IMPLEMENTAÇÕES DAS OPERAÇÕES GDSC EM FPGA E SIMULINK®

Murilo Mathias Barbosa Bezerra
mmbb@discente.ifpe.edu.br
Prof. Dr. Helber Elias Paz de Souza
helberelias@pesqueira.ifpe.edu.br

RESUMO

Este trabalho tem por objetivo realizar a implementação de um conjunto de operações matemáticas, propostas por Souza (2012), denominadas Generalização do Método de Cancelamento por Sinal Atrasado – GDSC (da sigla em inglês: *Generalized Delayed Signal Cancelation*) as quais podem operar com os vetores ortogonais alfa e beta, advindos da transformada de Clarke, em que essas operações são capazes de eliminar as perturbações harmônicas de baixa ordem. A implementação será dada em um FPGA – *Field Programmable Gate Array*, de forma a tratar sinais de uma rede elétrica trifásica e garantir um sinal de saída livre de perturbações harmônicas, o qual passará por um PLL – *Phase Locked Loop*, para estimar um sinal em sincronia com o sinal existente na rede. Ao término, foi possível realizar a descrição do *hardware* de forma satisfatória, observado o comportamento de diversos sinais submetidos ao sistema, além de reforçar a grande capacidade desse conjunto de operações.

Palavras-chave: Qualidade; Harmônico; GDSC; Matlab®; FPGA.

ABSTRACT

This work aims to implement a set of mathematical operations proposed by Souza (2012), called the Generalized Delayed Signal Cancelation- GDSC, which can operate with the alpha and beta vectors derived from the Clarke transform. These operations are capable of eliminating low-order harmonic disturbances. The implementation will be done on an FPGA – Field Programmable Gate Array, to process signals from a three-phase power grid, ensuring an output signal free from harmonic disturbances. Additionally, a PLL – Phase Locked Loop, will be coupled to the output of the set to synchronize with the existing network signal. In conclusion, it was possible to satisfactorily describe the hardware, observing the behavior of various signals subjected to the system, and reinforcing the significant capability of this set of operations.

Keywords: Quality; Harmonic; GDSC; Matlab®; FPGA.

1 INTRODUÇÃO

Com o avanço da tecnologia e a digitalização dos processos, é cada vez mais comum a utilização de aparelhos eletrônicos no dia a dia das pessoas segundo Garcia (1996), e o crescimento desse tipo de carga continua a chamar a atenção, conforme abordado por Bhagyashri e Pawar (2017), por conseguinte, alinhado as demandas de maior produção energética de forma sustentável, mais perceptível é a presença da geração distribuída (GD) entre os consumidores. Contudo, atrelado ao desenvolvimento tecnológico dos dispositivos, serviços e da demanda por energias limpas, surge a necessidade de evoluir os sistemas elétricos, tanto a geração e transmissão, quanto a rede dos consumidores, pois esta modernização do consumo traz consigo novos desafios. Um desses desafios é a estabilização do sinal da rede elétrica, de forma a manter o sinal elétrico livre de distorções, pois, com o novo modelo de consumo, ou seja, com o aumento dos aparelhos eletrônicos, como computadores, celulares, ou o crescente uso da geração distribuída, maior será a presença das perturbações harmônicas, pois essas cargas/geradoras, conhecidas eletricamente como não lineares, são responsáveis pela produção de harmônicos, inicialmente mais perceptíveis no sinal de corrente elétrica, contudo, devido a impedância da rede, também impactará no sinal da tensão.

Sabendo disso, estudar métodos capazes de manter o sinal elétrico livre de perturbações, tanto na presença quanto na ausência delas, torna-se importante. Sabe-se que fisicamente o que estará presente no sinal será uma distorção do valor ideal, contudo, partindo da série de Fourier, segundo Alexander (2013), que diz que qualquer sinal periódico pode ser decomposto em um somatório de vetores, compostos pelo sinal em frequência fundamental acrescido dos demais sinais com frequência múltipla da fundamental, pode-se separar na rede elétrica o sinal desejado e remover as possíveis anomalias presentes.

Um método já consolidado na literatura, capaz de atender a essa demanda de análise do sinal da rede elétrica, é a Generalização do Método de Cancelamento por Sinal Atrasado (GDSC), que foi proposto por Souza (2012), o qual pode operar na transformação de um sinal trifásico, comumente representado pelas fases *a*, *b* e *c*, para os vetores ortogonais Alfa e Beta, provenientes da transformada de Clarke. Após essa transformação, realiza o armazenamento de algumas amostras do sinal e depois de um intervalo de tempo inferior a um ciclo da fundamental, retorna-o livre de perturbações ou com uma forte atenuação delas.

Este trabalho representa a consolidação de um esforço contínuo de pesquisa e desenvolvimento, iniciado em 2020, no contexto do Programa de Iniciação Científica - PIBIC, e que se prolongou até junho de 2024. Ao longo desse período, orientador e discente realizaram diversas reuniões e trocas de conhecimento, com o objetivo de aprofundar o estudo do controle do sinal da rede elétrica. Como resultado, foi desenvolvido e implementado um controlador baseado em GDSC num FPGA – *Field Programmable Gate Array*, que demonstra a viabilidade e o potencial da solução proposta. Vale ressaltar que previamente foi implementado em Matlab® e no processador digital de sinais (DSP – *Digital Signal Processor*) todo o conjunto de operações GDSC, por Souza (2012) e Nascimento, Souza, Neves, Limongi (2013), e que o foco da implementação inicial em Matlab® foi para desenvolver os conceitos

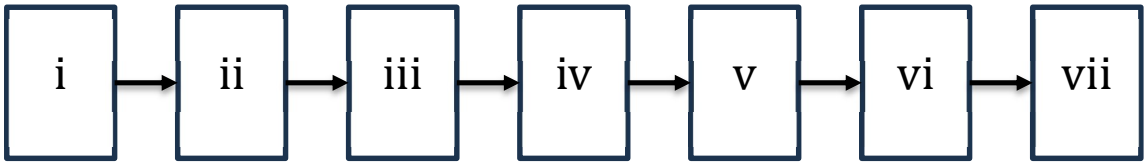
de lógica de programação, análise crítica de dados, de forma a permitir avançar para novas plataformas de programação.

2 DESENVOLVIMENTO

Definido o objetivo geral do trabalho, iniciou-se o desenvolvimento das atividades, que perpassaram por diversas fases, visando o êxito ao término dele. Como base de orientação ao desenvolvimento, seguimos os princípios da metodologia de gestão PDCA (*Plan, Do, Check, Act)*.

De todas as fases e etapas de desenvolvimento, é possível elencar quais seriam as grandes áreas de agrupamento de atividades, as quais são: i) qualidade do sinal na rede elétrica; ii) conceitos matemáticos que permitem a análise do sinal de forma genérica; iii) ferramentas de controle do sinal da rede elétrica; iv) operações GDSC; v) implementação em Matlab®; vi) implementação em FPGA; vii) implementação em Simulink®.

Imagem 01: Fluxograma de Atividades.



Fonte: Própria do Autor.

Cada uma dessas fases será abordada a seguir.

2.1 Qualidade do sinal na rede elétrica

No contexto da qualidade de um sinal elétrico, considerando o regime comumente utilizado na geração, transmissão e distribuição de energia, o qual é o regime de corrente alternada, é possível caracterizar que um sinal de alta qualidade, ou perfeito, segundo Rocha (2016), será aquele que possui tanto em tensão quanto em corrente forma senoidal pura na frequência fundamental, que geralmente é 50 ou 60 Hz, com amplitude de onda constante. Contudo, tal condição é impraticável, uma vez que existe uma constante oscilação de carga na rede, que irá afetar em algum grau essas características, o regime de operação dos transformadores, zona de saturação, além do grande conjunto de cargas que são conhecidas eletricamente como não lineares, sendo estas as grandes responsáveis pela distorção do sinal na rede elétrica.

No contexto do sinal, observando a técnica da série de Fourier, é possível subdividi-lo em um conjunto de senoides com frequências múltiplas da fundamental, chamadas de harmônicas, e observar quais conjuntos de frequências são mais comuns na rede, e qual o tipo de impacto causado por elas. Portanto, seria possível em um sinal elétrico a presença de diversos harmônicos, seguindo a série dos números inteiros, contudo, em redes elétricas trifásicas a presença dos harmônicos

de ordem par é de certa forma rara, sendo notável quando fornos a arco ou máquinas de soldagem são conectados à rede.

Sabendo disso, é possível analisar caso a caso as características daquele setor da rede e dimensionar filtros passivos, que irão atenuar os impactos daqueles harmônicos. Contudo, existem alguns problemas com esse tipo de filtro, pois a depender da variedade de harmônicos presentes seria necessária uma diversidade de componentes, além da possibilidade de haver paralelismo entre eles, sendo esse um tema que não será aprofundado por aqui. Logo, as soluções de filtro ativo estão cada vez mais atrativas, pois podem abranger grande quantidade de harmônicos em uma única solução.

