Universidade Federal de Campina Grande

Centro de Engenharia Elétrica e Informática

Programa de Pós-Graduação em Engenharia Elétrica

FILIPE RAFAEL FARIAS DE SÁ

SIMULAÇÃO DE TRANSITÓRIOS ELETROMAGNÉTICOS EM TEMPO REAL VIA FPGA

Campina Grande - Paraíba
Setembro de 2018

UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
Centro de Engenharia Elétrica e Informática
Programa de Pós-Graduação em Engenharia Elétrica

Simulação de Transitórios Eletromagnéticos em Tempo Real via FPGA

Filipe Rafael Farias de Sá

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para obtenção do grau de Mestre em Engenharia Elétrica.

Área de Concentração: Processamento da Energia

Prof. Washington Luiz Araújo Neves, Ph.D.
Orientador

Prof. Alexandre Cunha Oliveira, D. Sc.
Orientador

Campina Grande – PB
Setembro - 2018

S111s Sá, Filipe Rafael Farias de.
Simulação de transitórios eletromagnéticos em tempo real via FPGA / Filipe Rafael Farias de Sá. – Campina Grande, 2018.
124 f. : il. color.

Dissertação (Mestrado em Engenharia Elétrica) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2018.
"Orientação: Prof. Dr. Washington Luiz Araújo Neves, Prof. Dr. Alexandre Cunha Oliveira".
Referências.

1. Processamento de Energia. 2. Simulação de Transitórios em Tempo Real. 3. Modelos de Componentes Elétricos. 4. Transitórios Eletromagnéticos. 5. Múltiplos Dispositivos. 6. FPGA. I. Neves, Washington Luiz Araújo. II. Oliveira, Alexandre Cunha. III. Título.

CDU 621.311(043)

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECÁRIA SEVERINA SUELI DA SILVA OLIVEIRA CRB-15/225

"SIMULAÇÃO DE TRANSITÓRIOS ELETROMAGNÉTICOS EM TEMPO REAL VIA FPGA"

FILIPE RAFAEL FARIAS DE SÁ

DISSERTAÇÃO APROVADA EM 17/09/2018

WASHINGTON LUIZ ARAÚJO NEVES, Ph.D., UFCG
Orientador(a)

ALEXANDRE CUNHA OLIVEIRA, D.Sc., UFCG
Orientador(a)

MAURÍCIO BELTRÃO DE ROSSITER CORRÊA, D.Sc., UFCG
Examinador(a)

KARCIUS MARCELUS COLAÇO DANTAS, D.Sc., UFCG
Examinador(a)

CAMPINA GRANDE - PB

AGRADECIMENTOS

Agradeço primeiramente a Deus, por sempre ter me dado saúde e força de vontade para superar cada desafio que surgia durante esta caminhada em busca da excelência acadêmica.

Agradeço também a meu pai, Francisco Genesio, por sempre ter me incentivado a ter foco pleno nos estudos, me dando totais condições para tal, e à minha mãe, Rosilda (in memoriam), que enquanto esteve presente em vida fez tanto quanto meu pai.

Agradeço aos meus orientadores, Washington Neves e Alexandre Cunha Oliveira, pela paciência que sempre teve comigo e pela enorme solicitude nas (muitas) vezes em que os procurei em busca de solucionar adversidades surgidas. A orientação de ambos é parte fundamental na construção deste trabalho.

Agradeço também a todos os meus familiares, que sempre torceram bastante por mim, contribuindo com palavras de incentivo tanto em momentos bons quanto nos momentos de adversidade.

Agradeço a todos os amigos, os de longa data e os que ganhei nesses últimos dois anos, os compartilhavam troca de conhecimentos, os que traziam momentos de descontração e leveza, e os que faziam ambas.

Agradeço aos demais professores que puderam transmitir seus conhecimentos nas disciplinas de pós-graduação que cursei, ao PPgEE pela oportunidade de fazer parte deste Programa e a COPELE pela estimada presteza aos esclarecimentos e encaminhamentos de todos os trâmites do PPgEE.

Por fim agradeço ao CNPq, que proporcionou o suporte financeiro para realização deste trabalho.

RESUMO

Um ambiente para simulação em tempo real de transitórios eletromagnéticos em sistemas de potência processados integralmente em FPGA é apresentado nesta dissertação. O processo de construção dessa estrutura envolveu inicialmente a utilização de ambientes de simulação off-line para implementação e avaliação prévia das rotinas que seriam transcritas para linguagem de descrição de hardware. Em seguida se implementava os módulos que descrevem os modelos de componentes elétricos, assim como os que descrevem mecanismos de controle da simulação e de transferência de dados para meios externos ao dispositivo. Foi também implementado um mecanismo de comunicação entre múltiplos dispositivos com a finalidade de segmentar a simulação, sendo esta processada em mais de uma placa FPGA. O ambiente de simulação em tempo real foi avaliado comparando-se simulações dos modelos implementados e de um sistema-teste do IEEE com simulações implementadas off-line. Os resultados obtidos indicaram que o ambiente de simulação se configura uma alternativa viável e aplicável, em termos de exatidão dos resultados e da capacidade de processar passos de cálculo menores em relação a alguns simuladores de tempo real que não possuem recursos nativos para tal, dependendo de ferramentas desenvolvidas com próprios dispositivos FPGA.

Palavras-chave: FPGA, Múltiplos Dispositivos, Modelos de Componentes Elétricos, Simulação de Transitórios em Tempo Real, Transitórios Eletromagnéticos.

ABSTRACT

An environment for real-time simulation of electromagnetic transients in power systems processed in FPGA is presented in this dissertation. The process of constructing this structure initially involved the use of offline simulation environments for the implementation and prior evaluation of routines that are to be transcribed into hardware description language. Then, the modules describing the electric components models were implemented, as well as those describing mechanisms for control of simulation and data transfer to external platforms to the device. A mechanism of communication among multiple devices was also implemented in order to segment the simulation, which is processed in more than one FPGA board. The real-time simulation environment was evaluated by comparing simulations of the implemented models and an IEEE test system with simulations implemented offline. The results indicated that the simulation environment is a viable and applicable alternative in terms of accuracy of the results and the processing capacity in smaller calculation steps that other real time simulators that do not have native features for such, and depending of tools developed with devices FPGA.

Keywords: Electrical Components Models, Electromagnetic Transients, FPGA, Multiple Devices, Real Time Transients Simulation.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ATP – Alternative Transient Program

DLL – Dynamic-Link Library

EMTP – Electromagnetic Transients Program

FPGA – Field Programmable Gate Array

GPIO – General-purpose input/output

IEEE – Institute of Electrical and Electronics Engineers

RAM – Random Access Memory

RTDS – Real Time Digital Simulator

STATCOM – Static Synchronous Compesator

UFCG – Universidade Federal de Campina Grande

SUMÁRIO

1 INTRODUÇÃO

O crescimento da computação digital ocorrido durante a segunda metade do século XX, além de influenciar em incontáveis avanços nas mais diversas áreas do conhecimento, foi de grande contribuição para o surgimento e constante aprimoramento de simuladores digitais. Tratando em especial das plataformas de simulação voltadas a estudos de sistemas elétricos, as mais utilizadas possuem a denominação de programas computacionais do tipo EMTP (Electromagnetic Transients Program) (DOMMEL, 1969). Programas deste tipo realizam simulações off-line, na qual o sistema simulado não acompanha em tempo real possíveis variações das condições do sistema avaliado, quando se necessita de uma resposta em uma velocidade equivalente a resposta natural do sistema, considerando ele ser real. Com isso, apesar dos programas de simulação off-line terem sua importância inestimável nos estudos de transitórios, tem-se nas simulações do tipo tempo real a solução para estas necessidades apresentadas.

Simulações em tempo real são comumente processadas utilizando-se de hardwares dedicados a este tipo de simulação, realizando o processamento do equacionamento dos modelos utilizados na velocidade em que se deseja ter uma resposta. Assim é possível processar equações que modelam o funcionamento de um sistema em passos de tempo equivalentes ao qual o fenômeno ocorre na prática.

1.1 MOTIVAÇÃO

Nas últimas décadas algumas empresas se especializaram em desenvolver plataformas que atuam efetuando simulações em tempo real. Dentre elas vale destacar a RTDS Technologies com o RTDS Simulator e a Opal-RT com alguns simuladores como o Hypersim. Todos estes são muito difundidos e utilizados em muitas pesquisas na área de sistemas elétricos que existem atualmente. Entretanto, estes simuladores citados esbarravam em um fator limitador, que foi possível de ser otimizado. Este fator é o valor mínimo de passo de cálculo utilizado pelas plataformas mencionadas, que seria da ordem de grandeza de dezenas de microssegundos.

No RTDS, por exemplo, desenvolveu-se uma forma de reduzir esse passo de cálculo mínimo acoplando dispositivos FPGA (Field Programmable Gate Array) ao hardware da plataforma de simulação e gerando a ferramenta small time step. Ela é comumente aplicada a sistemas que possuem chaveamentos de alta frequência, simulando em FPGA os trechos que contem esses chaveamentos com um passo de cálculo de 1 a 4 μs e o restante do sistema no núcleo principal de processamento com o passo de cálculo padrão de 25 a 50 μs (RTDS TECHNOLOGIES, 2005).

A simulação processada com dois passos de cálculo distintos propicia a obtenção de um maior nível de detalhamento nas curvas de alta frequencia referentes aos sinais de elementos chaveados dos sistemas, bem como permite a representação de grandes redes processadas com passos de cálculo maiores a fim de se ter tempos de execução rápidos o suficiente para a finalidade de simular em tempo real.

Pesquisou-se trabalhos em que são avaliados alguns obstáculos encontrados ao se utilizar o small time step, onde são apresentadas soluções para a questão da mudança de matrizes admitância em função do estado de uma chave (modelos de chave com admitância constante) e da sincronização entre o dispositivo FPGA acoplado e o processador principal operando com passos de cálculo distintos (OU et al., 2014; QI et al., 2009; QI et al., 2007).

