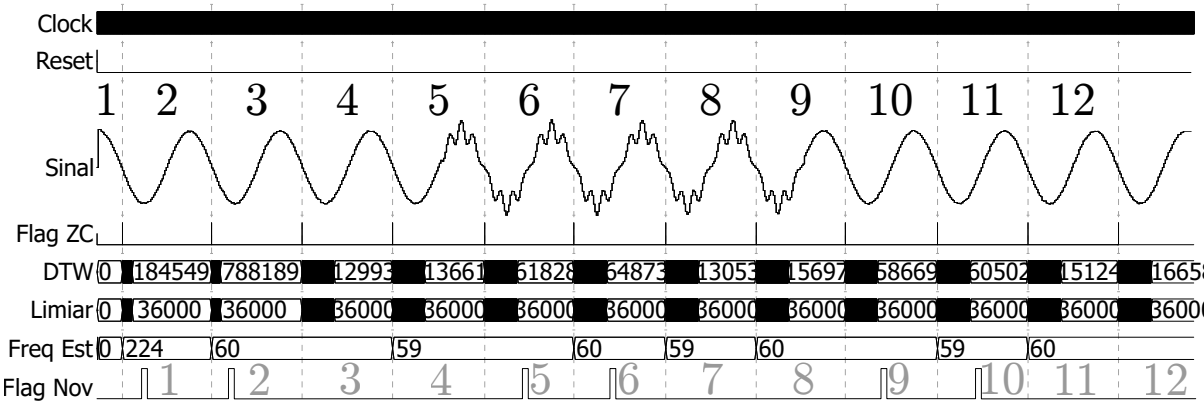


Figura 23: Teste no ModelSim® de detecção com harmônico variante no tempo.

4.2.4 Sinal com variação de frequência

A Figura 24 mostra o último caso analisado. Neste caso, é importante verificar que, mesmo que o sinal não seja estacionário, o formato dos segmentos do sinal dentro dos *frames* é sempre o mesmo, pois cada *frame* possui um ciclo inteiro. Este caso mostra uma das principais vantagens de se utilizar uma métrica de similaridade dinâmica como a DTW, pois mesmo que o *frame* 17 contenha um número bem menor de pontos que o *frame* 2 de referência, eles possuem o mesmo formato, resultando em uma DTW com valor abaixo do limiar. O mesmo ocorre para todos os *frames* após o segundo, conforme o esperado. O resultado da simulação funcional do ModelSim® é mostrado na Figura 25.

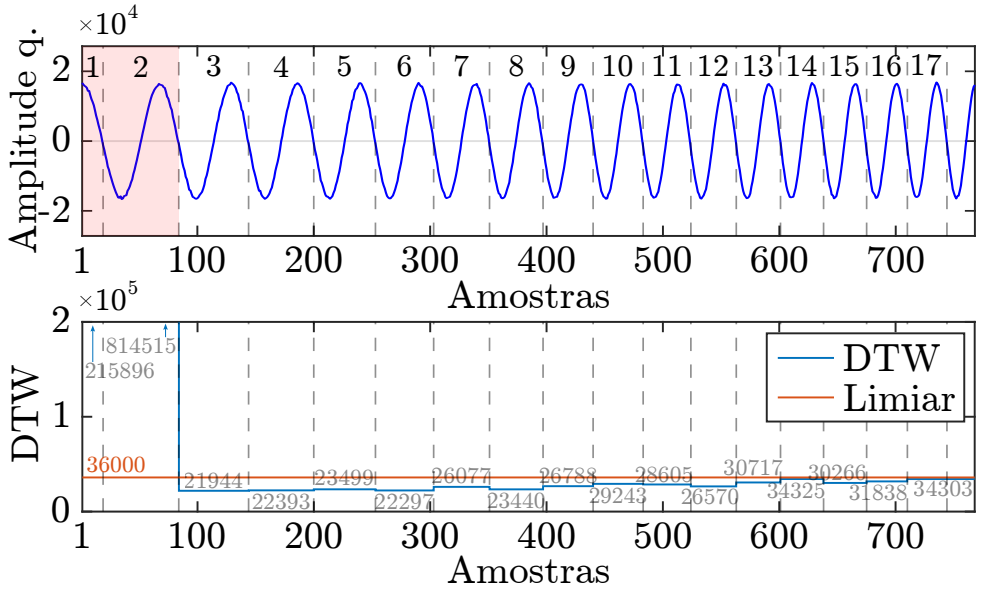


Figura 24: Teste em Matlab® com variação de frequência.

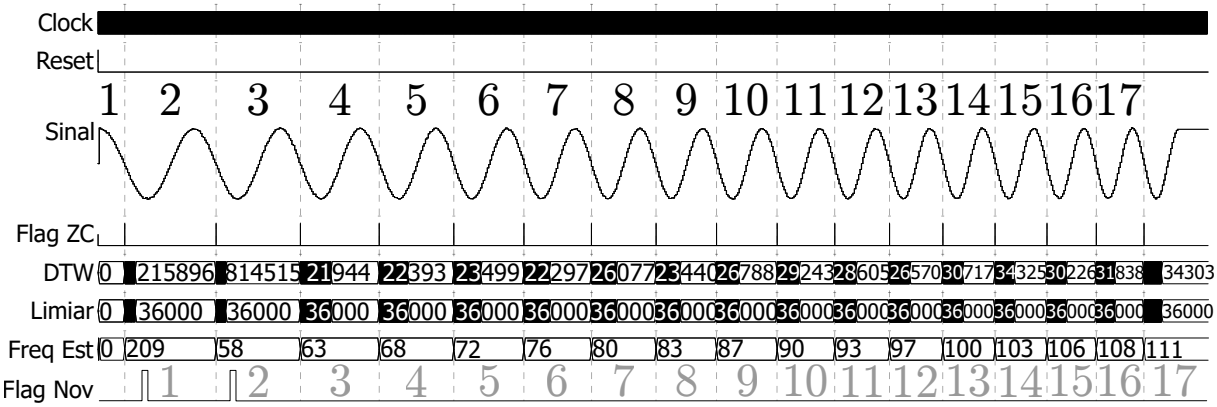


Figura 25: Teste no ModelSim® de detecção com variação linear de frequência.

4.2.5 Tempo de processamento na simulação funcional

A fim de obter uma estimativa do número total de ciclos de clock necessários para o processamento, foi feito um teste com o processador que demanda mais tempo, sendo este o SAPHO 2. A Figura 26 mostra resultados de simulação funcional no ModelSim considerando o sinal com harmônicos mostrado na Figura 23.



Figura 26: Teste no ModelSim® de tempo de processamento.

A partir da simulação foi comparado o tempo final de processamento com o inicial e levado em consideração que o *clock* escolhido para a simulação foi de 4 ns, chegando-se a uma estimativa do número de ciclos de *clock* máximo utilizado para a execução completa do processamento do sistema detector de novidades de 380 mil ciclos. Se considerarmos uma tolerância para o desvio de frequência máximo de 5 Hz acima da nominal, o que já é considerado muito superior aos valores reais, teremos o pior caso com os ciclos mais curtos possíveis com frequência fundamental de 65 Hz. O tempo de processamento disponível entre uma detecção e a próxima é de 1 *frame*. A duração de um *frame* contendo um ciclo inteiro, considerando a fundamental de 65 Hz, seria de aproximadamente 15,38 ms. Para

o FPGA ser capaz de executar 380 mil operações, a frequência de *clock* deveria ser de 24,70 MHz, o que é um valor factível, visto que em kits de desenvolvimento de entrada, as frequências de *clock* base tem valores a partir de 50 MHz. Para a implementação na placa, foi escolhida uma frequência superior a essa e que fosse múltipla da frequência de amostragem, por isso foi utilizada a frequência de 38,4 MHz.

4.3 Resultados Experimentais

Segue nesta seção a exposição e análise dos testes realizados com sinais reais em bancada utilizando o osciloscópio ou o Signal Tap do Quartus.