2.2 Conceitos matemáticos

Conforme já citado anteriormente, para tratar de sinais periódicos distorcidos é necessário utilizar algumas ferramentas matemáticas, e a base delas é a série de Fourier. Para tanto, foi realizado um estudo aprofundado sobre os conceitos que orbitam esse tema, como a própria série, a transformada e em conjunto a isso, algumas ferramentas de análise da resposta de sistemas no domínio da frequência, tais como os gráficos de Bode, a transformada de Laplace e suas aplicações relacionadas aos sinais elétricos. Para tal, foi tomado por principal caminho os capítulos quatorze ao dezoito do Alexander (2013). Por conseguinte, adentrou-se nos conceitos do teorema de Fortescue, permitindo assim uma análise detalhada do sinal trifásico.

Por fim, considerando que o objetivo final é aplicar esses conceitos em *softwares/hardwares* com características discretas, grande esforço foi empreendido para assimilar todos esses temas em seus equivalentes no domínio discreto, além de aprofundar o tema dos filtros de Butterworth, tomando por guia o canal no YouTube do Adam Panagos.

2.3 Ferramentas de controle do sinal da rede elétrica

Uma informação relevante no contexto do controle do sinal elétrico é a posição angular de fase. Para tanto, existem soluções que aplicam o chamado SRF-PLL (*Synchronous Reference Frame - Phase Locked Loop*) em sinais trifásicos, quando estes estão livres de perturbações harmônicas ou apenas com perturbações de ordem elevada. Contudo, caso existam perturbações de ordem inferior, devido a própria dinâmica do sistema ele ficaria inaceitavelmente lento ou teria uma resposta bastante imprecisa Souza (2012).

Visando retirar essa limitação do SRF-PLL, existem soluções que podem ser empregadas para remover essas perturbações e entregar na entrada do SRF-PLL um sinal limpo. Duas dessas soluções foram propostas por Souza (2012), sendo uma a Transformada de Fourier de Vetor Espacial (SVFT: *Space Vector Fourier Transform*) e a outra a Generalização do Método de Cancelamento por Sinal Atrasado (GDSC – PLL: *Generalized Delayed Signal Cancelation* – PLL). A SVFT, após ser devidamente compreendida, foi implementada em Matlab®. Alcançando êxito nos resultados, elevando assim a maturidade ao longo do desenvolvimento, permitindo seguir para propostas mais desafiadoras.

2.4 Operações GDSC

As operações GDSC consistem em um conjunto de transformações matemáticas associadas a um armazenamento de amostras capaz de posicionar zeros em determinadas frequências, possibilitando assim a eliminação de perturbações harmônicas do sinal. Conforme demonstrado por Souza (2012), exibido aqui por conveniência, observa-se que qualquer vetor harmônico de ordem hs de sequência positiva ou negativa pode ser representado no seu equivalente nos vetores alfa e beta da seguinte forma:

$$\vec{s}_{\alpha\beta}^{(h_s)} = s_{\alpha\beta}^{(hs)} e^{\mathrm{sgn}(hs) j\varphi^{(h_s)}} e^{j h_s \omega t}$$

aplicando um atraso no sinal de θd amostras resultará em:

$$\vec{s}_{\alpha\beta-\theta d}^{(h_s)} = s_{\alpha\beta}^{(hs)} e^{\mathrm{sgn}(hs) j\varphi^{(h_s)}} e^{j h_s(\omega t-\theta d)} = \vec{s}_{\alpha\beta}^{(h_s)} e^{-j h_s \theta d}$$

combinando o sinal atual e o atrasado, em conjunto com duas constantes, $\vec{a}$ e θr, resultará na transformada GDSC:

$$\vec{f}_{gdsc} = \vec{a}\left(\vec{s}_{\alpha\beta}^{(h_s)} + e^{j\theta r} e^{-j h_s \theta d} \vec{s}_{\alpha\beta}^{(h_s)}\right) = \vec{G}^{(hs)}\, \vec{s}_{\alpha\beta}^{(h_s)}$$

que consiste na multiplicação do sinal por um ganho $\vec{G}^{(hs)}$ sendo representado por:

$$\vec{G}^{(hs)} = \vec{a}\left(1 + e^{j(\theta r - h_s \theta d)}\right).$$

Como o foco do trabalho é implementar essas operações em *softwares*, as operações GDSC em tempo discreto podem ser representadas, conforme demonstrado por Souza (2012), da seguinte forma:

$$\vec{f}_{gdsc}(kT_s) = \vec{a}\left\{\vec{s}_{\alpha\beta}(kT_s) + e^{j\theta r} \vec{s}_{\alpha\beta}((k-kd)\,T_s)\right\}$$

em que T_s é o período de amostragem, $t = kT_s$ é o instante atual e t = $(k-kd)T_s$ é um instante atrasado em kd amostras, em que kd é uma aproximação referente a um atraso no tempo de θd radianos em relação à frequência fundamental, ou seja, $kd = round(\frac{N\theta d}{2\pi})$, em que N é a quantidade de amostras durante um período da fundamental. Considerando a frequência de amostragem adotada para o desenvolvimento de 16 kHz, resultará em $N = \frac{fa}{fr} = \frac{16000}{50} = 320$.

Por fim, conforme demonstrado por Souza (2012), a transformada GDSC em tempo discreto, considerando todos os harmônicos que possam estar presentes, será como descrito a seguir.

$$\vec{f}_{gdsc}(kT_s) = \sum_{h=0}^{\frac{N}{2}-1} \vec{s}_{\alpha\beta+}{}^{(h)}\vec{a}\left[\left(1 + e^{j(\theta r - h\frac{2\pi}{N}kd)}\right)\right] + \sum_{h=1}^{\frac{N}{2}-1} \vec{s}_{\alpha\beta}{}^{(h)*}\vec{a}\left[\left(1 + e^{j(\theta r - h\frac{2\pi}{N}kd)}\right)\right]$$

Ficando claro que as componentes de sequência positiva e negativa multiplicadas por um ganho, o qual é:

$$\vec{G}^{(hs)} = \vec{a}\left(1 + e^{j(\theta r - h\frac{2\pi}{N}kd)}\right).$$

Escolhendo adequadamente os valores de atraso θd, $\vec{a}$ e θr é possível criar um conjunto de operações capazes de eliminar sequências de harmônicos, conforme apresentado por Souza (2012), os quais podem ser vistos na Tabela 01.

Tabela 01 – Constantes das operações harmônicas.

Transformação	kd	θr	a
A	N/2	180°	1/2
B	N/4	90°	1/2
C	N/8	45°	1/2
D	N/16	22,5°	1/2
E	N/32	11,25°	1/2

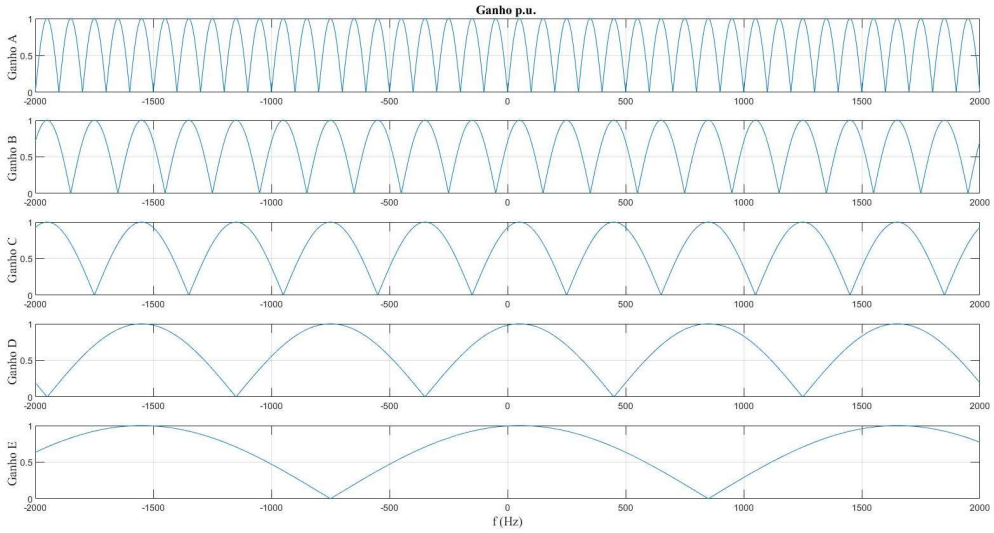
Fonte: Souza (2012).