Diante disto observou-se que, sendo um FPGA capaz de atuar como uma ferramenta de solução para restrições identificadas em recursos nativos do processador principal de um simulador em tempo real, seria possível buscar formas de realizar este processamento baseado em FPGA. Assim, foi proposto como tema deste trabalho de Mestrado a execução de simulações em tempo real de modelos de elementos que compõem sistemas de potência, sendo estes processados integralmente em FPGA. Avaliando a literatura abrangente ao tema percebeu-se que processar toda a simulação em FPGA é uma ideia que já é mais difundida em outros países (MATAR e IRAVANI, 2009; CHEN e DINAVAHI, 2009; RAZZAGHI, 2016), porém no Brasil ainda há poucos trabalhos voltados para esta linha de pesquisa. Especificamente dentro do ambiente da UFCG ainda não havia estudos que visasse explorar a potencialidade deste tipo de dispositivo atuando em auxilio ao desenvolvimento de pesquisas envolvendo simuladores de sistemas elétricos.

Tratando a respeito da escolha de dispositivos FPGA, ela ocorreu por alguns fatores. Um deles é devido aos recursos que este tipo de hardware oferece para otimizar a velocidade de execução das simulações, especialmente a capacidade de processamento

paralelo, onde em um único ciclo de clock o FPGA é capaz de executar múltiplas tarefas de forma simultânea (KILTS, 2007). A execução da simulação em um único passo de cálculo é um outro fator que suprime a necessidade de estabelecer uma sincronização requerida quando se tem partes da simulação processadas em passo de cálculo distintos. Além destes, outros fatores que influenciam na escolha do dispositivo têm a sua relevância e merecem ser mencionadas, como o formato mais compacto do hardware em relação aos simuladores existentes, permitindo, mediante otimizações do estado da arte, a utilização do dispositivo em campo, com a possibilidade de agregar o simulador a ensaios de sistemas reais, como por exemplo em: testes de sistemas de controle da proteção ou de controle de conversores, emuladores de condições características de sistemas para ensaios de comissionamento de equipamentos, dentre outros.

Portanto enxerga-se que esta linha de pesquisa se mostra relevante aos avanços tecnológicos almejados na área de sistemas elétricos, e por meio desta motivação deseja-se demonstrar que é factível investir neste tipo de tecnologia para utilização em pesquisas relacionadas a sistemas de potência.

1.2 OBJETIVOS

Como objetivo geral do trabalho almejou-se implementar um ambiente processado em FPGA, cuja definição é um setup de testes para modelos e ferramentas auxiliares a simulação. Ressalta-se que não há uma proposição de algo que, no geral, nunca houvesse sido implementado antes, nem se desejou realizar comparações qualitativas com os simuladores existentes exemplificados anteriormente. O que se pretendeu, na prática, foi apresentar procedimentos referentes a construção da forma de processamento computacional destes modelos e das ferramentas auxiliares, de modo que se tenha um equipamento que atue no sentido de ser uma caixa aberta relativamente a como se configurar um hardware dedicado para a finalidade de simular em tempo real.

Apesar de apresentar as vantagens descritas na seção anterior, dependendo dos recursos oferecidos pelos dispositivos FPGA que se tem disponíveis, nem sempre é possível simular o sistema que se deseja representando-o totalmente, se fazendo muitas vezes necessário o uso de modelos equivalentes reduzidos (MATAR et al., 2004; MATAR e IRAVANI, 2007), ou adquirir um dispositivo de maior recurso. Assim, além do objetivo geral apresentado, desejou-se realizar a comunicação destes dispositivos

realizando a simulação de forma simultânea como se estivesse sendo processada em uma única placa, apresentando assim uma alternativa a esta limitação mencionada.

A análise das simulações ocorreu por meio da obtenção de curvas de tensões e correntes e da comparação destas curvas, em termos de exatidão dos resultados, com curvas obtidas através de simulações off-line processadas com mesmo passo de cálculo da simulação em tempo real. O objetivo destas simulações off-line é apenas fornecer parâmetros de referência a serem atingidos nas curvas resultantes do processamento em tempo real. Uma eventual comparação em termos qualitativos entre os dois tipos de processamento foge do escopo deste trabalho, tendo em vista que requer uma análise mais minuciosa em relação ao tipo especifico de aplicação que será dada ao simulador, uma vez que, embora o tempo de simulação possa ser o mesmo, o tempo de execução (velocidade de processamento) de cada um difere. Para tal foi avaliada a ocorrência de transitórios em modelos de componentes de sistemas elétricos, como por exemplo, procedentes da propagação de ondas em linhas de transmissão baseado na Teoria das Ondas Viajantes e da análise de um sistema modelo de referência do IEEE para estudos de transitórios (IEEE Power System Relaying Committee, 2004) utilizados para validação da metodologia proposta.

Como objetivos específicos são elencados os seguintes pontos:

- Estudar modelos clássicos de elementos elétricos desenvolvidos para simulações de transitórios eletromagnéticos;
- Avaliar o estado da arte a respeito de simulações em tempo real de sistemas elétricos processadas integralmente em FPGA;
- Implementar um ambiente de simulação em FPGA configurado para sintetizar modelos de elementos de sistemas elétricos de potência;
- Construir um mecanismo de comunicação que possibilite a transferência bidirecional de dados entre múltiplos dispositivos FPGA conectados.

1.3 CONTRIBUIÇÕES

As principais contribuições que se almeja com o desenvolvimento desta pesquisa são:

- Construção de um conjunto de módulos representando de forma genérica modelos de componentes elétricos (passíveis de serem aplicados em trabalhos futuros);
- Desenvolvimento de um procedimento para conexão de múltiplos dispositivos FPGA, possibilitando simulações de sistemas de grande porte com maiores níveis de detalhes.
- Estimativas para definição dos tamanhos de registradores de armazenamento de variáveis intermediárias e da capacidade de processamento para uma das placas utilizadas neste trabalho.

1.4 ESTRUTURA DO TEXTO

Este documento é organizado em seis capítulos, sendo o primeiro este introdutório. Os demais são listados a seguir:

- Capítulo 2: Fundamentação teórica;
- Capítulo 3: Revisão Bibliográfica;
- Capítulo 4: Apresentação de aspectos estruturais referentes a construção do ambiente de simulação em FPGA;
- Capítulo 5: Apresentação e análise dos resultados obtidos;
- Capítulo 6: Apresentação das conclusões obtidas.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são discutidos alguns dos fundamentos mais relevantes referentes à pesquisa apresentada nesta dissertação. É mostrada a teoria que rege o comportamento físico da propagação de ondas eletromagnéticas em linhas de transmissão, que são partes integrantes de modelos estudados durante esta pesquisa. Também é descrita a modelagem computacional para simulações digitais tanto de linhas de transmissão, como de outros componentes elétricos, além de ser feita uma explanação acerca das características mais relevantes de um dispositivo FPGA, aplicadas ao desenvolvimento deste trabalho, e das demais plataformas de simulação utilizadas para fins de comparação com as simulações implementadas em FPGA.

2.1 PROPAGAÇÃO DE ONDAS EM LINHAS DE TRANSMISSÃO

Conforme demonstrado por conceitos da teoria eletromagnética, os parâmetros elétricos de uma linha de transmissão não são concentrados, mas sim distribuídos ao longo de toda extensão da linha, assim como tensões e correntes são variáveis ao longo desta extensão (espaço) e também variáveis no tempo. Sendo assim, mediante análises físicas e matemáticas foi possível deduzir as equações básicas da propagação de ondas em uma linha de transmissão monofásica, representada na Figura 1.

$$-\frac{\partial v(x,t)}{\partial x} = ri(x,t) + l\frac{\partial i(x,t)}{\partial t}; \tag{1}$$

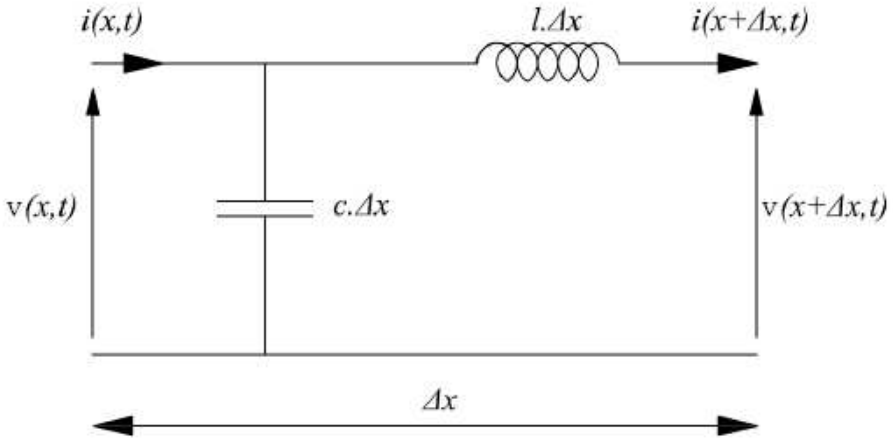
$$-\frac{\partial i(x,t)}{\partial x} = gv(x,t) + c\frac{\partial v(x,t)}{\partial t}. \tag{2}$$

Como o presente trabalho considera modelos de linha sem perdas, as equações (1) e (2) podem ser simplificadas desprezando-se os termos multiplicados por resistências e condutâncias.

$$-\frac{\partial v(x,t)}{\partial x} = l\frac{\partial i(x,t)}{\partial t}; \tag{3}$$

$$-\frac{\partial i(x,t)}{\partial x} = c\frac{\partial v(x,t)}{\partial t}. \tag{4}$$

Figura 1. Circuito incremental representando linhas de transmissão sem perdas a parâmetros distribuídos.



Fonte: Dommel (1969).

Uma possível solução para as equações (3) e (4), demonstrada por Dommel (1969), foi obtida partindo-se do fato de que uma onda eletromagnética de tensão ou corrente em determinado ponto da linha é uma soma de componentes que se propagam no sentido progressivo com componentes que se propagam no sentido regressivo. Matematicamente isto pode ser representado por uma composição de funções que variam no espaço e no tempo.

$$i(x,t) = f_1(x-ut) + f_2(x+ut); \tag{5}$$

$$v(x,t) = Z_c \cdot f_1(x-ut) + Z_c \cdot f_2(x+ut). \tag{6}$$

Em que: $u = 1/\sqrt{lc}$ e $Z_c = \sqrt{l/c}$.