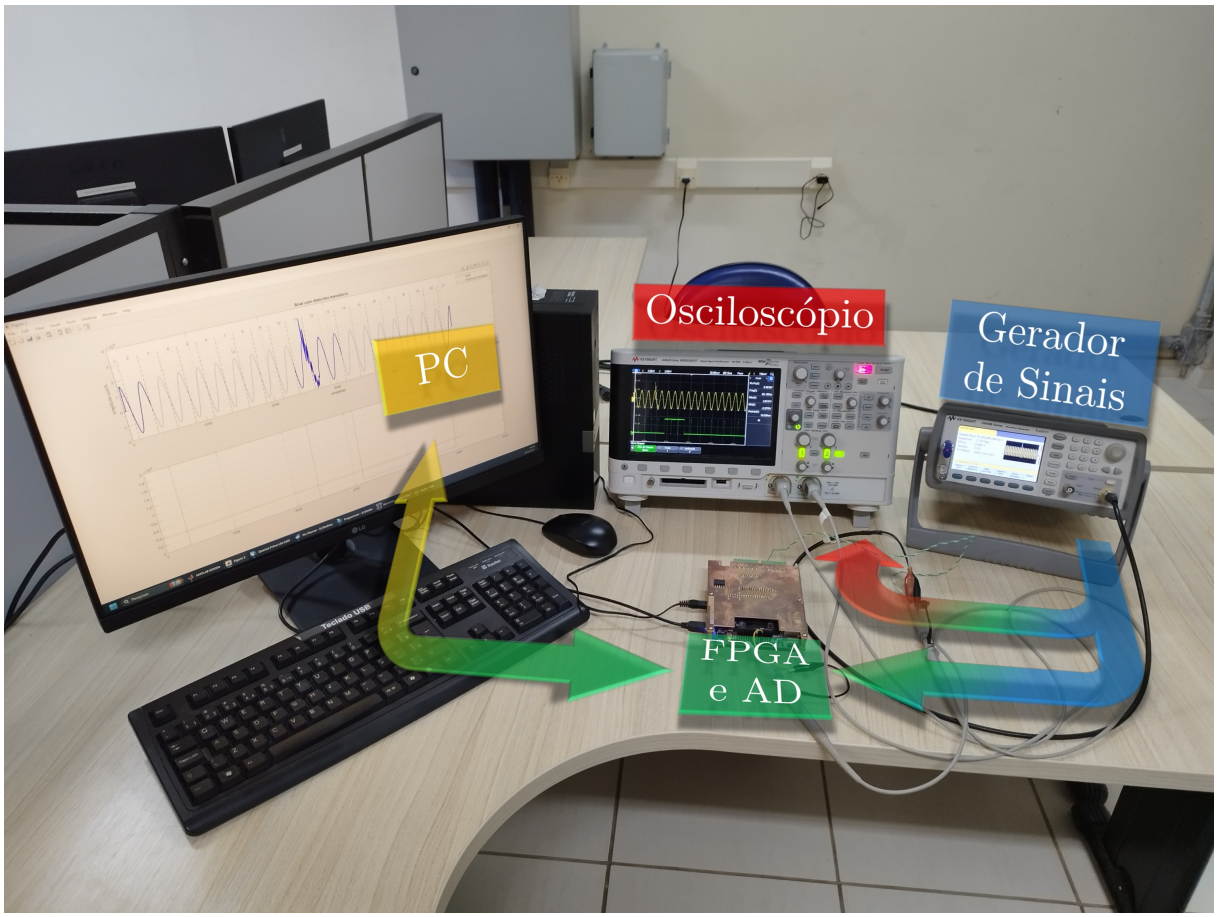


Figura 27: Bancada montada para captura de sinal no osciloscópio.

A Figura 27 mostra o *setup* de teste montado para os experimentos. Os equipamentos que compõem este *setup* são: um computador de mesa com um processador Intel Core i7 de 9ª geração, um osciloscópio KEYSIGHT modelo InfiniiVision MSOX3012T com largura máxima de banda de 100 MHz e taxa de amostragem de 5 GSa/s, um conversor AD7606 acoplado no kit de desenvolvimento FPGA DE10-Nano, e por fim, um gerador de sinais KEYSIGHT modelo 33500B.

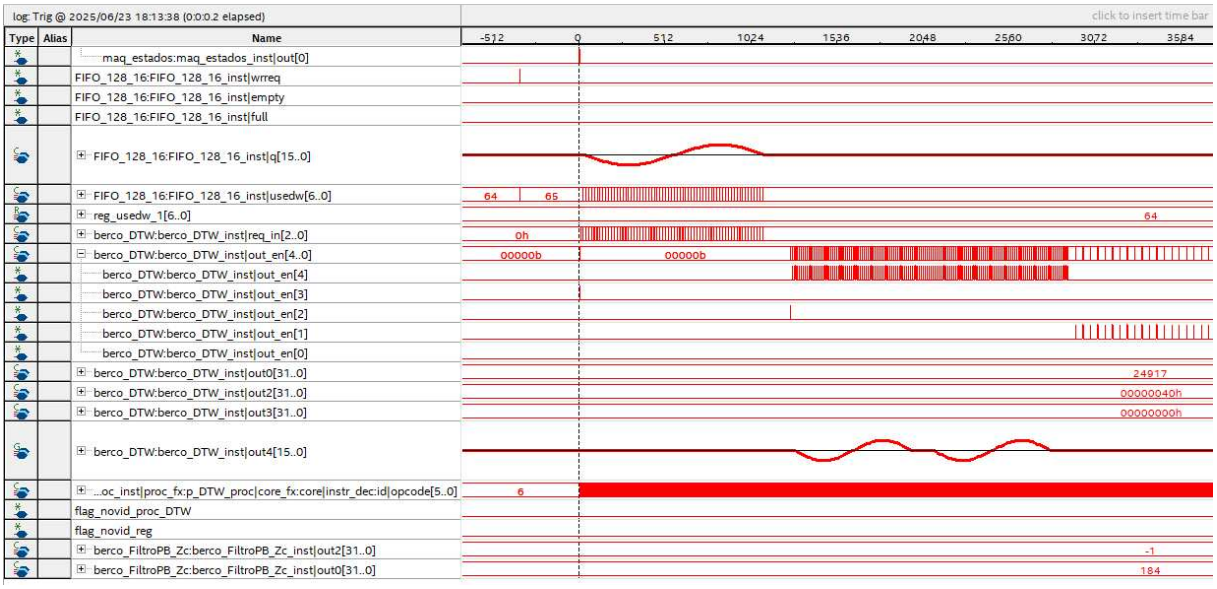


Figura 28: Sinal senoidal com ruído de fundo entrando no Signal Tap e sendo processado.

4.3.1 Resultados Signal Tap

Nesta subseção serão apresentados os resultados provenientes dos testes com um osciloscópio, um gerador de sinais e um kit FPGA na perspectiva do Signal Tap do Quartus.

4.3.1.1 Sinal senoidal sem distúrbios

Podemos observar na Figura 28 os sinais sendo processados pelo sistema e exibidos no Signal Tap. A quinta linha de cima para baixo mostra a saída da FIFO exibindo o *frame* atual sendo descarregado na entrada do processador DTW (SAPHO 2). Pode-se observar que o *frame* é composto de exatamente um ciclo da componente fundamental. O número de amostras por *frame* é exibido na linha logo abaixo, onde é mostrado o número 64 (amostras).

Na mesma figura, mais abaixo, na décima-oitava linha, são mostrado os *frames* a serem comparados dentro do processador DTW, o *frame* atual, descarregado da FIFO e o *frame* de referência anteriormente armazenado da última novidade detectada. O resultado da comparação é exibido na décima-quinta linha e, para este caso mostrado, vale 24917. Como o valor não extrapola o limiar de 36000, este *frame* analisado não é considerado de novidade. Neste caso, não são utilizadas todas as 75 posições pra cada vetor (máximo permitido pelo processador DTW), apenas 64 por vetor.

4.3.1.2 Sinal com Variação de Frequência

Na sequência, foi feito um teste de variação de frequência, para mostrar a robustez do protótipo a esse tipo de variação. Na Figura 29, foi aumentada a frequência do sinal no

gerador de funções a tal ponto que a componente fundamental possuía apenas 45 amostras por ciclo e ainda assim não foi detectada novidade. Podemos observar que claramente há um aumento de frequência pela comparação do número de pontos dos *frames* comparados, sendo o da direita o *frame* de referência na linha 18.

Em seguida, na Figura 30 a frequência é aumentada ainda mais, sendo o dobro da original, e ainda assim não é detectada a novidade. Finalmente, na Figura 31, a frequência atinge 132 Hz e é detectada como novidade, porque podemos ver que o *frame* de referência foi atualizado, não foi mostrado o momento exato de comparação e troca de referência pois se trata de um momento muito específico e difícil de ajustar o *trigger*.