Conforme notamos, a cada conjunto de valores, associamos a uma denominação, que vai de A a E, em que cada uma delas elimina a seguinte sequência de harmônicos:

- Operação A: $[\mathrm{hd} + (1 + 2\mathrm{n})]$;
- Operação B: $[\mathrm{hd} + (2 + 4\mathrm{n})]$;
- Operação C: $[\mathrm{hd} + (4 + 8\mathrm{n})]$;
- Operação D: $[\mathrm{hd} + (8 + 16\mathrm{n})]$;
- Operação E: $[\mathrm{hd} + (16 + 32\mathrm{n})]$,

sendo hd a componente harmônica desejada, que neste caso será a componente na frequência fundamental de sequência positiva, ou seja, $\mathrm{hd}=1$, em que neste trabalho foi optado por uma frequência fundamental de $50\mathrm{Hz}$. A fim de permitir-se uma maior visualização do comportamento do conjunto, é oportuno observarmos a resposta em frequência de cada uma dessa operações, Imagem 02.

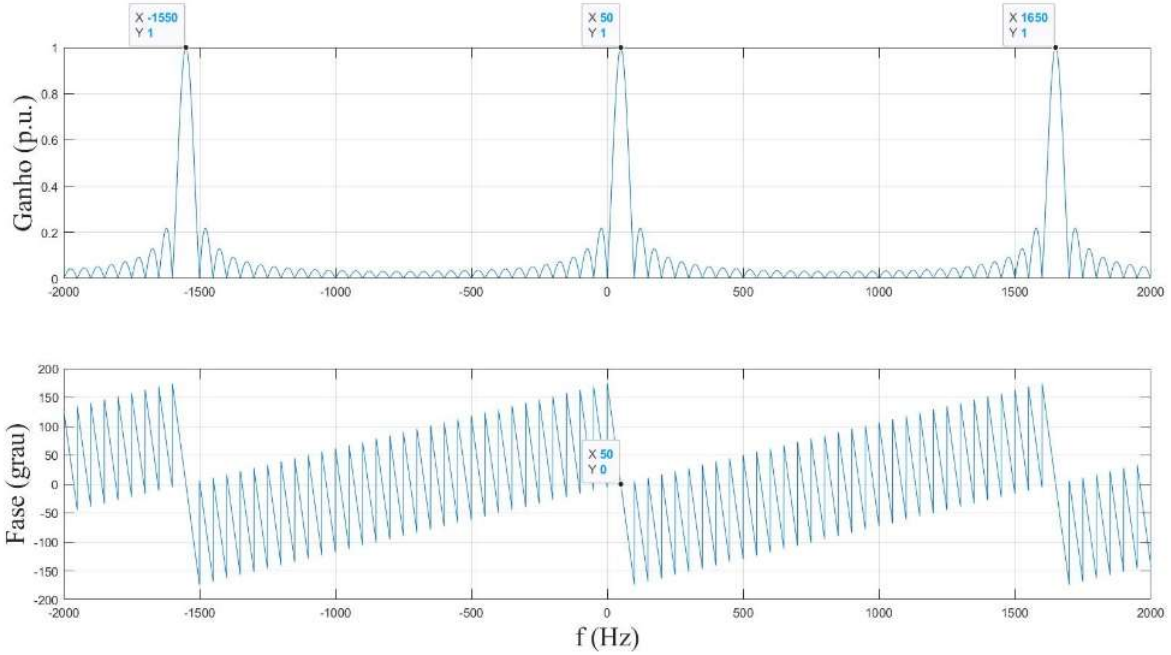
Imagem 02 – Ganhos das operações GDSC.



Fonte: Adaptado de Souza (2012).

Agrupando as cinco operações em cascata, num sinal livre de perturbações harmônicas de baixa ordem, em que a harmônica relevante que poderá estar presente no sinal será de frequência de 1650 Hz, ou seja, 33 vezes maior que a frequência fundamental, ou, simetricamente, -1550 Hz, 31 vezes menor que a frequência fundamental, esse comportamento pode ser observado na Imagem 03.

Imagem 03 – Ganho do conjunto com as cinco operações em cascata.



Fonte: Adaptado de Souza (2012).

Analisado o comportamento geral das operações GDSC, um ponto tem de ser observado, pois todo o princípio de operação delas parte do armazenamento das amostras, equivalente a um atraso na posição angular, conforme a equação $N = \frac{fa}{fr}$,

ou seja, caso exista uma variação na frequência da rede, o número de amostras a serem armazenadas será alterado. Uma forma de contornar isso, demonstrado por Souza (2012), é inserir um filtro passa baixas de Butterworth de segunda ordem, na saída de um primeiro conjunto de operações GDSC, e a saída do bloco, que será a frequência atual estimada da rede, será entregue a um segundo conjunto de operações GDSC, tornando ao final um sistema adaptativo em frequência, que foi denominado como AGDSC. O conjunto AGDSC foi o foco do desenvolvimento.

Logo, para possibilitar a implementação, é necessário analisar quais são as equações que descrevem a GDSC com foco na frequência fundamental.

Segundo Souza (2012), o conjunto de equações referente a cada uma das operações pode ser descrito da seguinte forma:

$$\begin{bmatrix} S\alpha TA(kTs) \\ S\beta TA(kTs) \end{bmatrix} = \frac{1}{2}\begin{bmatrix} S\alpha(kTs) - S\alpha(k-\frac{N}{2})Ts \\ S\beta(kTs) - S\beta(k-\frac{N}{2})Ts \end{bmatrix},$$

$$\begin{bmatrix} S\alpha TB(kTs) \\ S\beta TB(kTs) \end{bmatrix} = \frac{1}{2}\begin{bmatrix} S\alpha TA(kTs) - S\beta TA(k-\frac{N}{4})Ts \\ S\beta TA(kTs) + S\alpha TA(k-\frac{N}{4})Ts \end{bmatrix},$$

$$\begin{bmatrix} S\alpha TC(kTs) \\ S\beta TC(kTs) \end{bmatrix} = \frac{1}{2}\left\{\begin{bmatrix} S\alpha TB(kTs) \\ S\beta TB(kTs) \end{bmatrix} + \frac{\sqrt{2}}{2}\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} S\beta TB(k-\frac{N}{8})Ts \\ S\alpha TB(k-\frac{N}{8})Ts \end{bmatrix}\right\},$$

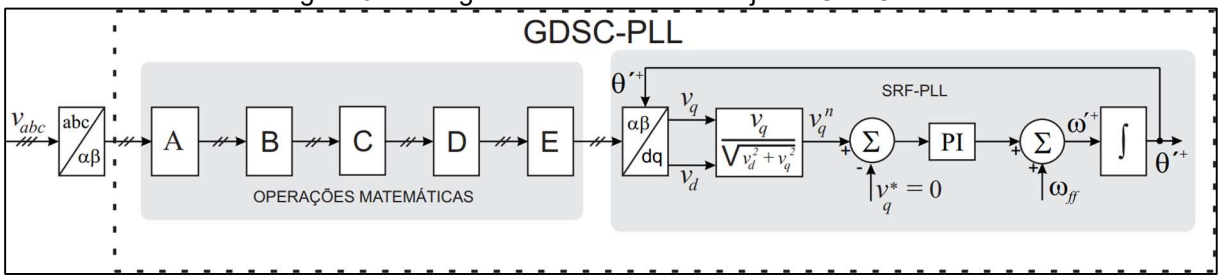
$$\begin{bmatrix} S\alpha TD(kTs) \\ S\beta TD(kTs) \end{bmatrix} = \frac{1}{2}\left\{\begin{bmatrix} S\alpha TC(kTs) \\ S\beta TC(kTs) \end{bmatrix} + \begin{bmatrix} \cos(22{,}5°) & -\sin(22{,}5°) \\ \sin(22{,}5°) & \cos(22{,}5°) \end{bmatrix}\begin{bmatrix} S\beta TC(k-\frac{N}{16})Ts \\ S\alpha TC(k-\frac{N}{16})Ts \end{bmatrix}\right\},$$

$$\begin{bmatrix} S\alpha TE(kTs) \\ S\beta TE(kTs) \end{bmatrix} = \frac{1}{2}\left\{\begin{bmatrix} S\alpha TD(kTs) \\ S\beta TD(kTs) \end{bmatrix} + \begin{bmatrix} \cos(11{,}25°) & -\sin(11{,}25°) \\ \sin(11{,}25°) & \cos(11{,}25°) \end{bmatrix}\begin{bmatrix} S\beta TD(k-\frac{N}{32})Ts \\ S\alpha TD(k-\frac{N}{32})Ts \end{bmatrix}\right\}.$$

Logo, conforme demonstrado por Souza (2012), fica evidente que para implementar as cinco operações em cascata é necessário 12 multiplicações reais e 16 somas reais. Além disso, em relação ao tempo necessário para a correta operação do conjunto, será equivalente ao total de amostras armazenadas, que para as cinco operações corresponderá a (31N/32), ou seja, inferior a um ciclo da fundamental, contudo, é possível escolher apenas algumas das cinco operações, o que proporciona a redução do tempo de resposta das operações.