Multiplicando os termos da equação (5) por Z_c e somando-a com a equação (6) é obtida a seguinte expressão:

$$v(x,t) + Z_c \cdot i(x,t) = 2 \cdot Z_c \cdot f_1(x-ut). \tag{7}$$

Realizando procedimento semelhante, desta vez subtraindo a equação (5) da equação (6) obtêm-se:

$$v(x,t) - Z_c \cdot i(x,t) = -2 \cdot Z_c \cdot f_2(x-ut). \tag{8}$$

O significado físico destas equações é que, para ondas eletromagnéticas que se propaguem com velocidade constante, os termos à esquerda das equações (7) e (8) serão uma constante, qualquer que seja o ponto analisado ao longo do comprimento da linha.

Definindo-se τ como o tempo necessário para uma onda percorrer toda a extensão de uma linha, a uma velocidade constante de valor u:

$$\tau = \frac{x}{u}. \tag{9}$$

Conclui-se que, quando uma onda parte de uma das extremidades da linha em um tempo t − τ, percorre toda sua extensão a uma velocidade constante u e atinge a outra extremidade em um tempo t, a expressão que relaciona as grandezas de corrente e tensão, de acordo com o princípio observado na equação (7) é:

$$v_k(t-\tau) + Z_c i_{km}(t-\tau) = v_m(t) + Z_c\big(-i_{mk}(t)\big). \tag{10}$$

Onde k e m são nomes dados aos terminais da linha mostrada na figura 1.

Para linhas polifásicas as equações que regem o comportamento da propagação de ondas (1) e (2) são substituídas por equações matriciais com matrizes de ordem correspondente ao número de fases do sistema.

$$-\frac{\partial[v(x,t)]}{\partial x} = [l]\frac{\partial[i(x,t)]}{\partial t}; \tag{11}$$

$$-\frac{\partial[i(x,t)]}{\partial x} = [c]\frac{\partial[v(x,t)]}{\partial t}. \tag{12}$$

As matrizes de indutâncias e capacitâncias são compostas por elementos próprios de cada fase, estando estes na diagonal principal, e elementos de acoplamento mútuo entre fases, dispostos fora da diagonal principal. A existência destes acoplamentos mútuos torna a solução destas equações bastante complexa de ser obtida. Essa dificuldade pode ser contornada utilizando-se o artifício das transformadas modais, em que as fases do sistema são convertidas em modos desacoplados entre si, com as equações de cada um desses modos podendo ser resolvidas de forma similar ao caso monofásico. As matrizes de transformação são aplicadas às grandezas de tensão e corrente de forma que:

$$\big[v_{fase}\big] = [S][V_{mod}]; \tag{13}$$

$$\big[i_{fase}\big] = [Q][J_{mod}]. \tag{14}$$

Em que [S] e [Q] são as matrizes de transformação. Para este trabalho foi utilizada a matriz de Karrenbauer como matriz de transformação.

$$[S] = [Q] = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 1-n & 1 & \cdots & 1 \\ 1 & 1 & 1-n & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \cdots & 1-n \end{bmatrix}. \tag{15}$$

Em que n é o número de fases.

Substituindo (13) e (14) nas equações (11) e (12), se obtém:

$$-\frac{\partial[V]}{\partial x} = [S]^{-1}[l][Q]\frac{\partial[J]}{\partial t} = [L]\frac{\partial[J]}{\partial t}; \tag{16}$$

$$-\frac{\partial[J]}{\partial x} = [Q]^{-1}[c][S]\frac{\partial[V]}{\partial t} = [C]\frac{\partial[V]}{\partial t}. \quad (17)$$

Após a transformação as matrizes [L] e [C] são matrizes diagonais, ou seja, não possuem elementos de acoplamento entre diferentes modos. Assim, para resolução das equações (16) e (17) é considerado cada modo como sendo um circuito desacoplado dos demais e são aplicados os mesmos princípios para a resolução das equações apresentados para o caso monofásico.

Utilizando-se destes conceitos apresentados até então, Dommel (1969) propôs um modelo de simulação para linhas transmissão, tanto monofásicas quanto polifásicas, que será apresentado e discutido na subseção seguinte.

2.2 MODELOS COMPUTACIONAIS DE ELEMENTOS ELÉTRICOS

Os modelos computacionais de elementos elétricos para simulações digitais são, em sua maior parte, obtidos por meio da aplicação de métodos de integração às equações diferenciais que regem o comportamento físico de tais elementos. Isso implica diretamente na possibilidade de existência de erros numéricos de integração de acordo com a escolha do método de integração realizado. Nesta pesquisa foi escolhido o método de integração trapezoidal para a representação de elementos de circuito a parâmetros concentrados (resistências, indutâncias e capacitâncias lineares). Para modelos de linhas de transmissão foi utilizado o modelo computacional de linhas sem perdas proposto por Dommel (1969), que é um modelo exato, diferentemente dos citados anteriormente, e possui uma característica fundamental para os objetivos desejados no processamento computacional das equações, conforme é visto a seguir.

2.2.1 LINHAS DE TRANSMISSÃO SEM PERDAS

O modelo elétrico para uma linha de transmissão monofásica sem perdas foi desenvolvido por meio das considerações apresentadas anteriormente acerca das equações de propagação de ondas eletromagnéticas, aplicando-se o que está demonstrado na equação (10). Através desta equação, rearranjando os termos, obtém-se a expressão que indica a corrente partindo de um dos terminais da linha:

$$i_{mk}(t) = Y_c \cdot v_m(t) - [Y_c \cdot v_k(t-\tau) + i_{km}(t-\tau)]. \quad (18)$$

Em que: $Y_c = 1/Z_c$

De forma análoga, a corrente que parte da outra extremidade tem a seguinte expressão:

$$i_{km}(t) = Y_c \cdot v_k(t) - [Y_c \cdot v_m(t-\tau) + i_{mk}(t-\tau)]. \quad (19)$$

Das equações (18) e (19) define-se:

$$I_k(t-\tau) = Y_c \cdot v_m(t-\tau) + i_{mk}(t-\tau); \quad (20)$$

$$I_m(t-\tau) = Y_c \cdot v_k(t-\tau) + i_{km}(t-\tau). \quad (21)$$

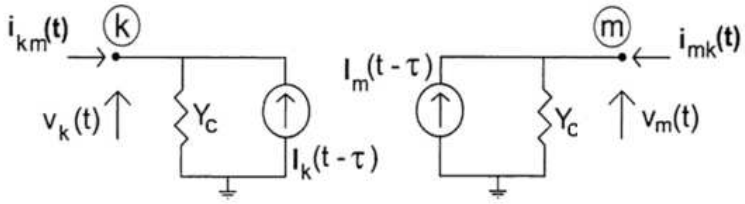
Os termos à esquerda das equações (20) e (21) são denominados fontes de corrente fictícias, que dependem do valor numérico, em termos de amplitude, das formas de onda de tensão e corrente em períodos de tempos passados no terminal oposto da linha. Com isso as equações podem ser reescritas da seguinte forma:

$$i_{km}(t) = Y_c \cdot v_k(t) - I_k(t-\tau); \quad (22)$$

$$i_{mk}(t) = Y_c \cdot v_m(t) - I_m(t-\tau). \quad (23)$$

O circuito elétrico que reproduz fisicamente estas equações é apresentado na Figura 2.

Figura 2. Modelo clássico de linhas de transmissão monofásicas sem perdas para cálculo de transitórios.



Fonte: Dommel (1969).

Este modelo possui duas características relevantes. Uma delas é ser um modelo exato, pois não foi necessária nenhuma aproximação ao longo de sua formulação matemática. A outra é o desacoplamento entre as duas partes do circuito, que simplifica os objetivos de execução de processamento paralelo do modelo.

Vale ressaltar também que o fato das fontes de corrente fictícias do modelo, a cada atualização em determinado passo de cálculo, serem relacionadas a grandezas calculadas em instantes de tempo passados, tornam necessária a presença de vetores atuando como janelas móveis para armazenamento e utilização dos dados históricos no instante necessário para os cálculos.

Para linhas de transmissão polifásicas pode-se calcular as correntes modais com expressões análogas às expressões (22) e (23) para linhas monofásicas.

$$J_{km}^{\mu}(t) = Y_c^{\mu} \cdot V_k^{\mu}(t) - J_k^{\mu}(t-\tau); \quad (24)$$

$$J_{mk}^{\mu}(t) = Y_c^{\mu} \cdot V_m^{\mu}(t) - J_m^{\mu}(t-\tau). \quad (25)$$

Em que:

$$J_k^{\mu}(t-\tau) = Y_c^{\mu} \cdot V_m^{\mu}(t-\tau) + J_{mk}^{\mu}(t-\tau); \quad (26)$$

$$J_m^{\mu}(t-\tau) = Y_c^{\mu} \cdot V_k^{\mu}(t-\tau) + J_{km}^{\mu}(t-\tau). \quad (27)$$

No domínio de fases as expressões das correntes podem ser escritas em sua forma matricial.

$$[i_{km}(t)] = [G][v_k(t)] - [I_k(hist)]; \quad (28)$$

$$[i_{mk}(t)] = [G][v_m(t)] - [I_m(hist)]. \quad (29)$$

Em que:

$$[G] = [Q][Y_c][S]^{-1}; \quad (30)$$

$$[I_k(hist)] = [Q][J_k(t-\tau)]; \quad (31)$$

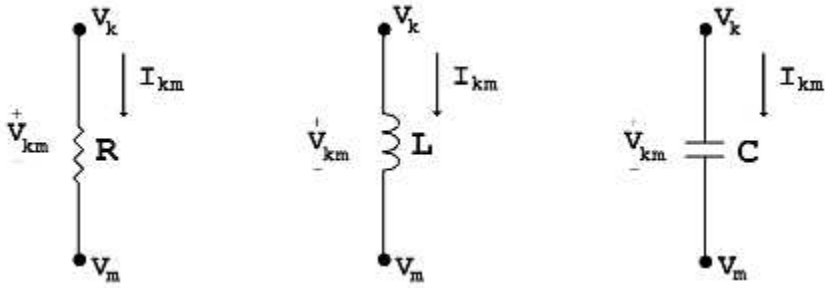
$$[I_m(hist)] = [Q][J_m(t-\tau)]. \quad (32)$$

2.2.2 RESISTOR

A equação física que rege o comportamento de tensão e corrente sobre um resistor é a primeira lei de Ohm, que consiste em uma equação algébrica. Portanto esse elemento tem em seu modelo computacional sua representação exata por meio da equação algébrica original.

$$i_{km}(t) = \frac{v_{km}(t)}{R}. \quad (33)$$

Figura 3. Modelos elétricos de resistores, indutores e capacitores.