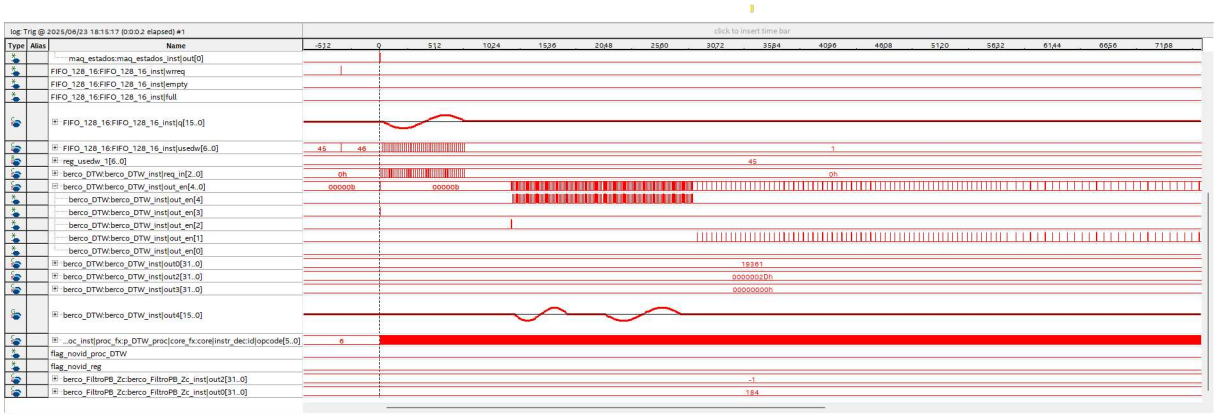


Figura 29: Signal Tap com os sinais senoidais começando a variar a frequência.

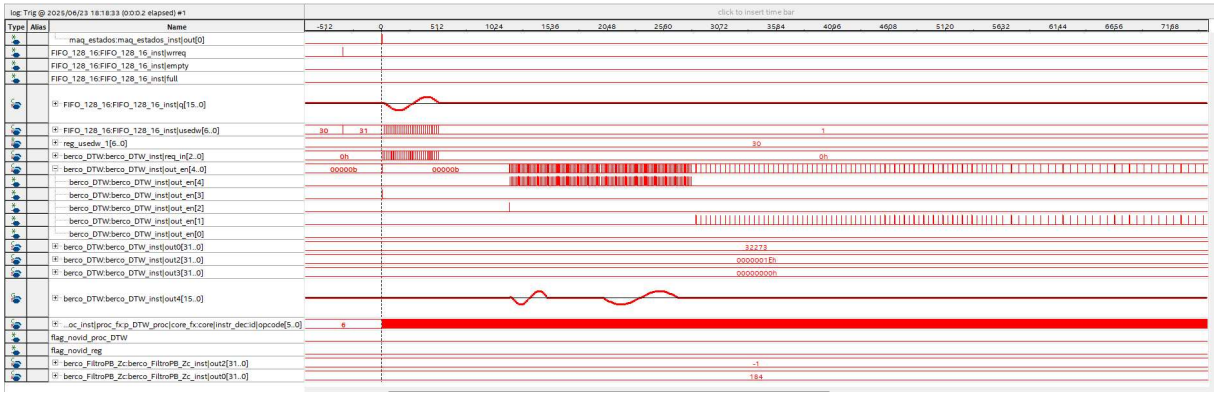


Figura 30: Signal Tap com os sinais senoidais com grande variação na frequência.

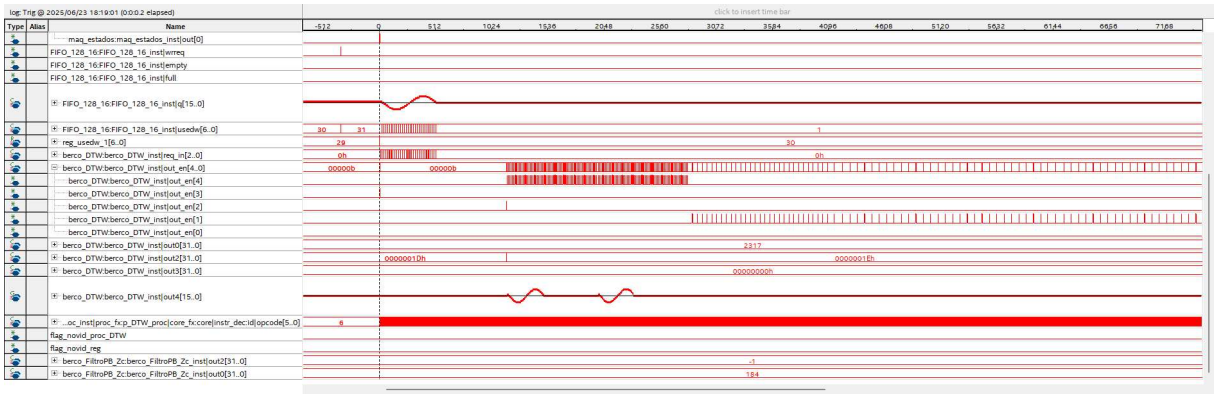


Figura 31: Signal Tap com os sinais senoidais com frequência acima do limiar onde não é detectada tal variação.

4.3.2 Tempo de processamento

A fim de estimar o número de ciclos de *clock* necessários para completar o processamento do processador SAPHO 2 (Processador DTW), foi adicionado um contador de ciclos de *clock* que é restado junto com o processador SAPHO 2. Pode-se observar na Figura 32 o contador chegando ao valor próximo a 302 mil ciclos de *clock* no final do processamento da DTW, indicado pela saída out_en[0]. O sinal utilizado neste teste foi um distúrbio harmônico com a fundamental em 60 Hz, o mesmo sinal mostrado na Figura 23.



Figura 32: Teste com detecção de harmônico para medição de ciclos de *clock*.

Outro teste para estimar o número de ciclos de *clock* pode ser visto na Figura 33. Neste teste, a frequência fundamental foi ajustada para 52 Hz resultando no número de pontos por ciclo de 74 amostras (o processador DTW é limitado a 75 amostras). A primeira linha da Figura 33 mostra o número de posições ocupadas pela FIFO, enquanto a última linha mostra o número de *clocks* que o processador SAPHO 2 gastou para processar o sinal, que foi próximo a 380 mil ciclos.

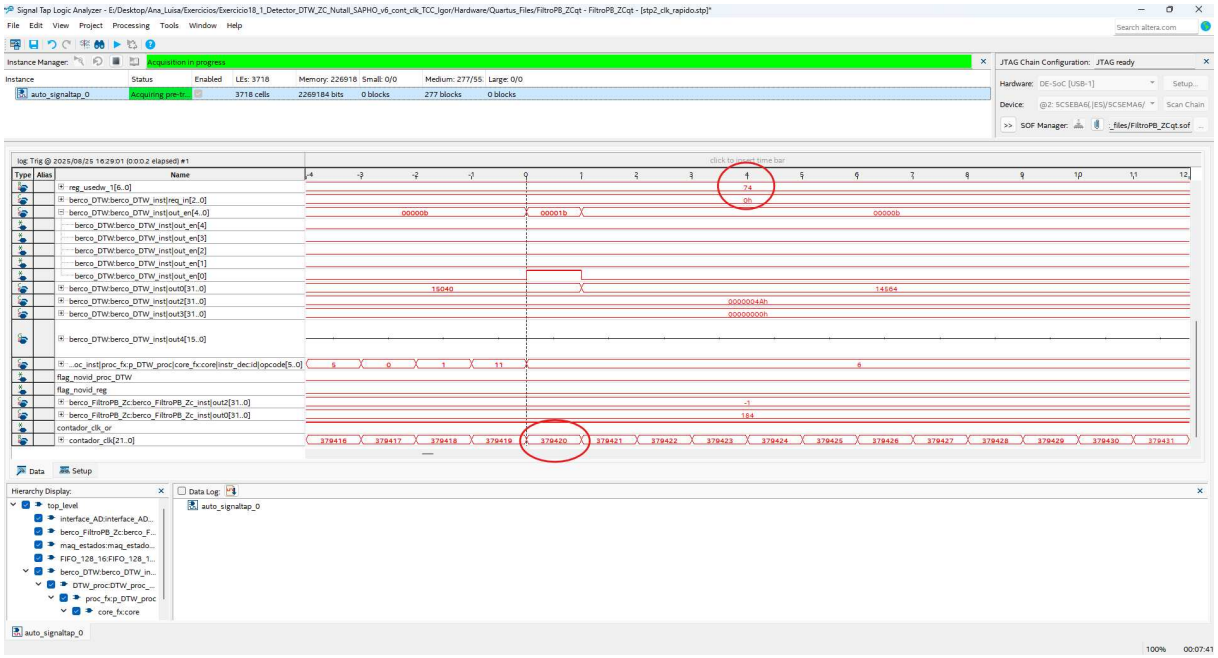


Figura 33: Teste com frequência off-nominal para medição de ciclos de *clock*.