O diagrama em blocos do conjunto GDSC-PLL pode ser observado na Imagem 04.

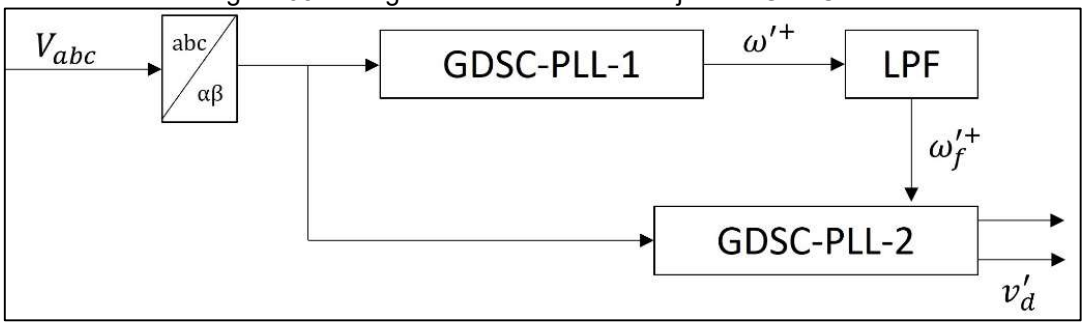
Imagem 04 – Diagrama em blocos do conjunto GDSC-PLL.



Fonte: Souza (2012).

E o conjunto final, AGDSC-PLL, pode ser representado como na Imagem 05.

Imagem 05 – Diagrama em blocos do conjunto AGDSC-PLL.



Fonte: Adaptado de Souza (2012).

Assimilado todo o caminho para chegar a um sinal livre de perturbações harmônicas, foi iniciado o processo de implementação em Matlab®, pois essa ferramenta possibilita um processo de desenvolvimento de certa forma simples, permitindo assim o aumento da maturidade na teoria matemática além de contribuir para o desenvolvimento dos conceitos da lógica de programação. O processo de implementação será descrito a seguir.

2.5 Implementação em Matlab®

Visando reduzir o esforço computacional, evitando o cálculo de cossenos e senos desnecessários, definiu-se como constantes os valores dessas funções trigonométricas, os quais são utilizados nas operações D e E. Por conseguinte, foram definidos os parâmetros do sinal elétrico utilizado, em que se optou por um valor de 100 para o pico das senoides, valor normalizado, podendo ser interpretado como o sinal de tensão ou corrente, o valor de 50 Hz como frequência fundamental, além dos demais parâmetros, como o valor angular inicial do sistema, que por conveniência optamos por 0°. Além disso, é fundamental definir qual será a frequência de amostragem do sistema, a qual foi de 16 kHz, resultando em um total de 320 amostras por ciclo da fundamental. Ainda na definição dos parâmetros, conforme exposto no diagrama em blocos, e de forma a permitir um sistema adaptativo em frequência, foram definidas as constantes do filtro de *Butterworth*, sendo escolhido por frequência de corte o valor de 2 Hz.

Conforme já mencionado, aquelas operações utilizam valores atrasados, ou seja, esses valores precisam ser armazenados na memória, para tanto, foram criados os

vetores, que conseguem armazenar os valores de acordo com a necessidade, além da criação de diversos outros vetores, com o objetivo de armazenar valores como o sinal de entrada, posição angular, sinal de saída, de forma a permitir a visualização do comportamento do conjunto.

A implementação seguiu a lógica sequencial, em que definimos inicialmente qual será o sinal de entrega, ou seja, quais perturbações estarão presentes, logo após realizamos a transformada do sinal para os vetores alfa e beta, os quais são entregues às operações GDSC, seguindo a ordem da A a E, em que cada operação inicia na lógica sequencial pelo vetor alfa e logo em seguida vai ao vetor beta.

Ao término das operações GDSC, o sinal seguirá para o SRF-PLL, sendo necessário realizar mais uma transformação matemática, transpondo o sinal do referencial de Clarke para Park. Um ponto de atenção é que a transformada de Park necessita do cálculo de cossenos e senos, o que é uma tarefa simples para o Matlab®, contudo, ao adentrarmos na implementação em FPGA isso foi um grande desafio, que será abordado posteriormente. Logo após a transformada de Park, uma normalização do sinal do eixo em quadratura é realizada, e em seguida esse valor é entregue ao sistema do SRF-PLL, passando por um controlador proporcional integral (PI), e tendo ao final o valor da frequência da rede, que será entregue ao integrador, resultando no valor estimado da posição angular atual.

Além disso, vale ressaltar que esse valor é o resultado do primeiro conjunto GDSC-PLL, ou seja, ainda não está adaptativo em frequência, para assim o tornar, o valor da frequência resultante desse primeiro bloco foi entregue ao filtro de Butterworth, removendo assim as possíveis perturbações de alta frequência, e com o valor estimado da frequência da rede elétrica, o valor preciso das amostras a serem armazenadas em cada uma das operações foi calculado. Logo após, iniciou-se as operações do segundo conjunto GDSC, seguindo o mesmo passo a passo do anterior, tendo por produto final o valor estimado da posição angular do vetor de sequência positiva na frequência fundamental, além do sinal de saída livre de perturbações, que pode ser obtido aplicando a inversa de Park e Clarke. Um pequeno adendo é que como o valor da posição angular tende a crescer infinitamente ao longo do tempo, realizou-se o “travamento” entre π e -π, ou seja, 180° e -180°, o que não alterará o valor angular, dada a sua característica cíclica.

Por fim, com o objetivo de aprofundar o entendimento sobre os métodos de controle e lógica de programação, realizamos a implementação em Matlab® da SVFT, que não será aprofundada no texto, mas os resultados serão exibidos.

2.6 Implementação em FPGA

Finalizadas as implementações e ajustes em Matlab®, sendo necessário um período em torno de nove meses, iniciaram-se as pesquisas sobre os requisitos para a implementação no FPGA, incluindo o estudo de um artigo do Nascimento, Souza, Neves, Limongi (2013), que traz uma metodologia de implementação das operações em FPGA, e um comparativo com o processador digital de sinais (*Digital Signal Processor - DSP*). Alguns pontos precisam estar definidos, sobre a placa eletrônica, ou seja, o próprio FPGA, escolhida para a implementação, a qual foi selecionada por um outro estudante durante sua iniciação científica, na qual foi dado seguimento após sua saída, e com base nessa escolha seguimos os procedimentos. A placa foi o FPGA Cyclone 10 LP Evaluation Kit, com o chip Intel 10CL025YU256I7G.

O *software* utilizado para realizar a implementação foi o Intel® Quartus® II Prime Lite Edition 17.1, que possui suporte à placa que foi selecionada. Em relação ao processo de desenvolvimento, apresentaram-se grandes desafios, pois tudo era uma novidade, e, ao menos ao longo da prototipação, poucas eram as informações disponíveis, sendo encontradas apenas em outros idiomas, como no caso das aulas no Youtube do canal Clock Fabrick Bard (2017), que apresenta em detalhes essa placa em específico. Um outro grande desafio ao desenvolvimento é que o sistema de descrição de *hardware* que se trabalhou é baseado em linguagem de baixo nível, a qual foi a SystemVerilog.

Definidos a placa, o software e a linguagem de descrição, iniciaram-se as etapas de desenvolvimento do código. Devido ao fato de ser uma descrição de *hardware*, ou seja, um sistema físico, alguns pontos precisam ser definidos previamente, como as entradas e saídas do sistema, para tal, definiu-se que esse sistema deverá possuir três canais de entrada, para cada uma das fases do sistema *abc*, um sinal de Clock, as três fases de saída, para o sinal corrigido, e uma outra saída, a qual informará a posição angular estimada da FFPS, além de um sinal de Clock de saída.

Logo após, um ponto de extrema importância é a criação e definição das características das variáveis utilizadas, como padrão. Foi definido que o desenvolvimento seria com variáveis de 32 bits, sendo um bit para o sinal, positivo ou negativo, 11 bits para inteiros e 20 bits para fracionários, possibilitando assim um amplo alcance para o desenvolvimento. Um segundo tipo de variável utilizada foi a reg, ou registradores, aplicada para armazenar o sinal, de acordo com as necessidades de cada uma das operações. Com o término da definição das variáveis, conforme feito no Matlab®, também foram definidas as constantes, reduzindo assim o esforço computacional ao longo do processamento. Na Imagem 06 aparece uma ilustração do processo.