Fonte: Dommel (1969).

2.2.3 INDUTOR

Para o indutor da Figura 3 a equação diferencial que descreve o comportamento da tensão e da corrente sobre o elemento é:

$$v_{km}(t) = L\frac{di_{km}(t)}{dt}. \tag{34}$$

Um dos métodos de solução numérica para resolução de uma equação diferencial é a utilização do método de integração trapezoidal. Matematicamente ele é representado pela seguinte expressão:

$$\int_{t-\Delta t}^{t} f(t)dt = \frac{\Delta t}{2}[f(t) + f(t-\Delta t)]. \tag{35}$$

Aplicando este método na equação (35) é obtida a seguinte equação algébrica:

$$i_{km}(t) = \frac{\Delta t}{2L}v_{km}(t) + \frac{\Delta t}{2L}v_{km}(t-\Delta t) + i_{km}(t-\Delta t). \tag{36}$$

Da equação (36) define-se:

$$R_L = \frac{2L}{\Delta t}; \tag{37}$$

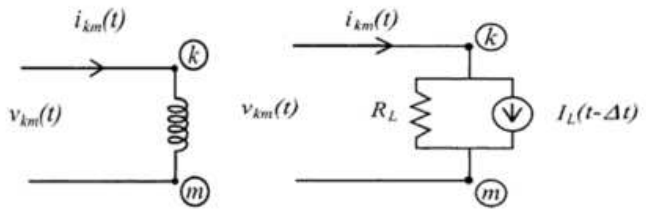
$$I_L(t-\Delta t) = \frac{\Delta t}{2L}v_{km}(t-\Delta t) + i_{km}(t-\Delta t). \tag{38}$$

Podendo ser reescrita:

$$i_{km}(t) = \frac{1}{R_L}v_{km}(t) + I_L(t-\Delta t). \tag{39}$$

A equação (39) pode ser representada em termos de circuito elétrico por uma resistência equivalente R_L em paralelo com uma fonte de corrente fictícia dependente de grandezas calculadas a um passo de tempo anterior. Este circuito equivalente discreto para um indutor é apresentado na Figura 4.

Figura 4. Indutor e seu circuito equivalente discreto.



Fonte: Dommel (1969).

2.2.4 CAPACITOR

A equação diferencial que descreve o comportamento do capacitor apresentado na Figura 4 pode ser escrita:

$$i_{km}(t) = C\frac{dv_{km}(t)}{dt}. \tag{40}$$

Aplicando o método de integração trapezoidal na equação (40) chega-se a seguinte equação:

$$i_{km}(t) = \frac{2C}{\Delta t}v_{km}(t) - \frac{2C}{\Delta t}v_{km}(t-\Delta t) - i_{km}(t-\Delta t). \tag{41}$$

Dela são definidos:

$$R_C = \frac{\Delta t}{2C}; \tag{42}$$

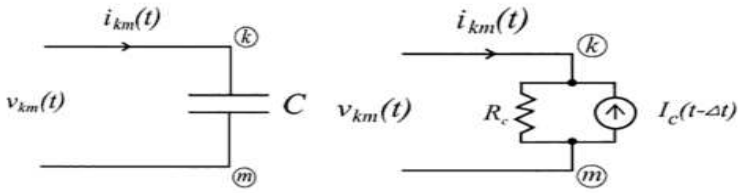
$$I_C(t-\Delta t) = -\frac{2C}{\Delta t}v_{km}(t-\Delta t) - i_{km}(t-\Delta t). \tag{43}$$

Assim a equação (41) pode ser reescrita:

$$i_{km}(t) = \frac{1}{R_C}v_{km}(t) + I_C(t-\Delta t) \tag{44}$$

Esta equação pode ser representada em termos de circuito elétrico por um circuito discreto equivalente, como no caso do indutor. Ele é representado por uma resistência equivalente R_C em paralelo com uma fonte de corrente fictícia que depende da tensão e da corrente calculadas em um passo de tempo anterior. O circuito discreto equivalente para o capacitor é apresentado na Figura 5.

Figura 5. Capacitor e seu circuito equivalente discreto.



Fonte: Dommel (1969).

2.2.5 ELEMENTOS CONCENTRADOS EM CIRCUITOS POLIFÁSICOS

Em circuitos polifásicos deve ser levado em consideração, além dos valores próprios de indutância e capacitância, o acoplamento mútuo que esses elementos geram com as outras fases do sistema. Este acoplamento é representado por meio da utilização

de matrizes de ordem correspondente ao número de fases do sistema para indutores e capacitores polifásicos. Uma expressão geral para o cálculo da corrente em sistemas com mais de uma fase é apresentada a seguir:

$$[i_{km}(t)] = [G]([v_k(t)] - [v_m(t)]) + [I_{km}(t - \Delta t)]; \quad (45)$$

Onde:

$$[G] = ([R] + \frac{2}{\Delta t}[L])^{-1}; \quad (46)$$

$$[I_{km}(t - \Delta t)] = [G][v_{km}(t - \tau)] + [i_{km}(t - \tau)]; \quad (47)$$

para indutores, e

$$[G] = ([R] + \frac{\Delta t}{2}[C]^{-1})^{-1}; \quad (48)$$

$$[I_{km}(t - \Delta t)] = -[G][v_{km}(t - \tau)] - [i_{km}(t - \tau)]; \quad (49)$$

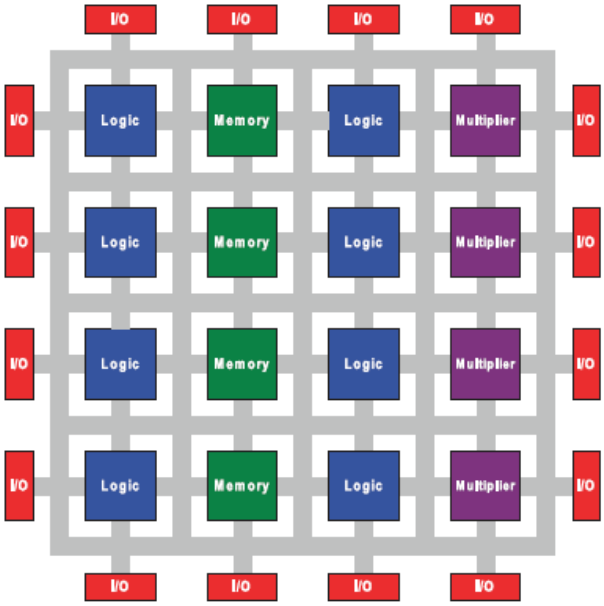
para capacitores.

2.3 PLATAFORMAS DE SIMULAÇÃO

Nesta seção é feita uma breve discussão a respeito das plataformas de simulação utilizadas no desenvolvimento deste trabalho e sobre a forma como são aplicadas para a obtenção dos resultados desejados. Para tal são utilizados três ambientes de simulação. Um deles, ponto chave do desenvolvimento deste trabalho, é a implementação de um ambiente de simulação em tempo real construído utilizando-se dispositivos FPGA. Além deste ambiente, escolheu-se implementar simulações off-line no Alternative Transients Program (ATP) e no PSIM desenvolvido pela Powersim, cujos resultados servem como parâmetros de comparação e análise dos resultados processados em FPGA.

Field Programmable Gate Arrays (FPGAs) são dispositivos eletrônicos programáveis e reconfiguráveis, fabricados a base de Silício, tendo neste elemento semicondutor a capacidade de constituir milhões de circuitos de lógica combinacional e sequencial. Estruturalmente o dispositivo é organizado nos seguintes blocos: blocos entrada e saída, responsáveis por receber e transmitir dados, blocos lógicos configuráveis, que são circuitos que geram a lógica combinacional necessária para o processamento dos dados, blocos de multiplicadores e blocos de memória. Um esquema de como é organizada essa estrutura é apresentado na Figura 6.

Figura 6. Estrutura de um dispositivo FPGA.



Fonte: Kuon, I. (2008).

Em termos de processamento o grande diferencial de um FPGA é sua capacidade de execução de tarefas de forma simultânea, devido ao processamento paralelo de instruções em um mesmo instante. O processamento paralelo pode ocorrer tanto em nível de instanciação de módulos como em nível de operações aritméticas. No segundo caso utiliza-se de múltiplos blocos lógicos de aritmética operando dados de forma simultânea, propiciando ganhos relevantes no tempo de execução em comparação com um processamento sequencial.

Para programar ou reconfigurar um FPGA para alguma aplicação específica utiliza-se uma linguagem de descrição de hardware. Este tipo de linguagem é utilizado para descrever para o hardware o funcionamento e a conectividade do circuito a ser configurado, ou seja, adapta o dispositivo para atender os requisitos específicos da aplicação que vai ser dada a ele. Existem inúmeras linguagens de descrição de hardware utilizadas para design de circuitos digitais. Nesta pesquisa utiliza-se a linguagem Verilog pelo fato de possuir um formato mais prático para descrição rápida de modelos e também uma estruturação mais próxima da linguagem C, utilizada nas simulações no PSIM.

A fim de realizar a configuração do dispositivo FPGA utilizando Verilog deve-se implementar descrições de hardware ao mais alto nível de abstração possível. Isto é obtido com a criação de módulos de simulação que comunicam para o hardware as

ligações lógicas necessárias para configurar cada aplicação requerida pelo usuário. Todos os módulos implementados podem ser instanciados em um módulo principal, que faz a comunicação direta com o FPGA. Uma das maiores vantagens deste tipo de implementação é a possibilidade de execução paralela de diferentes módulos, propiciando ganhos em tempo de execução da simulação.

Com isso são descritos na linguagem Verilog os modelos apresentados na sessão anterior. Essas estruturas de descrição vão gerar ligações entre os blocos do FPGA, permitindo que o hardware execute as simulações.