Para estes testes de contagem de *clocks* foi escolhido o processador SAPHO 2 pois o mesmo é o que tem maior tempo de processamento, ou seja, é exigido o maior número de *clocks* em relação ao processador SAPHO 1 (Processador ZC).

Estes resultados de contagem de ciclos de *clock* mostram que a estimativa de ciclos de clock proveniente da simulação funcional no ModelSim mostrada na figura XX foi um resultado próximo ao real.

Além dos testes de contagem de *clocks* foi verificada a frequência máxima permitida para a implementação na placa de FPGA utilizada. Este resultado está presente na Figura 34.

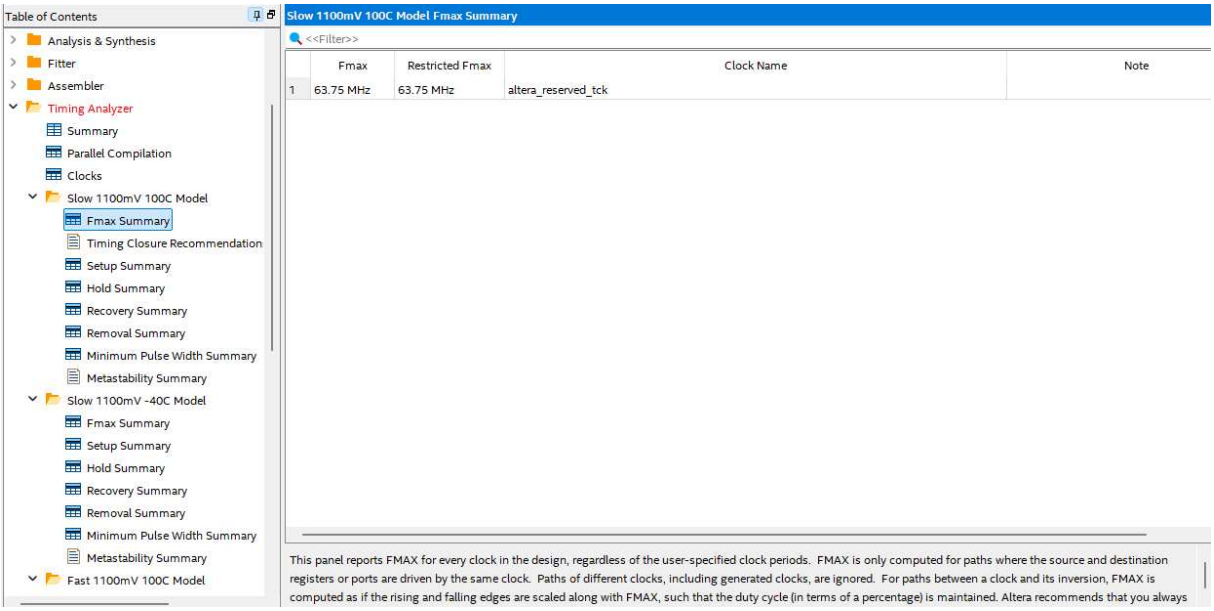


Figura 34: Relatório de compilação do Quartus mostrando a frequência máxima de operação para o chip selecionado.

Pode-se observar que a frequência utilizada de 38,4 MHz ainda está consideravelmente abaixo do limite máximo permitido de 63,75 MHz. Isso permite uma margem para aumentar a frequência de operação, possibilitando o aumento de número de pontos por *frame* analisado.

4.3.3 Resultados de Bancada no osciloscópio

Nesta subseção serão apresentados os resultados provenientes dos testes experimentais com visualização pelo osciloscópio.

4.3.3.1 Sinal com Distúrbio Transitório

A Figura 35 ilustra o funcionamento da DTW em tempo real para um sinal com um distúrbio transitório. No canal 1 do osciloscópio (em amarelo) é exibido o sinal analisado, enquanto no canal 2 (em verde) é mostrado o *flag* de novidades resultante do processo de detecção vindo de um pino de saída da FPGA. Foram evidenciados os divisores entre os *frames* localizados nos momentos de cruzamento pelo zero através de linhas tracejadas. Para uma melhor visualização das novidades, uma numeração dos *frames* foi inserida tanto no sinal de entrada do canal 1 quanto na *flag* de novidade do canal 2.

É importante observar que o canal 1 exibe o sinal de entrada antes do atrasador. Esse sinal é atrasado de 20 amostras, o que corresponde a um atraso pouco maior que $\frac{1}{4}$ de ciclo. Além disso, cada *frame* só é descarregado no processador SAPHO 2 após o seu término. Depois do cálculo da DTW, a *flag* de novidades é registrada no momento do próximo cruzamento pelo zero. Por estas razões, os resultados das detecções ficam

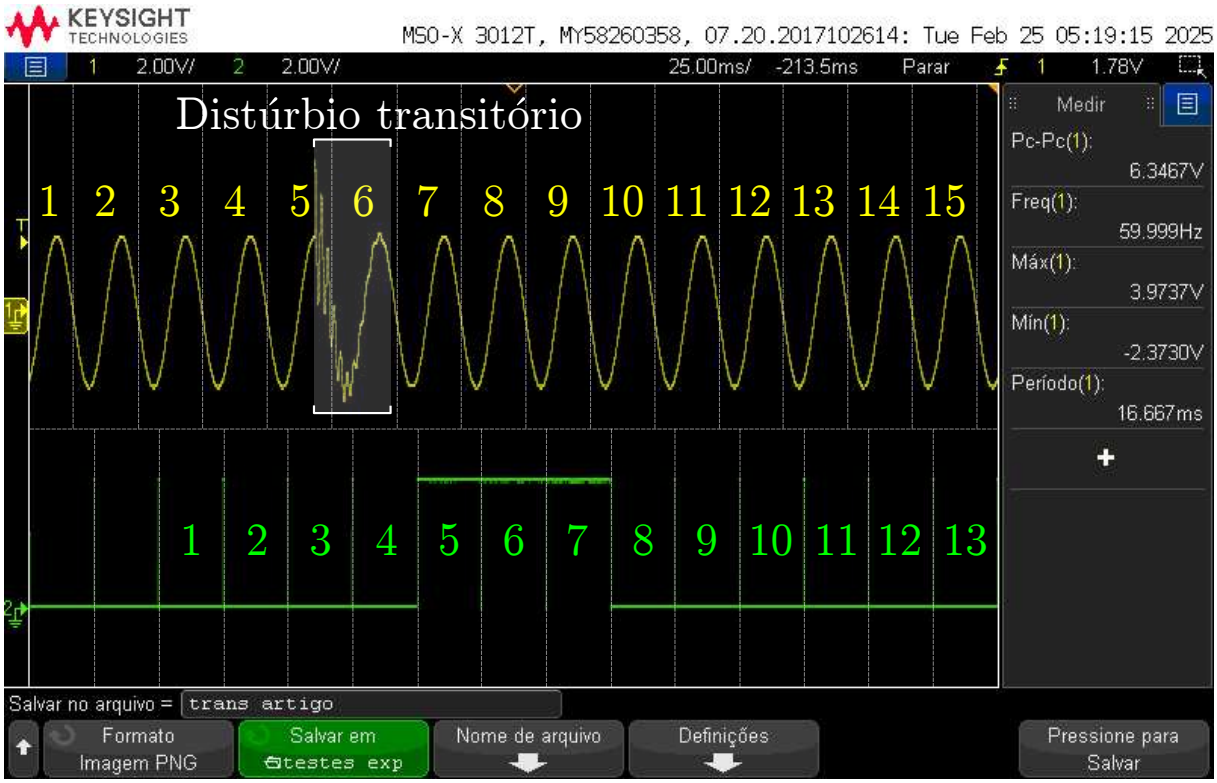


Figura 35: Teste com detecção de transitório.

atrasados de aproximadamente 2 ciclos e $\frac{1}{4}$ em relação ao sinal de entrada. Porém, isso em nada compromete a detecção, que ocorre em tempo real.

O distúrbio transitório inicia-se no final do *frame* 5 e termina no final do *frame* 6. O detector realiza a comparação *frame* a *frame*. Quando chega ao *frame* 5 e o compara com o 4, o sistema detecta uma novidade e sinaliza que o *frame* 5 é de novidade, tornando-o como novo *frame* de referência. Ao passar para o *frame* 6 ocorre mais uma novidade, pois este é diferente daquele. Então, o *frame* 6 é a nova referência. Ao passar para o *frame* 7, no qual o distúrbio já não existe, o detector percebe a diferença com o *frame* 6 e novamente sinaliza uma detecção. Após o *frame* 7 não há mais novidades a serem detectadas, pois o sinal voltou ao estado permanente. Observando-se o resultado de detecção, é possível notar que o detector identificou corretamente os *frames* de novidade.