Imagem 06 – Declaração de constantes.

```
bit [31:0] dst, meio, r3s2, r2s2, COSD, SIND, COSE, SINE, R3;

assign dst=  32'b0000_0000_0000_1010_1010_1010_1010_1010;
assign meio= 32'b0000_0000_0000_1000_0000_0000_0000_0000;
assign r3s2= 32'b0000_0000_0000_1101_1101_1011_0011_1101;
assign r2s2= 32'b0000_0000_0000_1011_0101_0000_0100_1111;
assign COSD= 32'b0000_0000_0000_1110_1100_1000_0011_0101;
assign SIND= 32'b0000_0000_0000_0110_0001_1111_0111_1000;
assign COSE= 32'b0000_0000_0000_1111_1011_0001_0100_1011;
assign SINE= 32'b0000_0000_0000_0011_0001_1111_0001_0111;
assign R3= 32'b0000_0000_0001_1011_1011_0110_0111_1010;
```

Fonte: Própria do Autor.

Na Imagem 06, pode-se observar o processo de criação das variáveis do tipo bit, com o comprimento de 32 bits, e logo em seguida a atribuição das constantes a cada uma das variáveis, no formato mencionado anteriormente.

Ademais, realizar operações básicas da matemática, como soma e multiplicação, não é algo trivial no SystemVerilog, para realizar uma soma de forma correta, foi necessário um algoritmo com um código em torno de 40 linhas, e cada soma é feita da seguinte forma: tem-se dois sinais de entrada, esses sinais serão somados, e

resultará num sinal de saída, sendo a soma dos anteriores, ou seja, apenas duas variáveis serão somadas por vez, o algoritmo de soma foi uma adaptação do código disponível na opencores, do desenvolvedor Burke (2019). Em relação à multiplicação, o desafio está na garantia da precisão, pois, quando duas variáveis de 32 bits são multiplicadas entre si, o resultado final será em uma variável de 64 bits, a qual deverá ser truncada, para retornar ao tamanho padrão de 32 bits. A seguir segue o processo de soma e multiplicação, adaptado do Burke (2019), Imagem 07.

Imagem 07 – Algoritmo de soma.

```
/-------------------------- Soma de sB e sC ----------
    if (sB[31] == sC[31] ) begin
          Ctd01[30:0]= sB[30:0] + sC[30:0];
          Ctd01[31]= sB[31];
    end

    else begin

       if (sB[31]==1) begin

          if (sB[30:0]>=sC[30:0]) begin

                Ctd01[30:0]=sB[30:0]-sC[30:0];


                if( Ctd01[30:0]==0) begin
                Ctd01[31]=0;
                end
                else begin
                Ctd01[31]=1;
                end

          end
          else begin
                Ctd01[30:0]=sC[30:0]-sB[30:0];
                Ctd01[31]=0;
          end


       end
       else begin

          if (sB[30:0]>=sC[30:0]) begin

                Ctd01[30:0]=sB[30:0]-sC[30:0];

                Ctd01[31]=0;


          end
          else begin
                Ctd01[30:0]=sC[30:0]-sB[30:0];
                Ctd01[31]=1;
          end

       end

    end
```

Fonte: Própria do autor.

Imagem 08: Algoritmo de multiplicação.

```
Csq01 = Ctd01[30:0] * meio[30:0];
Ctd02[31]=Ctd01[31] ^ meio[31];
Ctd02[30:0]=Csq01[50:20];
```

Fonte: Própria do autor.

Definidos os processos básicos de soma e multiplicação, Imagem 08, e tendo as variáveis criadas, um novo desafio surgiu, pois é fundamental acompanhar ao longo do desenvolvimento como está o comportamento do sistema, de forma a permitir a correção de erros logo no início. Semelhantemente ao que foi feito no Matlab®, alguns sinais de entrada precisam estar presentes, contudo, não existe uma forma simples de gerar esses sinais dentro do Quartus® II, para tanto, foi criado um sistema de interface com o bloco de processamento, ou seja, criou-se o bloco com as operações GDSC, no qual todo o processamento dos sinais será feito, e um segundo bloco, que servirá de base aos sinais de testes, conhecido como Testbench.

Voltando ao problema da geração de sinais, foi criado um algoritmo conversor, em Matlab®, de forma que os sinais com ou sem perturbações eram gerados em Matlab®, armazenados em vetores, e esses valores foram convertidos para o padrão adotado de 32 bits e analisados no sistema. Além disso, após a definição dos sinais, um segundo desafio surgiu, sendo a forma de analisar os sinais de saída do sistema. O Quartus® II conta com o suporte ao ModelSim, fazendo uma interligação entre testbench, processamento, e apresenta o comportamento geral do sistema. Contudo, a análise de dados no Modelsim também segue no padrão binário, o que praticamente impossibilita a análise dos dados. Visando superar essa dificuldade, um segundo algoritmo foi desenvolvido em Matlab®, capaz de converter os dados do sistema binário para o decimal, permitindo assim a visualização de forma gráfica do comportamento do sistema.

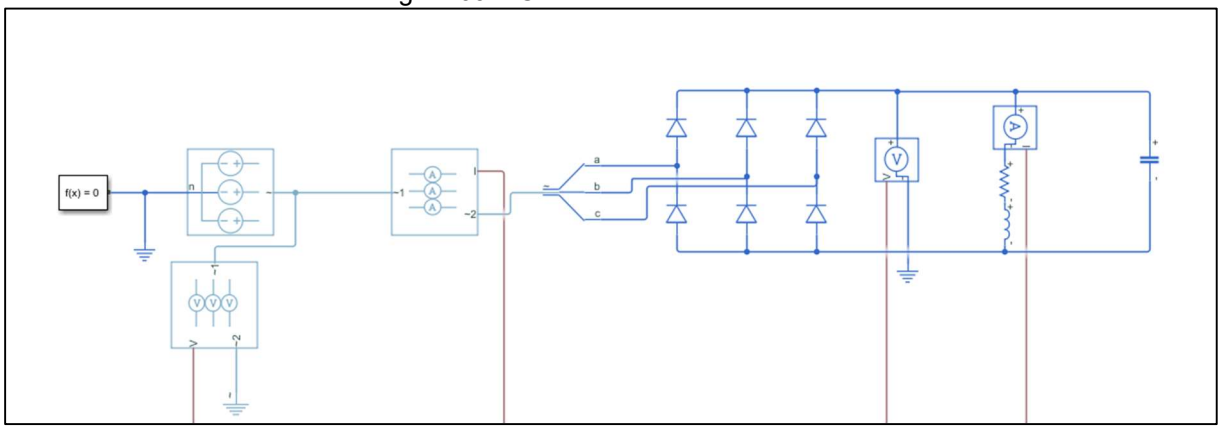
Foi necessário um período de quatro meses para observarmos um Mínimo Produto Viável (MVP) do sistema, com apenas um conjunto de operações GDSC-PLL, ainda não adaptativo em frequência. Mais de vinte versões do algoritmo foram criadas, além dos diversos ajustes no sistema final, pois, inicialmente tentamos implementar todo o conjunto em lógica paralela de processamento, contudo, tivemos diversos problemas de sincronismo, o que nos fez voltar a lógica sequencial, de forma a obter em tempo hábil o sistema funcional. Após a primeira versão do sistema, continuou-se com diversas conversas e estudos, de forma a permitir a implementação do sistema adaptativo em frequência.

No segundo período de desenvolvimento, foi investido em torno de oito meses até a versão final do sistema adaptativo em frequência, e em paralelo era analisado o processo de comunicação com conversores analógico-digital, e as formas de comunicação com a placa, contudo, teve-se um grande impasse nesse processo, pois não foi encontrada uma forma de comunicação adequada ao desenvolvimento. Seria necessário desenvolver todo um sistema de comunicação a parte, de forma a captar as informações dos sensores, garantir o sincronismo e entregar ao bloco de processamento, para posterior retorno a um conversor digital-analógico. Os resultados das implementações serão apresentados posteriormente.

2.7 Implementação em Simulink®

Por fim, como forma de consolidar o conhecimento e trazer a visualização do comportamento das operações GDSC em uma aproximação de um circuito real, foi feita a implementação na ferramenta de simulação de circuito elétricos do Matlab®, o Simulink® em conjunto com a toolbox Simscape, de um circuito retificador trifásico com carga resistiva indutiva, conforme Imagem 09 a seguir.

Imagem 09 – Circuito Retificador Trifásico.



Fonte: Própria do Autor.

As operações GDSC foram colocadas em cascata, conforme pode ser observado nas Imagens 10, 11 e 12.

Imagem 10 – Transformada de Clarke e operação A e B.