Apesar das vantagens apresentadas da linguagem Verilog ela possui um fator limitador que é não possuir recursos nativos de aritmética de ponto flutuante, havendo necessidade de se construir um módulo dedicado para este fim. Com isso optou-se por trabalhar com o processamento efetuado em aritmética de ponto fixo. A diferença mais relevante, neste trabalho, entre estes dois tipos de aritmética está no processamento de números fracionários, onde é reservado um espaço no armazenamento dos dados para partes fracionárias processados em aritmética de ponto flutuante e trabalha-se apenas com variáveis inteiras no processamento em ponto fixo.

Em aritmética de ponto fixo, realizando operações de variáveis inteiras em que o resultado pode vir a ser um número fracionário, como em uma divisão, por exemplo, parte do resultado se perde com arredondamento para o menor número inteiro. Uma forma de se amenizar essas perdas é converter o número fracionário em número inteiro utilizando um formato, manipulação algébrica que converte o dado em número inteiro de ordem relativamente grande. Assim a perda de casas decimais ao final do processamento se torna dentro de limites toleráveis, comparando-se com o mesmo resultado processado em ponto flutuante, e será menor quanto maior for a quantidade de bits utilizada na representação em ponto fixo.

O formato é um número inteiro em potência de base 2 que vai fornecer a quantidade de bits com a qual o número, seja ele fracionário ou inteiro, será representado em aritmética de ponto fixo, e é multiplicado pela razão entre este número a ser representado e um valor base, escolhido de forma que a razão possua um valor decimal entre 0 e 1. Matematicamente:

$$N_{PFIXO} = N_{NORMALIZADO} \cdot 2^{B} \tag{50}$$

Onde $N_{NORMALIZADO}$ é a razão entre o número a ser representado e o valor base e B é o número de bits da representação.

Para fins de comparação, em termos de exatidão, dos resultados das simulações em FPGA, outras duas plataformas, softwares de simulação comerciais dedicados à execução de simulações off-line, foram utilizadas. No ATP utilizou-se a ferramenta ATPDraw, que possui alguns blocos para simulação de componentes elétricos de sistemas, como fontes, elementos passivos, modelos de linha, dentre outros. O software utiliza rotinas para implementar os modelos de cada componente usado. O processamento das simulações é realizado em aritmética de ponto flutuante, por padrão do software.

Já no PSIM os modelos computacionais descritos na seção anterior são implementados em DLL's (Dynamic-Link Library), ferramentas computacionais do sistema operacional Microsoft que permitem a implementação e execução de códigos em algumas linguagens de programação, sendo utilizada por meio do PSIM para que o usuário possa implementar seus próprios modelos computacionais desejados para suas simulações. Com isso são elaboradas rotinas na linguagem C para implementação dos modelos da seção anterior, utilizando aritmética de ponto fixo. Dentro das simulações utiliza-se uma DLL para cada componente elétrico do sistema a ser representado. Isso permite aplicar uma condição semelhante ao que ocorre na divisão modular do FPGA, onde as DLL's fazem analogia aos módulos implementados para os componentes elétricos no ambiente de simulação em FPGA.

3 REVISÃO BIBLIOGRÁFICA

Como forma de se obter parâmetros de referência para esta pesquisa foram observados e analisados trabalhos que vêm sendo desenvolvidos ao longo dos anos dentro do tema desta dissertação. Todos eles apresentam a mesma finalidade de propor a execução de simulações em tempo real de sistemas de potência em dispositivos FPGA, explorando as vantagens possíveis na forma de processamento deste tipo de hardware, mas havendo algumas diferenças quanto à metodologia implantada para o desenvolvimento e a execução das pesquisas. A seguir são apresentadas as principais características dos trabalhos avaliados como sendo os mais relevantes dentro do tema proposto.

3.1 TRABALHOS DE MAIOR RELEVÂNCIA

3.1.1 MATAR *ET AL.* (2004)

Foi apresentado um dos primeiros trabalhos voltados para o processamento integral de simulações em tempo real de sistemas de potência, implementadas em FPGA. A metodologia proposta foi descrever para um dispositivo FPGA todas as equações que modelam o sistema a ser simulado, sendo a resolução destas divididas em módulos de processamento: um para cálculo de tensões nos nós e outro para atualização das fontes de corrente. Observou-se uma característica particular deste e de outros trabalhos desenvolvidos pelo autor, sendo o processamento das equações efetuado em um único ciclo de clock (cálculo das tensões na subida, atualização das fontes de corrente na descida), explorando, assim, os recursos de processamento paralelo que um dispositivo FPGA oferece. A utilização da aritmética de ponto fixo no processamento dos dados foi satisfatória, em termos de exatidão dos resultados.

Outra contribuição deste trabalho foi o desenvolvimento de um método para representação de um modelo equivalente para grandes sistemas de potência. Simulações envolvendo menos elementos elétricos em comparação com o sistema original reduzem os esforços computacionais sem reduzir a precisão, caso o modelo seja bem deduzido e

adequado ao problema. Diante disso, é proposta no método a divisão do sistema em duas zonas: uma zona de estudo, em que a parte do sistema onde ocorre o fenômeno eletromagnético a ser analisado é representada de forma integral, e uma zona externa representada por modelos equivalentes reduzidos dos trechos do sistema nas zonas mais remotas àquela onde ocorre o fenômeno a ser avaliado, sendo estes modelos baseados na resposta em frequência do sistema. Para a aplicação deste método se faz necessário recalcular os parâmetros do modelo correspondente de cada zona sempre que se desejar avaliar as condições em diferentes trechos do sistema.

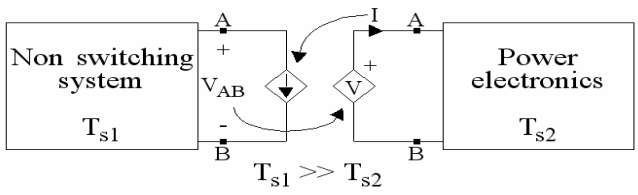
3.1.2 LE-HUY *ET AL.* (2006)

Foi proposta uma alternativa a simulações de sistemas que possuem chaveamentos de alta frequência. Devido às plataformas de simulação comerciais, que existiam até então, muitas vezes possuírem passos de cálculo superiores ao tempo de chaveamento de alguns circuitos, tornando propensa a ocorrência de erros numéricos nos resultados, propôs-se a utilização de dispositivos FPGA para obtenção de passos de cálculo menores.

A metodologia utilizada neste trabalho foi dividir a simulação em duas plataformas com passos de cálculo diferentes, utilizando um dispositivo FPGA para simulação dos modelos que contém chaveamentos com um passo de cálculo menor, e os demais modelos em uma plataforma convencional, de acordo com o ilustrado na Figura 7.

Como são utilizadas plataformas de simulação distintas deve haver a transferência de informações entre os dispositivos, que é realizado por meio de fontes dependentes de dados calculados na outra plataforma. Um problema que pode existir está na possibilidade de ocorrência de eventuais delays neste procedimento de transferência de dados, podendo comprometer resultados no caso de estarem acima de limites toleráveis.

Figura 7. Sistema desacoplado em dois subsistemas para simulação em plataformas distintas.



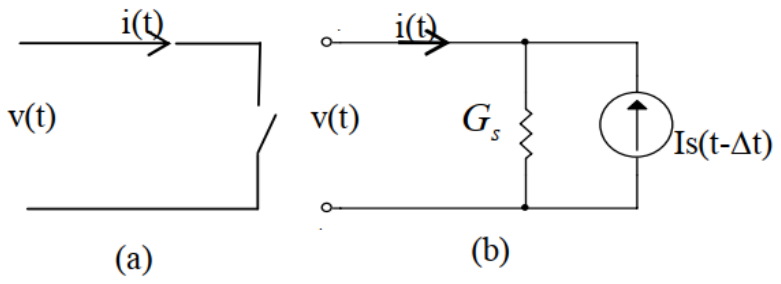
Fonte: Le-Huy et al. (2006).

3.1.3 MATAR & IRAVANI (2007)

Propuseram, assim como Le Huy et al. (2006), a utilização de dispositivos FPGA para simulação de transitórios em sistemas com chaveamentos de alta frequência, a fim de se obter passos de cálculo menores que simuladores ofereciam. Para tal foi proposta uma metodologia semelhante à de Matar et al. (2004), utilizando módulos de processamento operando em paralelo para cálculo das tensões dos nós e outros módulos para atualização das fontes de corrente, sendo executados em um único ciclo de clock.

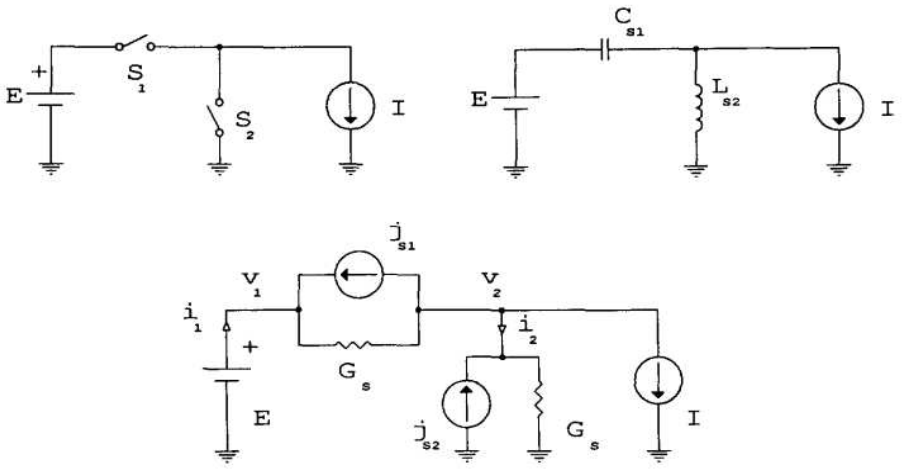
A principal contribuição deste trabalho foi a elaboração de um modelo de chave que represente este elemento, independentemente de estar aberta ou fechada, conforme ilustrado na Figura 8. Foi proposto um modelo que utiliza uma admitância fixa, para qualquer estado, com uma fonte de corrente paralela cujo valor vai estar relacionado com o estado da chave, mantendo assim a matriz admitância do sistema sem modificações durante chaveamentos. Justificou-se a escolha deste modelo por ser possível eliminar riscos de ocorrência de oscilações numéricas em comparação com outros modelos, como o de uma representação L/C, utilizado pelo RTDS (OU et al., 2014), cujo modelo é apresentado na Figura 9. Constatou-se ainda que a determinação do valor exato da admitância fixa não é uma tarefa simples, tanto para o modelo deste trabalho quanto para outros, pois não é um parâmetro real da chave, requerendo testes de estimativa e aproximação e de comparações com resultados de simulações de referência para otimização deste valor. No caso do modelo L/C a indutância ou capacitância pode ser estimada como sendo proporcional ao período de chaveamento e a admitância fixa também estimada (PEJOVIC & MAKSIMOVIC, 1994).