4.3.3.2 Sinal com Distúrbio *Sag*

A Figura 36 mostra o resultado em tempo real de um sinal contendo um *sag* que tem início no meio do *frame* 4 e término no meio do *frame* 10. Os *frames* que apresentam novidades já estão destacados na mesma figura. Seguindo o raciocínio adotado no caso anterior, conclui-se que os *frames* de novidade são os 4, 5, 10 e 11, conforme corretamente identificados pelo detector. É importante frisar que, durante o distúrbio *sag*, o sinal permanece estacionário, o que impede a detecção de novidades. Esse comportamento ajuda a garantir o armazenamento eficiente, registrando apenas as informações essenciais

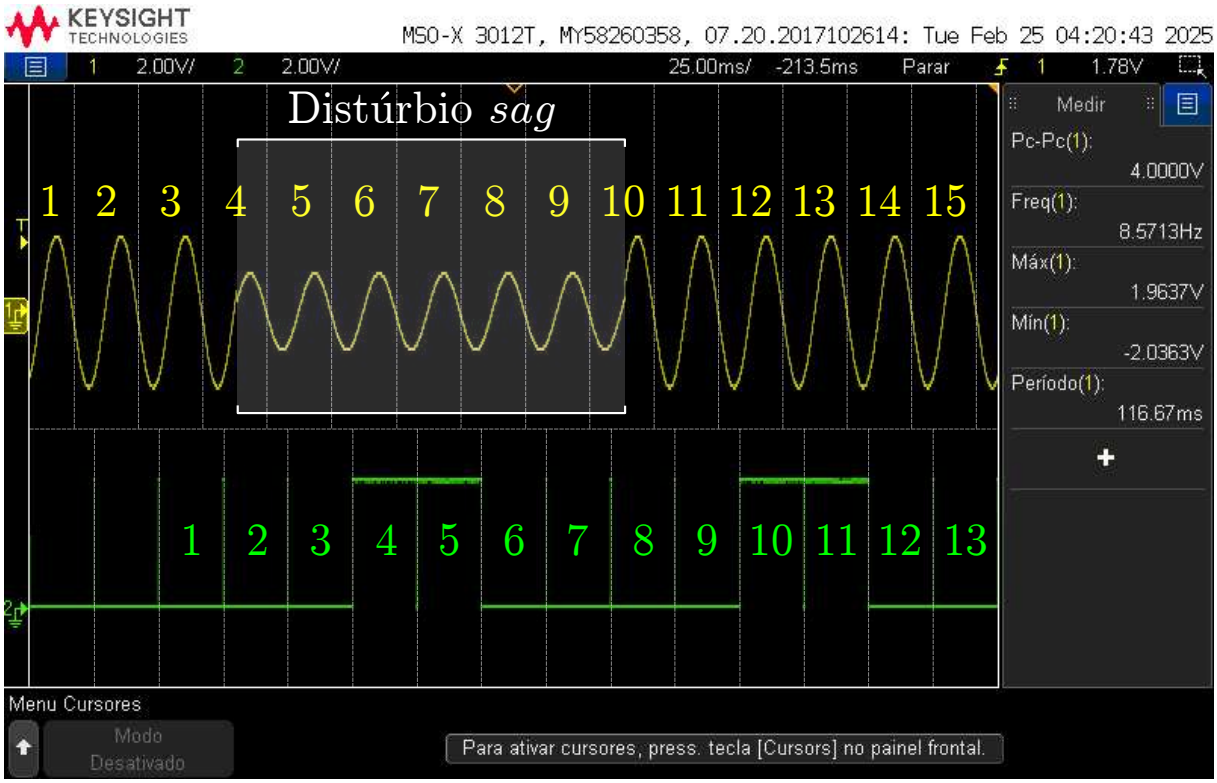


Figura 36: Teste de detecção com *sag*.

e evitando redundâncias.

4.3.3.3 Sinal com Distúrbio Harmônico Variante no Tempo

Este caso explora a capacidade do detector ser imune à presença de conteúdo harmônico que prejudicaria a detecção do cruzamento por zero, caso não houvesse um estágio de filtragem passa-baixas. Isso evidencia a robustez do sistema às harmônicas e ruído.

Como as harmônicas iniciam-se durante o *frame* 4 e terminam durante o *frame* 9, os *frames* de novidade devem ser: 4, 5, 9 e 10. Pode-se ver que o detector identificou corretamente os *frames* de novidade, como mostrado na Figura 37.

4.3.3.4 Sinal com Variação de Frequência

Este teste representa o caso principal para o detector, pois ele não deve detectar quando ocorre apenas variação de frequência fundamental. Durante o *frame* 5 inicia-se a variação linear de frequência partindo de 60 Hz e atingindo 120 Hz durante o *frame* 19. Propositalmente, foi inserido um distúrbio *spike* no *frame* 1 antes de começar a variação de frequência, de modo a comprovar a sensibilidade da detecção, visto que este distúrbio altera pouco a energia do *frame*, uma vez que o mesmo é muito curto. Sendo assim, os *frames* de novidade devem ser os *frames* 1 e 2.