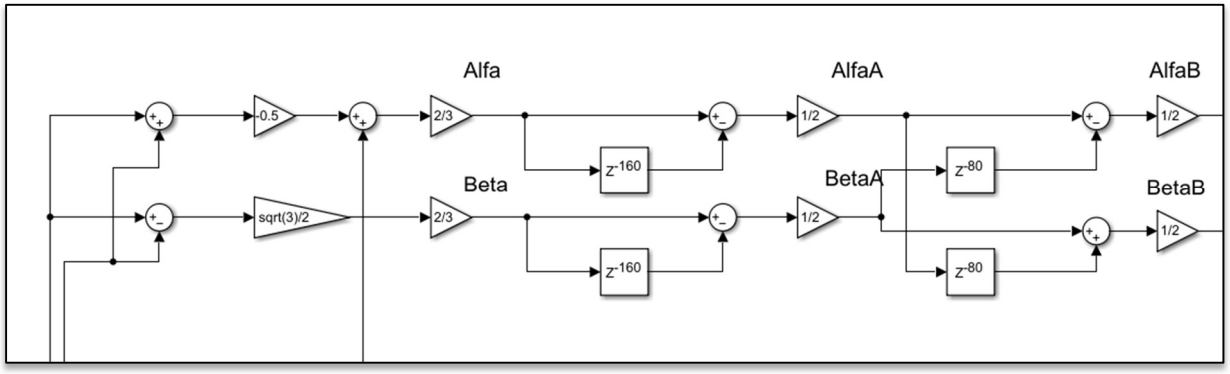


Imagem 11 – Operação C.

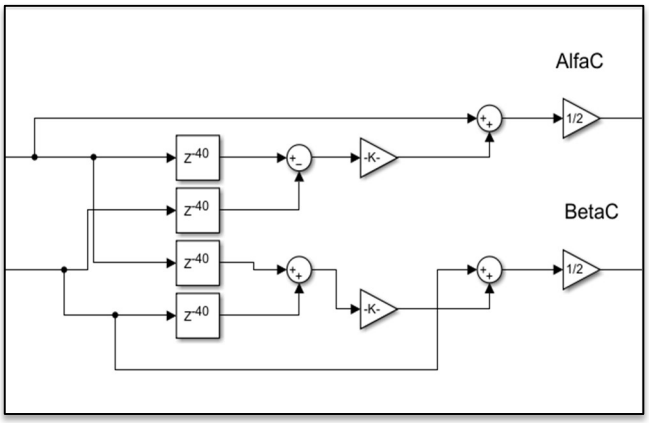
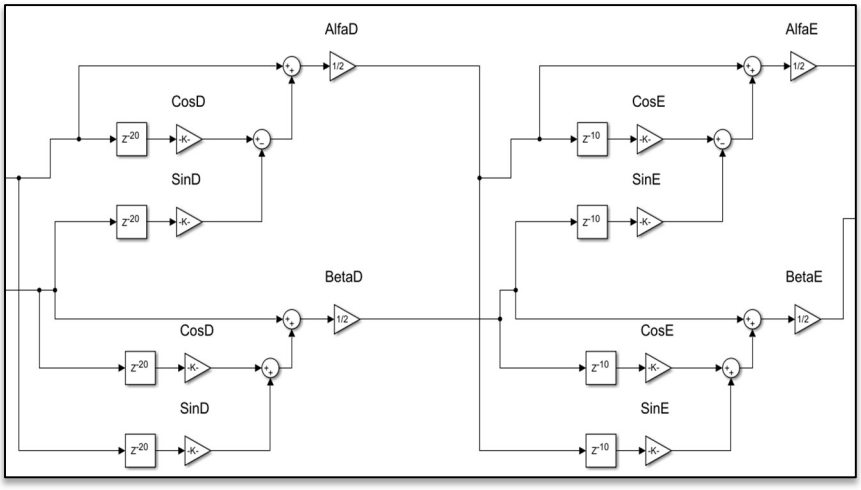


Imagem 12 – Operações D e E.



O comportamento do circuito será apresentado posteriormente, além disso, o sinal que foi gerado no Simulink® foi extraído e inserido no sistema implementado no Quartus® II.

3 METODOLOGIA

A metodologia utilizada ao longo do desenvolvimento, conforme citado anteriormente, tomou por base as boas práticas do PDCA. Alinhado também aos métodos ágeis de gestão e desenvolvimento, definiu-se as grandes áreas do conhecimento e a cada semana um subtema de uma área foi estudado, avanços foram feitos no desenvolvimento do código, na análise dos resultados e assim por diante.

De forma inicial, foi realizado um mapeamento dos requisitos para o sucesso no projeto, definindo assim as grandes áreas, e quais subtemas deveriam ser estudados, implementados, analisados e ajustados. O primeiro passo foi estudar a literatura que orbita o tema, conceitos matemáticos e o próprio conjunto de operações. Por conseguinte, foi necessário estudar conceitos relacionados à linguagem de programação, lógica do sistema e as ferramentas de análise. Em seguida, com o entendimento mínimo da matemática relacionada e da lógica de programação, a implementação foi iniciada, inicialmente num ambiente mais propício ao desenvolvimento do entendimento, que é o Matlab®, e com o sucesso da implementação nele, partiu-se para o objetivo final, que é a descrição do FPGA.

Inicialmente, foi necessário encontrar um software capaz de descrever o hardware, estudar o próprio FPGA (hardware), as linguagens de programação disponíveis nesse software, e iniciar as etapas de descrição e análise do sistema. Cabe ressaltar as peculiaridades do sistema, o ambiente de simulação utilizado foi o ModelSim, que fornecia apenas listas de dados, no sistema binário, e um primeiro desafio seria gerar os sinais de entrada no formato adequado, e analisar a saída do sistema, sendo necessário desenvolver conversores, de binário para decimal, exibir de forma gráfica o comportamento, pois dada a quantidade de amostras por ciclo, seria inviável analisar amostra por amostra.

Praticamente, toda etapa precisou ser criada no código, uma simples soma ou multiplicação necessita de uma série de condições, armazenar os dados, definir as constantes e exibir o resultado. A quantidade de linhas de código necessária, apenas no bloco central do sistema, que contém o processamento dos dados, ultrapassou as 4000 linhas. Uma simples simulação, de um sinal com cinco segundos de duração, dado a quantidade de amostras por ciclo da fundamental, chegou a oitenta mil linhas de dados por fase, para cada sinal analisado.

Concluindo, a metodologia consistiu numa série de revisitações na literatura, em lógicas de programação, linguagens de programação, hardware e software utilizado, mecanismos de análise de dados, e semanalmente foram analisados os resultados, recalculamos as rotas, de forma a garantir que ao término se teria um sistema viável. O esforço deste trabalho consiste na consolidação de tudo o que foi realizado ao longo desses quatro anos, todo o material produzido, em termos de documentos, simulações e códigos fonte.

4 RESULTADOS E ANÁLISES

Diversas foram as simulações realizadas em cada etapa do desenvolvimento, e em cada um dos sistemas desenvolvidos. Algumas dessas simulações já foram apresentadas nos relatórios finais das iniciações científicas, e alguns deles serão aqui reapresentados. Além disso, de forma a consolidar todo o sistema desenvolvido, novas simulações foram realizadas, com algumas modificações, com o intuito de demonstrar a estabilidade do sistema.

Os sinais aqui analisados serão os seguintes:

i) 100% da fundamental de sequência positiva (100% de +1) mais 20% da quarta harmônica de sequência positiva (20% de +4) mais 20% da décima sexta harmônica de sequência positiva (20% de +16) mais 20% da décima primeira harmônica de sequência negativa (20% de -11) mais 20% da nona harmônica de sequência negativa (20% de -9);

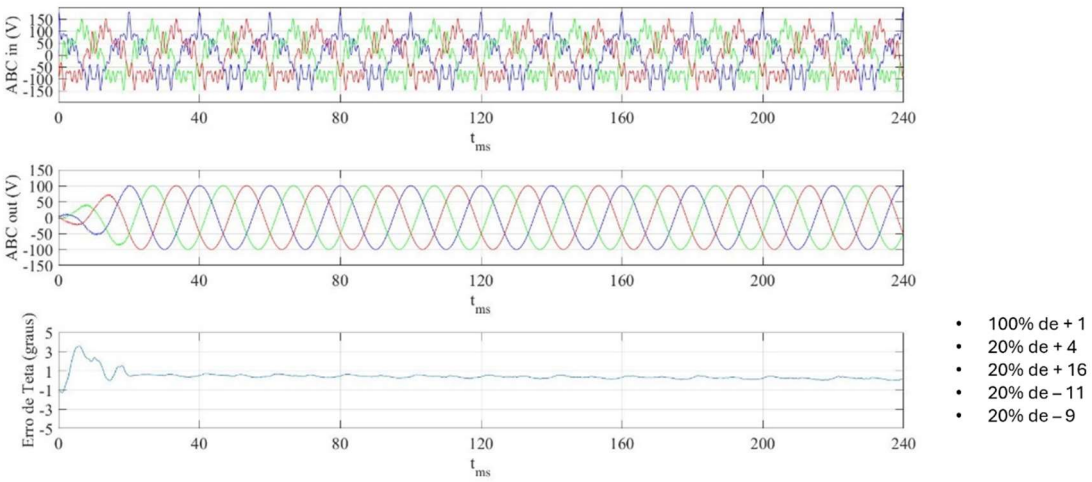
ii) 100% da fundamental de sequência positiva (100% de +1) mais 50% da fundamental de sequência negativa (50% de -1) mais 25% da quinta harmônica de sequência negativa (25% de -5) mais 25% da sétima harmônica de sequência positiva (25% de -7);

iii) Sinal produzido por um circuito com carga RLC no Simulink®.