Figura 8. Modelo de admitância fixa para circuitos chaveados.



Fonte: Matar e Iravani (2007)

Figura 9. Modelo L/C para circuitos chaveados.

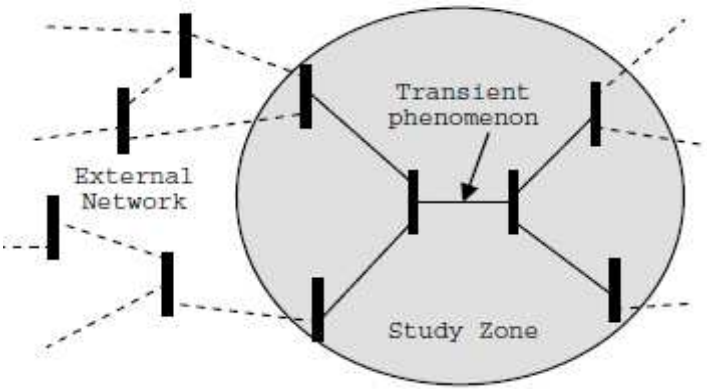


Fonte: Pejovic e Maksimovic (1994)

3.1.4 MATAR & IRAVANI (2009)

Propuseram uma otimização ao método proposto em Matar et al. (2004) para divisão de sistemas em uma zona externa e uma zona de estudo. Pretendeu-se, então, a obtenção de um modelo mais completo, para representação de toda a zona externa. Assim, dividiu-se esta zona externa em duas partes, de acordo com a distância dos elementos elétricos à zona de estudo. Para os elementos mais distantes da zona de estudo utilizou-se o mesmo modelo de zona externa apresentado em Matar et al. (2004). Já os elementos da zona externa na fronteira com a zona de estudo foram modelados por circuitos baseados na resposta em frequência com alocação de pólos e resíduos utilizando convolução recursiva. Um esquema relativo a divisão proposta é apresentado na Figura 10.

Figura 10. Divisão do sistema em uma zona de estudo e uma zona externa.



Fonte: Matar e Iravani (2009).

A metodologia da arquitetura de processamento foi organizada de forma similar aos dois trabalhos anteriores dos autores. Discutiu-se que seria possível executar também operações algébricas em paralelo, ou seja, não apenas entre módulos. Foi mostrado que o cálculo matricial realizado nas simulações era composto por equações formadas por soma de produtos, onde as multiplicações poderiam ser processadas simultaneamente em paralelo, armazenadas e em seguida realizadas as somas destes produtos.

3.1.5 CHEN & DINAVAHI (2009)

Foi proposta a utilização de um dispositivo FPGA para simulações de transitórios em sistemas elétricos, com uma arquitetura similar a dos trabalhos anteriores, tendo módulos de cálculo para tensões nos nós e para atualização das fontes de corrente. Diferencia-se com relação ao armazenamento dos dados, onde foi utilizado memórias RAM, e não registradores simples como é padrão. Uma das contribuições apresentadas é a conexão do dispositivo a um computador, para utilização de dados externos, gerados pelo software ATP, como dados iniciais de entrada para a simulação. Este trabalho também se diferenciou por utilizar aritmética de ponto flutuante no processamento dos dados.

3.1.6 CHEN & DINAVAHI (2012)

Propuseram a execução de simulações de sistemas de maior porte que os estudados em Chen & Dinavahi (2009). Foi utilizado, ainda, o modelo equivalente de divisão em zonas proposto por Matar & Iravani (2009) para redução do número de equações, combinado com a utilização de múltiplos dispositivos FPGA, devidamente sincronizados, com cada FPGA simulando partes do sistema completo.

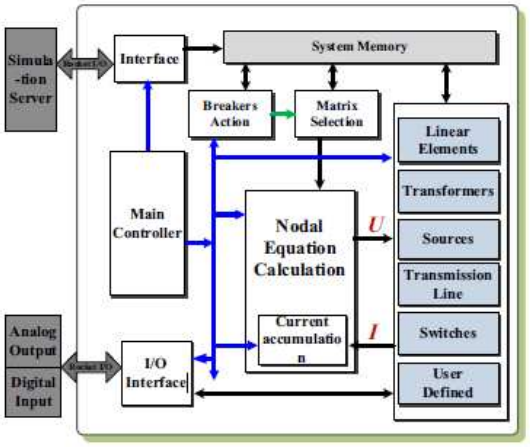
Foi proposta uma modularização funcional, onde em cada dispositivo são implementados módulos que processem equações referentes a um mesmo tipo de componente elétrico (um dispositivo só com módulos para processamento de equações de linhas, outro só com módulos para equações de transformadores, assim por diante), com exceção de elementos como resistores, indutores e capacitores, que são descritos juntos em um mesmo módulo. Assim como em Chen & Dinavahi (2009) o processamento dos dados foi realizado em aritmética de ponto flutuante. Também foi

mencionada a necessidade de conectar e sincronizar corretamente os dispositivos utilizados a fim de evitar problemas na transmissão de dados.

3.1.7 ZHANG ET AL. (2015)

Apresentaram também uma proposta para simulação de sistemas de maior porte, porém utilizando apenas um dispositivo FPGA. Neste trabalho foi mostrado que é possível organizar uma arquitetura que permita a simulação de um sistema de até no máximo 74 barras em um chip Virtex-7. A arquitetura apresentada consiste de uma unidade principal de cálculo de equações nodais, segmentado em cálculo das tensões e atualização de fontes de corrente, de forma similar a trabalhos anteriores, e de unidades periféricas de controle e condicionamento de parâmetros a serem utilizados nos cálculos. Um esquema desta arquitetura é apresentado na Figura 11.

Figura 11. Arquitetura do simulador tempo real em FPGA.



Fonte: Zhang et al. (2015).

Foi utilizada uma matriz de admitância variável de acordo com o estado das chaves do sistema, diferentemente de modelos equivalentes usados em outros trabalhos, justificada pela ideia de avaliar condições mais extremas. Para o processamento das equações utilizou-se aritmética de ponto flutuante de precisão dupla.

3.1.8 RAZZAGHI (2016)

A arquitetura proposta para o simulador em FPGA possui uma unidade de processamento semelhante à dos demais trabalhos discutidos. Também apresenta outra

unidade que recebe e armazena dados pré processados de parâmetros do sistema, gerados em um computador utilizando LabVIEW, sendo esta unidade responsável por realizar a seleção dos dados que são necessários para a simulação. Optou-se por utilizar aritmética de ponto fixo no cálculo das equações, justificada pela obtenção de maior desempenho no hardware nesta configuração. Como forma de validação foi testada, de forma satisfatória, a condição de hardware in the loop do simulador, utilizando-o como gerador de sinal PWM para o controle de um inversor trifásico real.

3.1.9 ZHU ET AL. (2016)

É proposta a execução de simulações de sistemas envolvendo STATCOMS (Static Synchronous Compesator) (KUMAR et al., 2003) formadas por pontes em cascata, desenvolvendo-se uma arquitetura para associação de múltiplos dispositivos FPGA com esta finalidade. Foi apresentada a necessidade de se ter uma relação mestre-escravo entre os dispositivos para fins de controle e sincronização. Foi testado um sistema com 5 pontes, sendo cada uma processada em um dispositivo FPGA, comparando-se com a resposta de um sistema real. Optou-se por utilizar os dois tipos de aritmética a nível de operações, processando multiplicações em ponto flutuante e acumulações em ponto fixo.

3.1.10 LI ET AL. (2017)

Propõem o uso de múltiplos dispositivos FPGA para o desenvolvimento de uma arquitetura a ser aplicada a simulações de pequenas redes de geração distribuída. Esta arquitetura consiste de módulos de processamento das equações, módulos de controle e módulos de comunicação, implementados em aritmética de ponto flutuante. O procedimento proposto de comunicação entre dispositivos permite configurar estruturas de múltiplos dispositivos em loop, em cadeia e estruturas radiais. É proposto também um mecanismo de sincronização entre os clocks de cada dispositivo. Foram utilizados para testes quatro FPGAs conectados em estrutura radial para simulação de um sistema contendo dois painéis fotovoltaicos, um painel com bateria e a rede de distribuição, organizados em uma placa processando as equações da rede, em outra placa as equações do painel com bateria e as outras duas processando as equações dos outros dois painéis.

3.1.11 SÍNTESE DAS REFERÊNCIAS

Diante do que foi exposto até então, pôde-se concluir que:

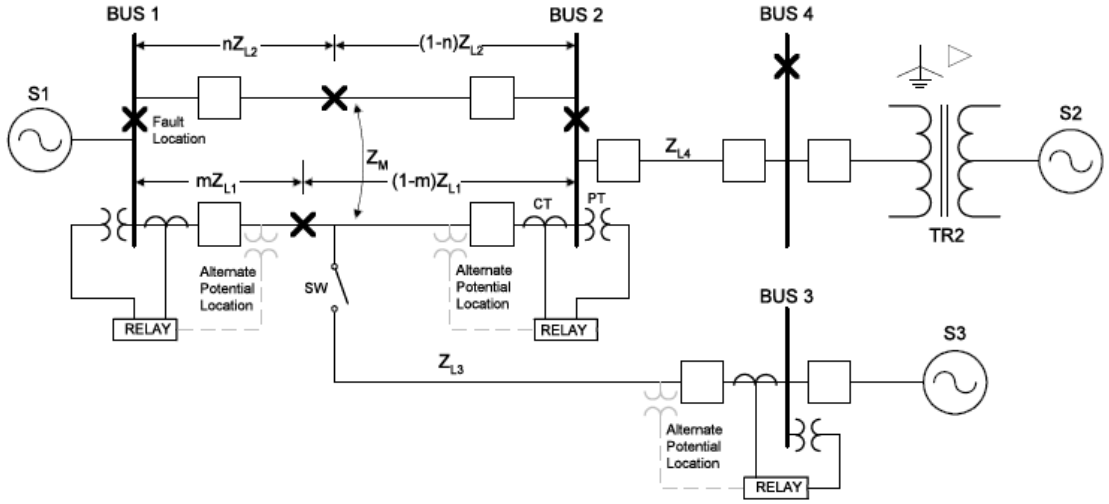
- A temática envolvendo simulações em tempo real implementadas em FPGA ainda é incipiente e vem progredindo diante das pesquisas desenvolvidas durante os últimos anos. Contribuições podem ser realizadas tanto por meio da otimização do que vem sendo trabalhado até então quanto pelo desenvolvimento de novas metodologias para as simulações.
- Todos os trabalhos apresentam um algoritmo padrão de solução: formação da matriz admitância nodal, posteriormente processamento do cálculo das tensões e em seguida a atualização das fontes de corrente. Nestes três passos sequenciais são exploradas a capacidade de processamento paralelo em todos os níveis possíveis.
- Apresentam também distinções entre detalhes na metodologia de implementação como o tipo de aritmética utilizada, a representação do sistema por inteiro ou por circuitos equivalentes a utilização de um ou mais de um dispositivo na simulação e, eventualmente, dispositivos auxiliares ao processamento em FPGA. Cada um desses detalhes influencia diretamente no tempo computacional da simulação.