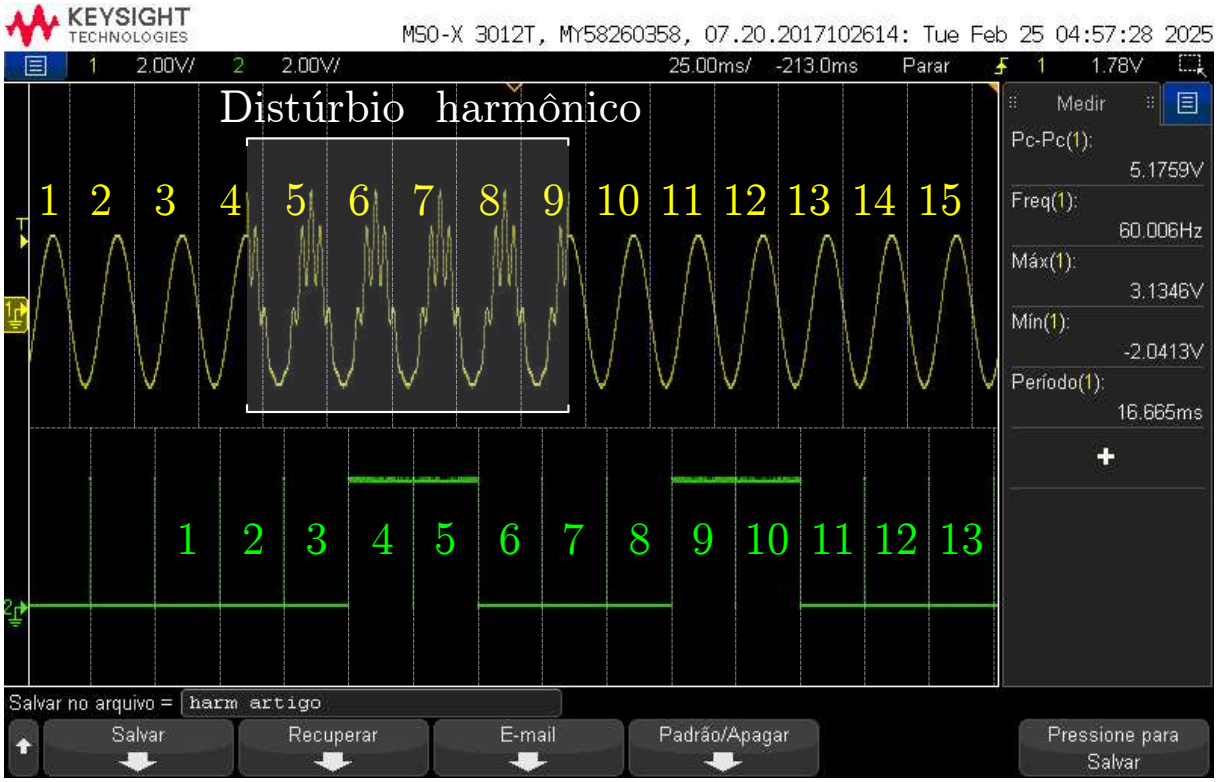


Figura 37: Teste de detecção com harmônico variante no tempo

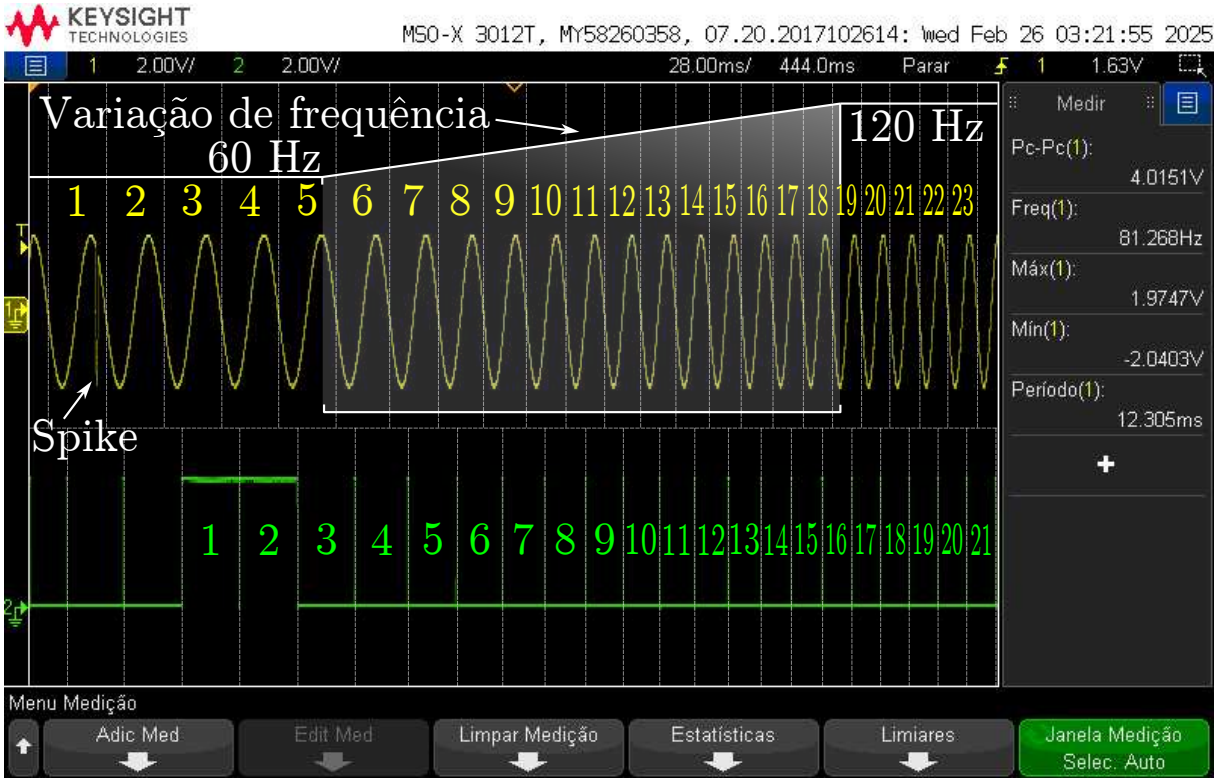


Figura 38: Teste de detecção com variação de frequência.

Ao mesmo tempo que o detector é sensível aos distúrbios que se desejam detectar, o mesmo é imune às variações de frequência, conforme mostrado na Figura 38. Como temos o *frame* de novidade, basta a frequência estimada de cada ciclo para a reconstrução do sinal. Isso é uma vantagem do método, pois o trecho com variação de frequência pode ser reconstruído sem a necessidade de armazenar todas as amostras. O método de reconstrução não faz parte do escopo deste trabalho, mas tem como base os requisitos mostrados em SILVA, (2016).

5 CONCLUSÃO

Esta monografia apresentou um protótipo funcional de um detector de novidades para aplicações em sinais de Qualidade de Energia Elétrica que utiliza a métrica DTW e foi desenvolvido em plataforma FPGA utilizando o kit de desenvolvimento DE10NANO, além disso, traz um estudo da contextualização do tema, motivação do trabalho e os objetivos que foram almejados.

Pode-se concluir que o método proposto mostrou-se adequado para a aplicação desejada, apresentando desempenho consistente mesmo em ambientes com ruído e harmônicos devido à aplicação do filtro FIR de fase linear Nutall e ao método de secção de *frames* por cruzamento por zero. Sua capacidade de detectar novidades nos distúrbios elétricos, ao mesmo tempo em que evita falsos positivos associados a flutuações na frequência, evidencia a robustez e a confiabilidade da abordagem adotada.

Os resultados apresentados evidenciam o processo de desenvolvimento e melhora do protótipo até que chegasse na versão final, que foi utilizada nos testes de bancada. Mesmo com limites de Hardware e limites temporais, o protótipo opera da maneira esperada e detecta as novidades com eficiência. Assim, podemos concluir que é satisfatório para uma aplicação em tempo real.

Podem ser apontadas como limitações do trabalho:

- A amplitude do sinal tem que ser quantizada para 16 bits, já que o limiar de detecção de novidade foi definido para essa quantização fixa. Um possível tratamento para esta limitação seria um estágio de normalização dos sinais antes do processamento.
- Ainda não armazena todos os *frames* detectados, apenas o *frame* de referência, que, quando atualizado, é perdido. Uma melhoria nesse sentido necessitaria de um dispositivo de armazenamento em massa externo ao processador.

Trabalhos futuros que podem ser desenvolvidos são:

- A otimização do cálculo da DTW para que não necessite de guardar a matriz inteira e seja menos custoso computacionalmente, logo, seria mais econômico e mais adequado para aplicações embarcadas.
- O aprimoramento dos testes, para que seja viável a concepção de um produto com as funcionalidades propostas.
- O armazenamento e reconstrução do sinal incorporando estampas de tempo, assim, adicionando mais uma função ao protótipo e o deixando mais completo para se adequar ao mercado.