Convém abreviar o termo frequência fundamental de sequência positiva, que será daqui por diante representado por FFPS.

Iniciando pelo primeiro sinal, Imagem 13, que já foi apresentado no relatório da iniciação científica, mostra-se o comportamento do sistema implementado em FPGA.

Imagem 13 – Entrada, saída e erro na posição angular da FFPS.

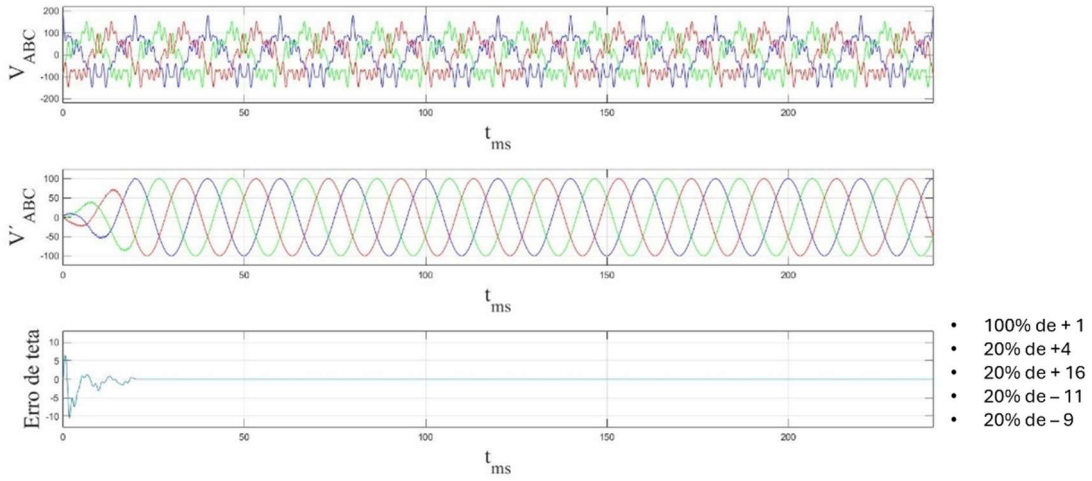


Fonte: Própria do Autor.

Conforme pode ser observado, o sinal de entrada, “ABC in”, apresenta uma perceptível perturbação harmônica, e quando analisamos o sinal de saída, após o período inferior a um ciclo da fundamental, o sinal está totalmente livre de perturbações harmônicas, além do erro angular tender a zero, ocorre uma oscilação ao longo de zero, devido as sucessivas aproximações matemáticas que ocorrem dentro do sistema.

Quando analisamos esse mesmo sinal de entrada no Matlab®, o comportamento é mostrado na Imagem 14.

Imagem 14 - Entrada, saída e erro na posição angular da FFPS.



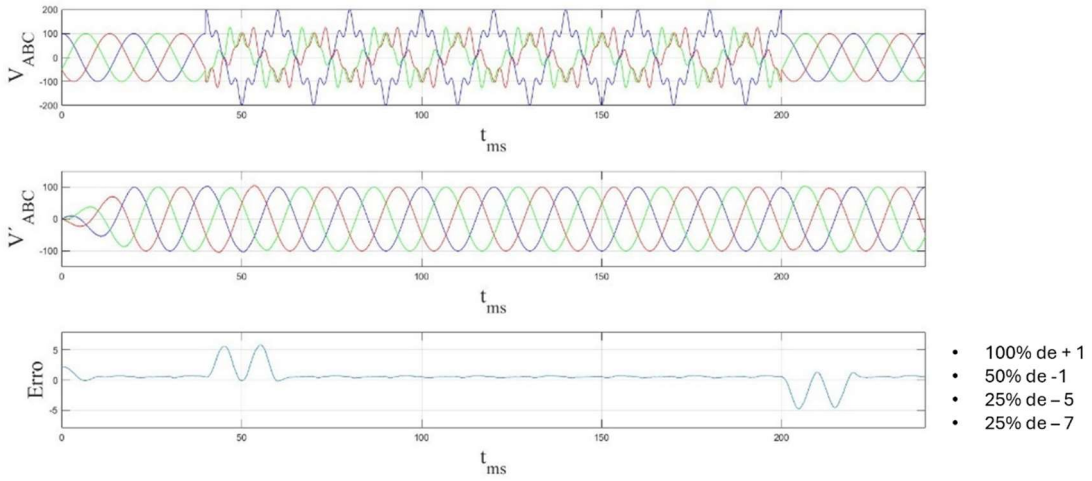
Fonte: Própria do Autor.

Conforme pode ser observado, em termo de eliminação da perturbação harmônica, ambos os sistemas atingem o objetivo, contudo, o que foi implementado em Matlab® apresenta maior estabilização do erro de teta, que de fato tende a zero,

o que já era esperado, uma vez que não existe de forma tão significativa o truncamento durante as operações, além de que o Matlab® é uma ferramenta já consolidada em termos de simulações, os algoritmos internos, como os responsáveis pelas somas e multiplicações, são bem mais avançados e precisos do que aqueles utilizados no Quartus® II.

Continuando para a segunda simulação, tivemos o comportamento mostrado na Imagem 15.

Imagem 15- Entrada, saída e erro na posição angular da FFPS.

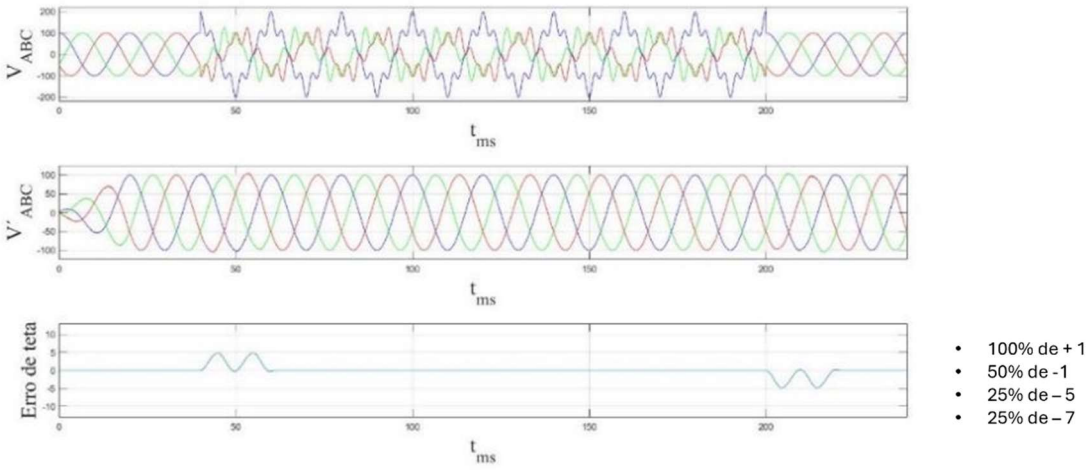


Fonte: Própria do Autor.

Conforme podemos observar, mesmo com a entrada momentânea, durante um período de dez ciclos, o sinal de saída continua estável, livre de perturbações harmônicas, apenas com uma pequena variação na entrada e saída da perturbação, já esperado pois é o tempo necessário, sempre inferior a um ciclo da fundamental, para a correta estimação pelo sistema. Em termos do erro de teta, ele oscila em torno do zero, repetindo o comportamento do sinal anterior, devido às sucessivas aproximações, e apresenta maior variação ao longo da estabilização do sistema, na entrada e saída das perturbações, o que também já é esperado, pois é o tempo necessário para armazenar as amostras para o correto funcionamento.

O comportamento apresentado em Matlab® aparece na Imagem 16.

Imagem 16 - Entrada, saída e erro na posição angular da FFPS.