Assim propõe-se nesta dissertação englobar algumas dessas características, elaborando uma plataforma que represente integralmente os sistemas, realizando parte das simulações em múltiplos dispositivos conectados, utilizando dados provenientes de uma unidade de processamento externa para cálculo da matriz admitância inversa. Pretendeu-se também realizar estimativas de capacidade do hardware quanto ao tamanho dos registradores de armazenamento das variáveis intermediárias ao cálculo das equações dos modelos e da capacidade total de processamento de uma das placas utilizadas neste trabalho, avaliando-se quanto cada modelo consome de recursos durante seu processamento.

3.2 MODELOS DE REFERÊNCIA PARA SIMULAÇÃO DE SISTEMAS DE TRANSMISSÃO

A fim de se ter um sistema de teste mais completo para testes de validação do método proposto nesta pesquisa foi escolhido o sistema de testes proposto pelo IEEE Power System Relaying Committee (2004), apresentado da Figura 12. Embora não seja o real propósito deste trabalho obter simulações envolvendo especificamente sistemas de proteção, este sistema apresenta parâmetros suficientes para servir como teste de validação da proposta de implementação, tendo em vista que se constitui um sistema de transmissão contendo os elementos elétricos mais relevantes a este tipo de análise.

Figura 12. Sistema de referências para testes de proteção.



Fonte: IEEE Power System Relaying Committee (2004).

Diante disto, o sistema que é mostrado na integra na figura anterior é adaptado para os propósitos do trabalho, sendo considerados os trechos de linha entre as barras 1, 2 e 4, além do gerador S1 e do transformador T1, sendo conectado a este uma carga indutiva L1 representando a máquina S2, desconsiderando-se o trecho de linha que vai até a barra 3 e os elementos relativos a proteção. Assim o sistema adaptado é ilustrado na Figura 13.

Figura 13. Sistema de referência adaptado.

Fonte: Adaptado de Lopes, F. (2015).

Também são indicados os modelos específicos de cada elemento do sistema, aplicáveis às simulações. Com relação às linhas de transmissão são apontados alguns modelos para simulações de transitórios, e destes é selecionado o modelo já apresentado no capítulo anterior para uma linha sem perdas.

Para os geradores e máquinas são apontados modelos detalhados de máquinas síncronas, incluindo-se resistência de armadura, reatâncias de eixo direto e em quadratura, reatâncias transitórias e sub transitórias, dentre outras. Para este trabalho decidiu-se substituir o modelo completo de máquina por uma carga indutiva, visto que o objetivo é possibilitar a execução de simulações em tempo real em uma plataforma ainda em desenvolvimento, optando-se por modelos menos complexos, desde os de linhas até os modelos de carga.

Para o transformador são indicados modelos de simulação considerando o equipamento como sendo ideal ou real. Sendo assim foi selecionado um modelo de transformador ideal em série com resistências tanto no lado de alta, como no lado de baixa tensão, representado as perdas nestes enrolamentos.

4 CONSTRUÇÃO DO AMBIENTE DE SIMULAÇÃO

Definiu-se que seriam implementadas simulações em tempo real referentes a propagação de sinais eletromagnéticos em sistemas de transmissão contendo elementos mais comuns de um sistema deste tipo, como geradores, linhas de transmissão, transformadores, dentre outros, avaliando-se, inicialmente, o comportamento individual de cada modelo, e finalizando-se com um estudo de caso de um sistema de transmissão padrão, sendo selecionado um modelo do EMTP referenciado em IEEE Power System Relaying Committee (2004), apresentado no capítulo anterior. Assim apresenta-se neste capítulo aspectos relativos a construção dos módulos de descrição dos modelos destes elementos elétricos e dos módulos de ferramentas de auxílio as simulações, compondo o ambiente de simulação proposto, implementado em FPGA.

Também foi estabelecida uma sistemática de simulação, aplicada a cada cenário avaliado, seja de um modelo individual ou de um sistema composto dos modelos implementados, definida pelos seguintes passos:

1. Simulação dos modelos no ATP, utilizando-se componentes implementados por rotinas desenvolvidas na própria construção do software, como referência para os resultados na demais plataformas;
2. Simulação dos modelos no PSIM, escrevendo rotinas em C para execução de DLLs implementadas em aritmética de ponto flutuante e de ponto fixo;
3. Simulação dos modelos no ambiente construído em FPGA em aritmética de ponto fixo.

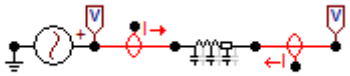
4.1 SIMULAÇÕES OFFLINE

Devido a reconhecida confiabilidade nos resultados fornecidos em simulações nos softwares PSIM e ATP, estes foram selecionados para produzir resultados com a finalidade de servir como parâmetros de comparação com os resultados gerados nas simulações executadas em tempo real. A escolha do ATP é justificada também por ser um dos softwares de simulação mais completos para estudos de transitórios

eletromagnéticos, sendo tomado, então, como referência para o desenvolvimento de um novo tipo de plataforma de simulação.

As simulações efetuadas no ATP foram realizadas utilizando a ferramenta ATPDraw, variando-se os sistemas analisados e suas condições por meio da variação tanto de parâmetros quanto dos próprios componentes elétricos fornecidos pelo software. Na Figura 14 é apresentada a configuração dos componentes para efetuar um teste de energização de linha de transmissão em aberto.

Figura 14. Modelo de simulação para energização de uma linha de transmissão monofásica no ATPDraw.

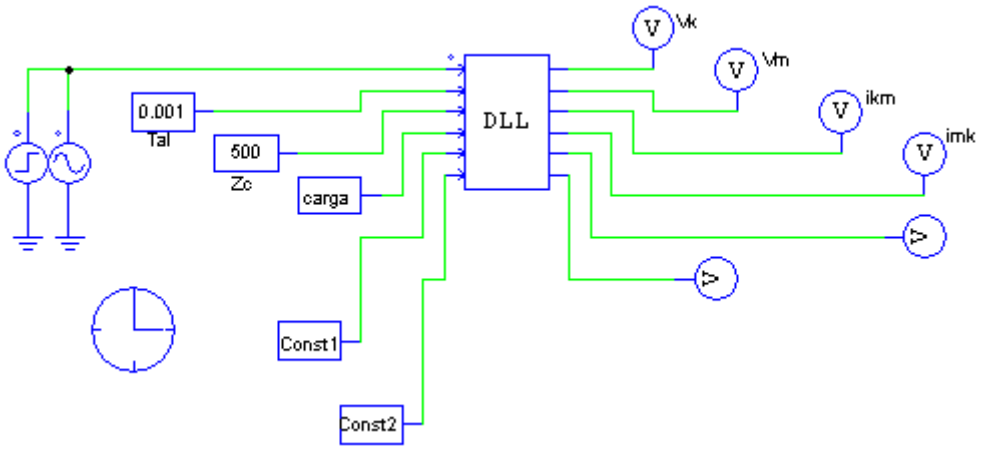


Fonte: Produzido pelo autor, via ATP.

Já o PSIM, embora não seja comumente aplicável a estudos de redes elétricas, foi selecionado a fim de se ter um passo intermediário para a transcrição dos modelos implementados em aritmética de ponto flutuante em modelos implementados em aritmética de ponto fixo. Para isso são utilizadas DLLs, ferramentas computacionais descritas no capítulo 2, podendo realizar o processamento nos dois tipos de aritmética e serem comparados os resultados entre ambas, a fim de se ter um parâmetro inicial a respeito da consistência da representação em ponto fixo adotada. Além disso, a implementação de ponto fixo no PSIM tem um papel importante na avaliação dos resultados simulados em tempo real, em termos de ser o parâmetro de comparação com o formato de implementação mais próximo ao que se tem no FPGA.

Na Figura 15 é apresentado o diagrama esquemático para simulações no PSIM, contendo um sistema com mesma configuração do sistema simulado na Figura 14, no ATP. A DLL implementada representa a linha a ser energizada, contendo o mesmo modelo de linhas que é utilizado pelo ATP, apresentado na Figura 2. Observa-se que, pelo fato do modelo ter sido construído e não utilizado um componente já pronto, os parâmetros deste modelo se tornam variáveis de entrada.

Figura 15. Modelo de simulação para energização de uma linha de transmissão monofásica no PSIM.



Fonte: Produzido pelo autor, via PSIM.

4.2 AMBIENTE DE SIMULAÇÃO EM FPGA

Para a implementação de um ambiente de simulação em FPGA é necessário, primeiramente, a construção de módulos escritos em Verilog a fim de constituir a estrutura que irá descrever para o hardware o modo de sintetização para o processamento da simulação, e em seguida realizar a compilação desta estrutura de descrição com o descarregamento no hardware destas ligações lógicas necessárias para configurá-lo conforme é pretendido.