APÊNDICE A – Códigos em Verilog desenvolvidos no trabalho

```
1  module top_level(
2              //PLL:
3              input clk_in, botao_rst_geral_n,
4              output wire clk_out, locked_sig, clk_50M,
5
6              //ADC
7             input  signed [15:0]  adc_db_i,// ADC parallel data bus
8             input  adc_busy_i,            // ADC BUSY signal
9             output [2:0] adc_os_o,        // ADC OVERSAMPLING signals
10            output adc_range_o,  // ADC RANGE signal
11            output adc_cs_n_o,            // ADC CS signal
12            output adc_rd_n_o,            // ADC RD signal
13            output adc_reset_o,  // ADC RESET signal
14            output adc_stdby_o,  // ADC STAND BY signal
15            output adc_convst_o,          // ADC CONVST signal
16            output adc_convst_o_B,
17            output AD_parserbyte,
18            output AD_refselect,
19
20             //OR GERAL:
21             output wire or_geral,
22
23             output wire flag_novid_proc_DTW,
24             output reg flag_novid_reg,
25
26             output wire flag_zc,
27             output wire flag_div_frame, //flag que indica divisor do
                  frame mais largo
28             output wire flag_div_frame_or_flag_novid,
29             output wire flag_novid_final
30             );
31
32 assign or_geral = |(out0_ZC|out1_ZC|out2_ZC|out3_ZC|out4_ZC|out5_ZC|
      out0_DTW|out1_DTW|out2_DTW|out3_DTW|out4_DTW|empty|full|q|usedw|
      req_in_DTW|req_in_DTW|out_en_DTW|clk|data|data_out_FIFO1|
      data_out_FIFO2|data_out_FIFO3|data_out_FIFO4|data_out_FIFO5|
      data_out_FIFO6|data_out_FIFO7|data_out_FIFO8|valid_FIFO_1|
      valid_FIFO_2|valid_FIFO_3|valid_FIFO_4|valid_FIFO_5|valid_FIFO_6|
      valid_FIFO_7|valid_FIFO_8);
33
34
35
36 wire [2:0] req_in_DTW;
37 wire [4:0] out_en_DTW;
38 wire signed [31:0] in1_proc_DTW = q;
```

```
39
40 //PLL
41 wire clk;
42 wire clr_n;
43 assign clk_out = clk;
44 wire rst_geral = !clr_n;
45
46 pll_entrada pll_entrada_inst (
47     .refclk   (clk_in),   //  refclk.clk
48     .rst      (!botao_rst_geral_n),      //   reset.reset
49     .outclk_0 (clk), // outclk0.clk
50     .locked   (locked_sig)    //  locked.export
51 );
52 assign clr_n = botao_rst_geral_n & locked_sig;
53
54 assign clk_50M = clk_in; //so pra testa o clk de entrada 50MHz na saida
55
56 /////// CONTROLE DO ADC ///////
57 wire signed [15:0] data_out_FIFO1;
58 wire signed [15:0] data_out_FIFO2;
59 wire signed [15:0] data_out_FIFO3;
60 wire signed [15:0] data_out_FIFO4;
61 wire signed [15:0] data_out_FIFO5;
62 wire signed [15:0] data_out_FIFO6;
63 wire signed [15:0] data_out_FIFO7;
64 wire signed [15:0] data_out_FIFO8;
65
66 wire valid_FIFO_1;
67 wire valid_FIFO_2;
68 wire valid_FIFO_3;
69 wire valid_FIFO_4;
70 wire valid_FIFO_5;
71 wire valid_FIFO_6;
72 wire valid_FIFO_7;
73 wire valid_FIFO_8;
74
75 assign adc_convst_o_B = adc_convst_o;
76
77 interface_AD interface_AD_inst(
78         .clr_n(clr_n),
79         .clk(clk),
80         .adc_db_i(adc_db_i),    // ADC parallel data bus
81         .adc_busy_i(adc_busy_i),        // ADC BUSY signal
82         .adc_os_o(adc_os_o),    // ADC OVERSAMPLING signals
83         .adc_range_o(adc_range_o),      // ADC RANGE signal
84         .adc_cs_n_o(adc_cs_n_o),        // ADC CS signal
85         .adc_rd_n_o(adc_rd_n_o),    // ADC RD signal
```

```
86          .adc_reset_o(adc_reset_o),     // ADC RESET signal
87          .adc_stdby_o(adc_stdby_o),     // ADC STAND BY signal
88          .adc_convst_o(adc_convst_o),     // ADC CONVST signal
89          .AD_parserbyte(AD_parserbyte),
90          .AD_refselect(AD_refselect),
91          .get(1'b1),
92
93          //saidas das FIFOS e valids para bloco processamento:
94          .data_out_FIFO1(data_out_FIFO1),
95          .data_out_FIFO2(data_out_FIFO2),
96          .data_out_FIFO3(data_out_FIFO3),
97          .data_out_FIFO4(data_out_FIFO4),
98          .data_out_FIFO5(data_out_FIFO5),
99          .data_out_FIFO6(data_out_FIFO6),
100         .data_out_FIFO7(data_out_FIFO7),
101         .data_out_FIFO8(data_out_FIFO8),
102
103         .valid_out_FIFO1(valid_FIFO_1),
104         .valid_out_FIFO2(valid_FIFO_2),
105         .valid_out_FIFO3(valid_FIFO_3),
106         .valid_out_FIFO4(valid_FIFO_4),
107         .valid_out_FIFO5(valid_FIFO_5),
108         .valid_out_FIFO6(valid_FIFO_6),
109         .valid_out_FIFO7(valid_FIFO_7),
110         .valid_out_FIFO8(valid_FIFO_8)
111         );
112
113 wire signed [15:0] data = data_out_FIFO1;
114
115 //registrador do used_w - 1:
116 reg [6:0] reg_usedw_1;
117 wire signed [31:0] in2_proc_DTW = reg_usedw_1;
118 always @ (posedge clk or posedge rst_geral)
119 begin
120     if(rst_geral)
121     begin
122         reg_usedw_1 <= 7'd0;
123     end
124     else
125     begin
126         if(out_maq[1])
127             reg_usedw_1 <= usedw - 7'd1;
128     end
129 end
130
131 //Atrasador 20 amostras
132 wire signed[15:0] q, data_atrasado;
```

```
133
134 atrasador_20 atrasador_20_inst (
135             .clk(clk),
136             .enable(/*wrreq*//*rst_proc*/valid_FIFO_1),
137             .rst_geral(rst_geral),
138             .in(data_out_FIFO1),
139             .out_data_atrasado(data_atrasado)
140             );
141
142 //FIFO 128 posicoes de 16 bits
143 wire  empty, full;
144 wire [6:0] usedw;
145
146 FIFO_128_16     FIFO_128_16_inst (
147             .clock (clk),
148             .data (data_atrasado),
149             .rdreq (req_in_DTW[1]),
150             .sclr (rst_geral),
151             .wrreq (/*wrreq*/valid_FIFO_1),
152             .empty (empty),
153             .full (full),
154             .q (q),
155             .usedw (usedw)
156             );
157
158 // Processador Zero Crossing (com filtro Nutall):
159 wire signed [22:0] data_proc_ZC = data_out_FIFO1;
160 wire signed [31:0] out0_ZC, out1_ZC, out2_ZC, out3_ZC, out4_ZC, out5_ZC;
161 assign flag_zc = |out5_ZC;
162
163 berco_FiltroPB_Zc berco_FiltroPB_Zc_inst (
164             .clk(clk),
165             .rst_geral(rst_geral),
166             .rst_proc(/*rst_proc*/valid_FIFO_1),
167             .in0(23'b0),
168             .in1(data_proc_ZC),
169             .in2(23'b0),
170             .out0(out0_ZC),
171             .out1(out1_ZC),
172             .out2(out2_ZC),
173             .out3(out3_ZC),
174             .out4(out4_ZC),
175             .out5(out5_ZC)
176             );
177
178
179 //Maquina de estados:
```

```
180 wire [1:0] out_maq;
181
182 maq_estados maq_estados_inst(
183             .clk(clk),
184             .usedw(usedw),
185             .rst_geral(rst_geral),
186             .flag_zc(flag_zc),
187             .out(out_maq)
188             );
189
190
191 //Processador DTW:
192 wire signed [31:0] out0_DTW, out1_DTW, out2_DTW, out3_DTW, out4_DTW;
193
194 berco_DTW berco_DTW_inst (
195             .clk(clk),
196             .rst_geral(rst_geral),
197             .rst_proc(out_maq[0]),
198             .in0(32'b0),
199             .in1(in1_proc_DTW),
200             .in2(in2_proc_DTW),
201             .out0(out0_DTW),
202             .out1(out1_DTW),
203             .out2(out2_DTW),
204             .out3(out3_DTW),
205             .out4(out4_DTW),
206             .req_in(req_in_DTW),
207             .out_en(out_en_DTW)
208             );
209
210 assign flag_novid_proc_DTW = |out3_DTW;
211
212
213 always @(posedge clk or posedge rst_geral)
214 begin
215     if(rst_geral)
216     begin
217         flag_novid_reg <= 1'b0;
218     end
219
220     else if(flag_zc)
221     begin
222         flag_novid_reg <= flag_novid_proc_DTW;
223     end
224 end
225
226 maq_mais_cont_div_frame maq_mais_cont_div_frame_inst(
```

```
227             .clk(clk),
228             .rst_geral(rst_geral),
229             .flag_zc(flag_zc),
230             .flag_div_frame(flag_div_frame)
231             );
232 
233 assign flag_div_frame_or_flag_novid = flag_div_frame | flag_novid_reg;
234 
235 assign flag_novid_final = (flag_novid_reg && flag_div_frame) ? (1'b0) :
    (flag_div_frame_or_flag_novid);
236 
237 endmodule
```

Listing .1 – Top Level

```
1  module atrasador_20 (
2              input clk, enable, rst_geral,
3              input signed[15:0] in,
4              output signed [15:0] out_data_atrasado
5              );
6 
7  reg signed [15:0] in_n_1, in_n_2, in_n_3, in_n_4, in_n_5, in_n_6,
8  in_n_7, in_n_8, in_n_9, in_n_10, in_n_11, in_n_12, in_n_13, in_n_14,
9  in_n_15, in_n_16, in_n_17, in_n_18, in_n_19, in_n_20;
10 
11 always @(posedge clk or posedge rst_geral)
12 begin
13         if(rst_geral)
14         begin
15                 in_n_1  <= 16'd0;
16                 in_n_2  <= 16'd0;
17                 in_n_3  <= 16'd0;
18                 in_n_4  <= 16'd0;
19                 in_n_5  <= 16'd0;
20                 in_n_6  <= 16'd0;
21                 in_n_7  <= 16'd0;
22                 in_n_8  <= 16'd0;
23                 in_n_9  <= 16'd0;
24                 in_n_10 <= 16'd0;
25                 in_n_11 <= 16'd0;
26                 in_n_12 <= 16'd0;
27                 in_n_13 <= 16'd0;
28                 in_n_14 <= 16'd0;
29                 in_n_15 <= 16'd0;
30                 in_n_16 <= 16'd0;
31                 in_n_17 <= 16'd0;
32                 in_n_18 <= 16'd0;
33                 in_n_19 <= 16'd0;
```

```
34                  in_n_20 <= 16'd0;
35          end
36
37          else if(enable)
38          begin
39                  in_n_1 <= in;
40                  in_n_2 <= in_n_1;
41                  in_n_3 <= in_n_2;
42                  in_n_4 <= in_n_3;
43                  in_n_5 <= in_n_4;
44                  in_n_6 <= in_n_5;
45                  in_n_7 <= in_n_6;
46                  in_n_8 <= in_n_7;
47                  in_n_9 <= in_n_8;
48                  in_n_10 <= in_n_9;
49                  in_n_11 <= in_n_10;
50                  in_n_12 <= in_n_11;
51                  in_n_13 <= in_n_12;
52                  in_n_14 <= in_n_13;
53                  in_n_15 <= in_n_14;
54                  in_n_16 <= in_n_15;
55                  in_n_17 <= in_n_16;
56                  in_n_18 <= in_n_17;
57                  in_n_19 <= in_n_18;
58                  in_n_20 <= in_n_19;
59          end
60  end
61
62  assign out_data_atrasado = in_n_20;
63
64  endmodule
```