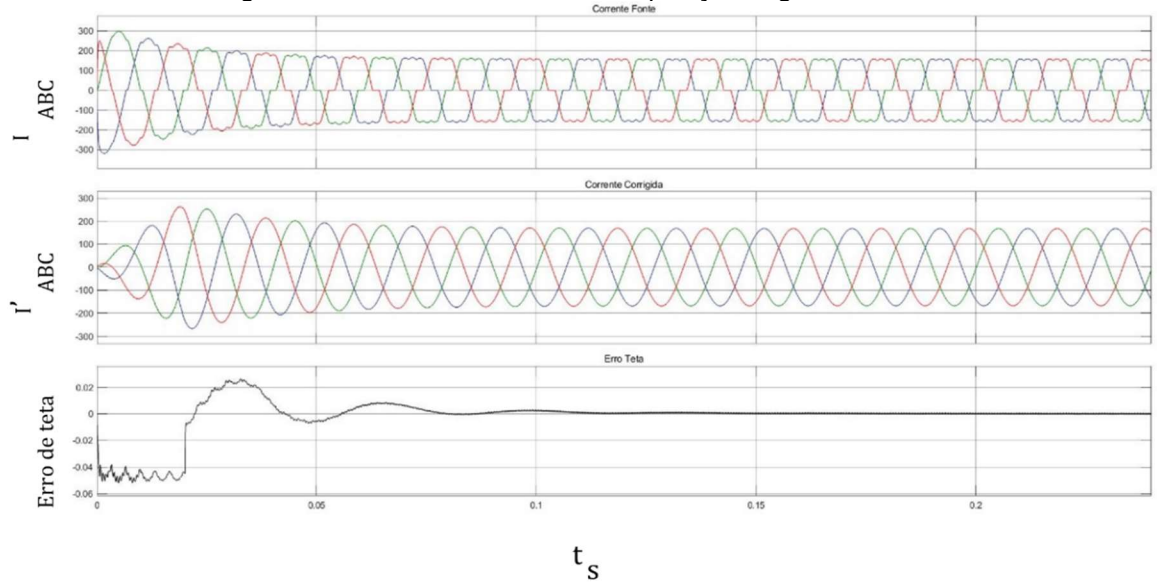


Fonte: Própria do Autor.

Novamente, comportamento bem semelhante ao apresentado no sinal anterior, em termos de eliminação harmônica, ambos os sistemas são satisfatórios, e o Matlab® continua apresentando uma estabilidade do erro de teta em torno do zero maior que a implementação em FPGA, pelos motivos já citados anteriormente.

Por fim, o circuito implementado em Simulink®, Imagem 17, apresentou o seguinte comportamento.

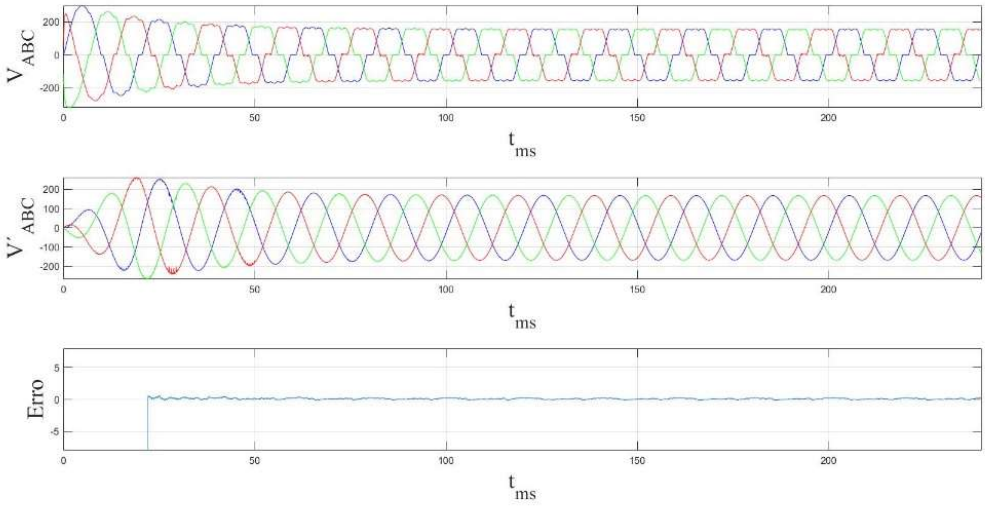
Imagem 17 - Entrada, saída e erro na posição angular da FFPS.



Fonte: Própria do Autor.

Conforme se pode observar, em termos da eliminação harmônica, o sistema é satisfatório, removendo completamente as distorções harmônicas. Em relação ao erro de teta, ele tende a zero ao longo do tempo. O comportamento do sinal gerado em Simulink® e introduzido no sistema em FPGA aparece na Imagem 18.

Imagem 18 - Entrada, saída e erro na posição angular da FFPS.



Fonte: Própria do Autor.

Conforme se pode observar, um comportamento bem semelhante ao que vinha sendo observado nas simulações anteriores se repete, em termos de eliminação da perturbação harmônica, a resposta é satisfatória, e o erro de teta oscila em torno do zero, com uma variação inicial, neste caso, devido a característica transitória gerada pelo Simulink®, o que não acontece com sinais “puros” que são gerados com a somatória de distúrbios harmônicos no Matlab®.

5 CONSIDERAÇÕES FINAIS/ CONCLUSÕES

É perceptível a grande capacidade de controle do sinal da rede elétrica trifásica das operações GDSC, além de que o FPGA é plenamente capaz de realizar todas as operações necessárias para tratar o sinal e ter como produto final um sinal livre de distorções, além de estimar de forma satisfatória a posição angular do sinal da FFPS, com um erro médio por amostra inferior a um grau. Além disso, todo o caminho percorrido para resultar no sistema final foi essencial para o desenvolvimento do discente, elevando em diversos aspectos sua maturidade sobre os temas de controle de sinal elétrico, lógica de programação, e, em especial, análise crítica, pois ao longo do desenvolvimento, é necessário seguir um estreito caminho, onde qualquer desvio resultará em anomalias, gerando imprecisão, fazendo com que o sistema não atue da forma esperada.

Além disso, é evidente a relevância do tema para o profissional de engenharia elétrica, com a crescente demanda por sistemas mais eficientes e que causem cada vez menos impacto ao meio ambiente, como no caso das fontes renováveis, que necessitam desses sistemas de controle da qualidade do sinal da rede elétrica, e do rastreamento da posição angular de forma precisa.

REFERÊNCIAS

[1] Bhagyashri, S. Patil. Pawar, V. S. Power Quality Effects on Nonlinear Loads, International Research Journal of Engineering and Technology (IRJET), 06 de junho de 2017.
[2] ALEXANDER, Charles K. Matthew N. O. Sadiku. Fundamentos de circuitos elétricos, 5° edição, Porto Alegre: AMGH, 2013, Mc Graw Hill e Bookman. 5.ed.
[3] BARD. Clock Fabric. Youtube, 21 dez. 2017. Disponível em: <https://www.youtube.com/watch?v=w_FmjOG6kxQ&ab_channel=ClockFabric>. Acesso em 22 jun. 2024.
[4] GARCIA, Flávio Resende, Sistemas Elétricos de Potência, Inepar Capacitores 1996.
[5] Paulo S. B. Nascimento, Helber E. P. de Souza, Francisco A. S. Neves, Member, IEEE, and Leonardo R. Limongi, FPGA Implementation of the Generalized Delayed Signal Cancelation—Phase Locked Loop Method for Detecting Harmonic Sequence Components in Three-Phase Signals, IEEE Trans. Ind. Elect., vol. 60, n. 2, february 2013.
[6] PANAGOS, Adam. The Z-Transform. Youtube, 01 jul. 2014. Disponível em: <https://www.youtube.com/watch?v=3IllOfXXetA&list=PLdciPPorsHulgKbIuP770TkF4V0vZ3i1O&ab_channel=AdamPanagos>. Acesso em 22 jun. 2024.
[7] ROCHA, Joaquim Eloir, Qualidade de energia elétrica, Universidade Tecnológica Federal do Paraná, Curitiba 2016.
[8] SOUZA, H. E. P. Uma Abordagem Vetorial para a Detecção em Tempo Real de Componentes Harmônicas de Sequência Positiva e Negativa em Sinais Trifásicos. Orientador: Prof. D.Sc. Francisco A. S. Neves. 2012. 176 p. Tese de doutorado (Pós-Graduação em Engenharia Elétrica) - Universidade Federal de Pernambuco, Recife, 2012.
[9] BURKE, Tom. Fixed Point Math Library for Verilog. OpenCores. 2019. Disponível em: https://opencores.org/projects/verilog_fixed_point_math_library. Acesso em 04/08/2022