Listing .2 – Atrasador de 20 amostras

```
1   module berco_DTW (
2               input clk, rst_geral, rst_proc,
3               input signed [31:0] in0, in1, in2,
4               output wire [2:0] req_in,
5               output wire [4:0] out_en,
6               output reg signed [31:0] out0, out1, out2, out3, out4
7               );
8
9   wire signed [31:0] out_proc;
10  reg signed [31:0] in_proc;
11
12  //multiplexador
13  always @(*)
14  begin
```

```
15          case(req_in)
16                  4'b001: in_proc <= in0;
17                  4'b010: in_proc <= in1;
18                  4'b100: in_proc <= in2;
19                  default: in_proc <= 23'd0;
20          endcase
21  end
22
23
24  //instancia processador
25  DTW_proc DTW_proc_inst (
26              .clk(clk),
27              .rst(rst_geral),
28              .io_in(in_proc),
29              .io_out(out_proc),
30              .req_in(req_in),
31              .out_en(out_en),
32              .itr(rst_proc)
33              );
34
35
36  //registradores de saída
37  always @(posedge clk)
38  begin
39          if(rst_geral)
40          begin
41                  out0 <= 32'd0;
42                  out1 <= 32'd0;
43                  out2 <= 32'd0;
44                  out3 <= 32'd0;
45                  out4 <= 32'd0;
46          end
47          else
48          begin
49                  if(out_en[0])
50                          out0 <= out_proc;
51                  if(out_en[1])
52                          out1 <= out_proc;
53                  if(out_en[2])
54                          out2 <= out_proc;
55                  if(out_en[3])
56                          out3 <= out_proc;
57                  if(out_en[4])
58                          out4 <= out_proc;
59          end
60  end
61
```

```
endmodule
```

Listing .3 – Berço para o Processador DTW

```
module berco_FiltroPB_Zc (
            input clk, rst_geral, rst_proc,
            input signed [22:0] in0, in1, in2,
            output reg signed [31:0] out0, out1, out2, out3, out4, out5
            );

wire signed [31:0] out_proc;
reg signed [22:0] in_proc;
wire [2:0] req_in;
wire [5:0] out_en;

//multiplexador
always @(*)
begin
    case(req_in)
        4'b001: in_proc <= in0;
                4'b010: in_proc <= in1;
                4'b100: in_proc <= in2;
                default: in_proc <= 23'd0;
        endcase
end

//instancia processador
FiltroPB_Zc FiltroPB_Zc_inst(
            .clk(clk),
            .rst(rst_geral),
            .io_in(in_proc),
            .io_out(out_proc),
            .req_in(req_in),
            .out_en(out_en),
            .itr(rst_proc)
            );

// registradores de saída
always @(posedge clk)
begin
        if(rst_geral)
        begin
                out0 <= 32'd0;
                out1 <= 32'd0;
                out2 <= 32'd0;
                out3 <= 32'd0;
                out4 <= 32'd0;
                out5 <= 32'd0;
```

```
45          end
46          else
47          begin
48                  if(out_en[0])
49                          out0 <= out_proc;
50                  if(out_en[1])
51                          out1 <= out_proc;
52                  if(out_en[2])
53                          out2 <= out_proc;
54                  if(out_en[3])
55                          out3 <= out_proc;
56                  if(out_en[4])
57                          out4 <= out_proc;
58                  if(out_en[5])
59                          out5 <= out_proc;
60          end
61  end
62
63  endmodule
```

Listing .4 – Berço para o Processador Filtro Nutall + Zero Crossing

```
1   module maq_estados(
2                   input   clk, rst_geral, flag_zc,
3                   input [6:0] usedw,
4                   output reg [1:0] out // out[1] o enable do registrador
                        used_w - 1; out[0] o rst do proc dtw ;
5               );
6
7
8   (* syn_encoding = "safe" *) reg [1:0] state;
9
10  // Declaração dos estados
11  parameter S0 = 0, S1 = 1, S2 = 2, S3 = 3;
12
13  always @ (state) begin
14      case (state)
15          S0:
16              out = 2'b00; //aguardar o flag_zc, nao da reset proc dtw e
                    nao armazena used_w - 1: out[1:0] = 00
17          S1:
18              out = 2'b10; //teve flag_zc armazena num reg o valor de
                    usedw - 1 (enable) e nao da rst proc dtw ainda
19          S2:
20              out = 2'b01; //volta pra zero o enable do reg usedw - 1, e
                    da um rst_proc_dtw
21          S3:
22              out = 2'b00; //volta o rst_proc_dtw para 0
```

```
23          default:
24              out = 2'b00;
25      endcase
26  end
27
28  always @ (posedge clk or posedge rst_geral) begin
29      if (rst_geral)
30          state <= S0;
31      else
32          case (state)
33              S0:
34                  if (flag_zc)
35                      state <= S1;
36                  else
37                      state <= S0;
38              S1:
39                  state <= S2;
40              S2:
41                  state <= S3;
42              S3:
43                  if (usedw > 7'd1)
44                      state <= S3;
45                  else
46                      state <= S0;
47          endcase
48  end
49
50  endmodule
```

Listing .5 – Máquina de Estados

REFERÊNCIAS

BABAYOMI, O. *et al.* Smart grid evolution: Predictive control of distributed energy resources—a review. **International journal of electrical power & energy systems**, Elsevier, v. 147, p. 108812, 2023.

FERNANDES, A. A. *et al.* Detecção de novidades aplicada ao armazenamento de dados em qualidade da energia elétrica. In: **Congresso Brasileiro de Automática-CBA**. [S.l.: s.n.]. v. 2, n. 1.

Intel Corporation. **Cyclone V Device Overview**. [S.l.], 2019. <https://www.intel.com/content/www/us/en/products/details/fpga/cyclone/v.html>.

International Electrotechnical Commission. IEC Standard, **Communication networks and systems for power utility automation - Part 9-2: Specific communication service mapping (SCSM) - Sampled values over ISO/IEC 8802-3**. Geneva, Switzerland: IEC, 2011. Edition 2.0.

KAPISCH, E. B. **Estimação de Frequência e Detecção de Novidades Aplicadas à Compressão de Sinais Elétricos de Potência**. Tese (Doutorado) — Universidade Federal de Juiz de Fora, Faculdade de Engenharia, Programa de Pós-Graduação em Engenharia Elétrica, Juiz de Fora, MG, Brasil, 2019.

KEOGH, E.; RATANAMAHATANA, C. A. Exact indexing of dynamic time warping. **Knowledge and information systems**, Springer, v. 7, n. 3, p. 358–386, 2005.

MOSAYEBI, M.; AZAD, S.; ANVARI-MOGHADDAM, A. Detection and classification of power quality disturbances using deep learning algorithms. In: _____. **Artificial Intelligence in the Operation and Control of Digitalized Power Systems**. Cham: Springer Nature Switzerland, 2024. p. 233–266. ISBN 978-3-031-69358-8. Disponível em: <https://doi.org/10.1007/978-3-031-69358-8_10>.

PIRES, P. V. L. *et al.* Detecção de novidades baseada nas métricas de similaridades dinâmicas: Dtw, edr e twed, aplicadas em sinais de qualidade de energia. In: **Congresso Brasileiro de Automática-CBA**. [S.l.: s.n.], 2022. v. 3, n. 1.

PROCHAZKA, A.; KUKAL, J.; VYSATA, O. Wavelet transform use for feature extraction and eeg signal segments classification. In: **2008 3rd International Symposium on Communications, Control and Signal Processing**. [S.l.: s.n.], 2008. p. 719–722.

SILVA, L. *et al.* Gapless power-quality disturbance recorder. **IEEE Transactions on power delivery**, IEEE, v. 32, n. 2, p. 862–871, 2017.

SILVA, L. R. M. Estudo e implementaçao de técnicas de detecçao e compressao de distúrbios elétricos. 2016.

SILVA, L. R. M. *et al.* The concept of novelty detection applied to power quality. In: IEEE. **2016 IEEE Power and Energy Society General Meeting (PESGM)**. [S.l.], 2016. p. 1–5.