A definição de sintetização, mencionada neste trabalho, é dada por este procedimento de compilação da estrutura de descrição e descarregamento na placa. Uma analogia que pode ser feita para uma melhor compreensão do termo é que a sintetização vai configurar no FPGA um mecanismo semelhante a uma estrutura construída manualmente utilizando circuitos integrados de eletrônica digital, com elementos de lógica combinacional e sequencial e fios de interligação, sendo esta estrutura responsável por constituir operadores aritméticos para o processamento das equações descritas nos módulos de simulação.

Outra definição importante, já mencionada no Capítulo 1, é referente ao conceito que é utilizado neste trabalho para o termo ambiente de simulação, o qual é assumido como um setup de testes em que se processa a simulação cujos dados gerados podem ser transmitidos para meios externos, possibilitando também variações das condições do sistema avaliado mediante o uso de chaves seletoras disponíveis nas placas FPGA. Neste trabalho foi construído um ambiente de simulação contendo modelos de linhas

monofásicas e trifásicas, modelo de transformador, gerador de sinais e mecanismos de seleção de passo de cálculo e de transmissão de dados para visualização em uma interface gráfica e comunicação entre placas.

Para a criação dos módulos escritos em Verilog e sintetização dos circuitos a serem configurados foi utilizado o Quartus II. Neste software, além da criação, é realizada a associação entre os módulos e a sintetização é efetuada durante a compilação de toda a estrutura de descrição implementada. Finalizada a compilação é ativada a função do Quartus que descarrega na placa o arquivo que a configura para realizar as ligações dos blocos lógicos a fim de reproduzir fisicamente a estrutura definida na sintetização.

Em relação ao uso de Verilog são reforçados os conceitos da linguagem que foram mais aplicados para a implementação dos módulos. Em termos de variáveis a maior parte delas foi definida como sendo dos tipos *reg* ou *wire.* As variáveis do tipo *reg* são registradores de tamanho variável definido pelo usuário e se comportam como uma memória, mantendo seu valor até que se tenha uma instrução dentro da estrutura de descrição que indique a modificação deste valor. Podem ser associadas, dentro dos módulos ao qual estão inseridas, a variáveis *input* e *output* que são as responsáveis por receber os dados de entrada e armazenar as saídas de cada um dos módulos. Já as variáveis do tipo *wire* se comportam como fios de associação entre entradas e saídas de módulos e, no geral, devem possuir um número de bits equivalente ao número definido para as variáveis *input* e *output* (THOMAS e MOORBY, 2002; PALNITKAR, 2003).

Para o processamento das equações são definidas variáveis intermediárias cujos dados são armazenados em registradores, e estas são modificadas dentro de blocos denominados *always*. Esta estrutura possui uma lista de sensibilidade em que as operações ou equações contidas dentro deste bloco são processadas mediante variação nesta lista de sensibilidade. Neste projeto utilizou-se a variação síncrona em que o *always* é executado na borda de subida de um clock de frequência proporcional ao passo de cálculo definido para a as simulações. Outras estruturas de sintaxe utilizadas dentro de blocos *always* são laços aplicados de acordo com a lógica considerada para solução das equações. São utilizadas estruturas *if/else*, *for* e *case*, com sintaxe próxima a linguagem C, diferenciando-se pela substituição das chaves de início e fim do laço por comandos denominados *begin* e *end* ou *endcase*, para o *case* (THOMAS e MOORBY, 2002; PALNITKAR, 2003).

A etapa de construção da estrutura de descrição para implementação do ambiente de simulação é notadamente a mais complexa e requer uma análise mais minuciosa, apresentada nas subseções seguintes. São apontados alguns aspectos referentes a construção dos módulos de descrição dos modelos e de mecanismos para sincronização e transmissão de dados. O hardware utilizado durante este trabalho foi placas DE2 com chip Cyclone II e DE2-115 com chip Cyclone IV, ambas da Altera.

4.2.1 Geração de Sinais e Visualização dos Dados Gerados

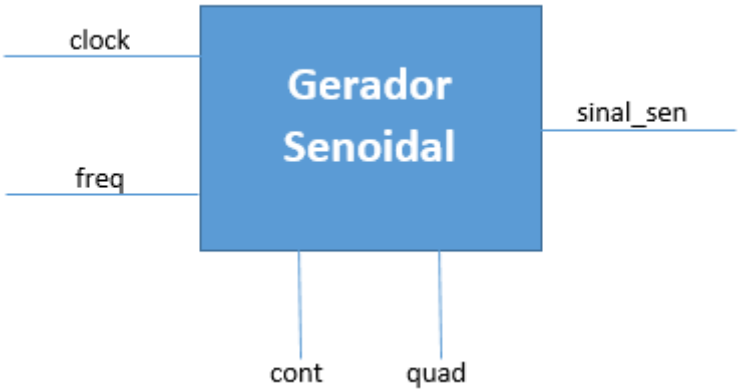
Duas funcionalidades muito importantes presentes nos simuladores utilizados nesta pesquisa não são tão simples de serem executadas em uma simulação efetuada em dispositivos FPGA, comparado com outros simuladores. Uma delas é com relação à aplicação de sinais de entrada como fonte de tensão aos componentes que representam os elementos elétricos. Nos simuladores comerciais há um bloco já pronto que basta ser acoplado ao circuito e fornecer o tipo de sinal desejado. No dispositivo FPGA é necessário realizar uma geração digital destas formas de onda. Para este trabalho foi desenvolvida a geração digital de um sinal senoidal. Neste sentido utilizou-se para a construção deste sinal um conjunto de registradores atuando como look-up table armazenando valores de seno representados em ponto fixo para ângulos de 0 a 90 graus, calculando o seno dos ângulos não presentes na tabela por meio de propriedades trigonométricas e técnicas de interpolação. Também é utilizada uma estrutura construída com *case* para a seleção da frequência do sinal utilizando chaves seletoras da placa FPGA. Por meio de um contador de sincronização e de um registrador que define o quadrante para uso das propriedades trigonométricas, foi possível construir sinais defasados de 120 graus entre si. O valor deste contador é proporcional ao ângulo do sinal no passo de cálculo vigente, sendo os valores iniciais do contador e do quadrante responsáveis por definir o ângulo do primeiro ponto calculado, e por consequência a fase do sinal. O valor inicial do contador é calculado por meio da expressão:

$$cont = \frac{\left|180 - |ang\_fase|\right|}{90 \cdot 4f \cdot \Delta t} \tag{51}$$

Em que ang_fase é o ângulo inicial (0°, 120° ou -120°), f é a frequência e Δt o passo de cálculo. Dentro do algoritmo o quadrante muda de valor quando o contador atinge limiares que indicam um ângulo de transição para esta mudança de quadrante e seu valor inicial é definido de acordo com o ângulo inicial da fase representada. O algoritmo

utilizado no módulo de descrição para uma das fases está apresentado no Anexo A. Na Figura 16 é apresentado um esquema com as entradas, saídas e variáveis de seleção da fase do gerador senoidal.

Figura 16. Diagrama do gerador senoidal.



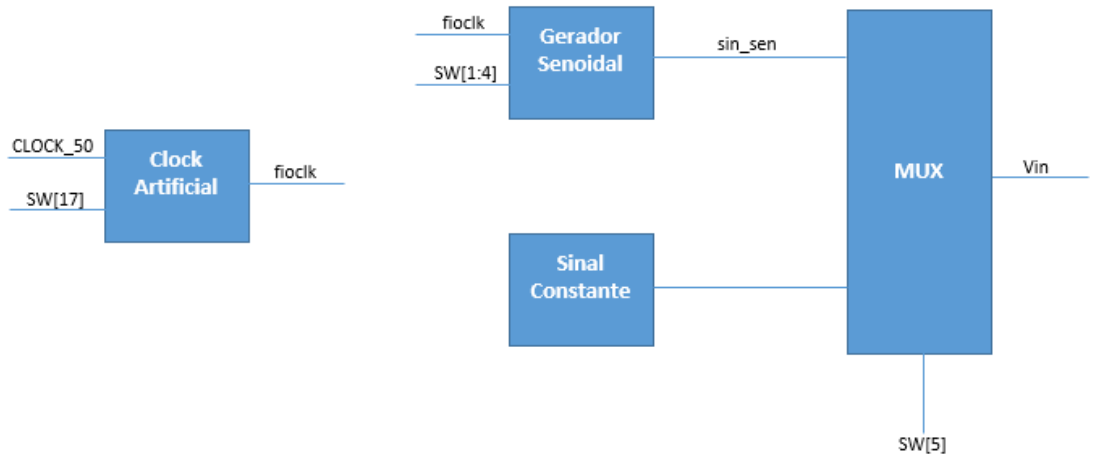
Fonte: Produzido pelo autor.

Os módulos de geração digital de sinal senoidal foram englobados em módulos denominados *Tensao_EntA, Tensao_EntB, Tensao_EntC*. Eles propiciam, além de sinais gerados em três fases defasadas de 120º, funcionando como um modelo de gerador trifásico ideal, a ativação de um sinal constante, cuja amplitude, assim como a do sinal senoidal, pode ser selecionada pelo usuário do ambiente de simulação por meio de uma chave seletora do dispositivo FPGA. No caso do sinal senoidal o valor do seno já é gerado em ponto fixo ao contrário do sinal constante que deve passar pelo procedimento descrito na equação (50). Conforme esta equação o valor da amplitude é normalizado, dividindo-o por um valor base, em ambos os tipos de sinais. A estrutura de descrição destes módulos de seleção do tipo de sinal é apresentada no Anexo B.

O passo de cálculo destes e dos demais módulos utilizados para implementar modelos de elementos elétricos é obtido por meio de um outro módulo, denominado *clock,* que fornece um clock alternativo aos que são nativos do dispositivo, de 50 MHz e de 27 MHz. A frequência deste clock é selecionada de modo que seja proporcional a um passo de cálculo genérico, aplicável a qualquer simulação em geral, sendo de 1 µs especificamente para este trabalho. O clock artificial foi construído a partir do de 50 MHz, possuindo uma chave seletora que, estando em nível lógico alto, ativa o processo de cálculo para geração deste clock. Está presente também no Anexo B o algoritmo de descrição deste clock artificial.

Na Figura 17 são apresentados diagramas relacionando entradas e saídas para os módulos de seleção do tipo de sinal e do clock artificial.

Figura 17. Diagrama esquemático do seletor de tipo de sinal e do clock artificial.



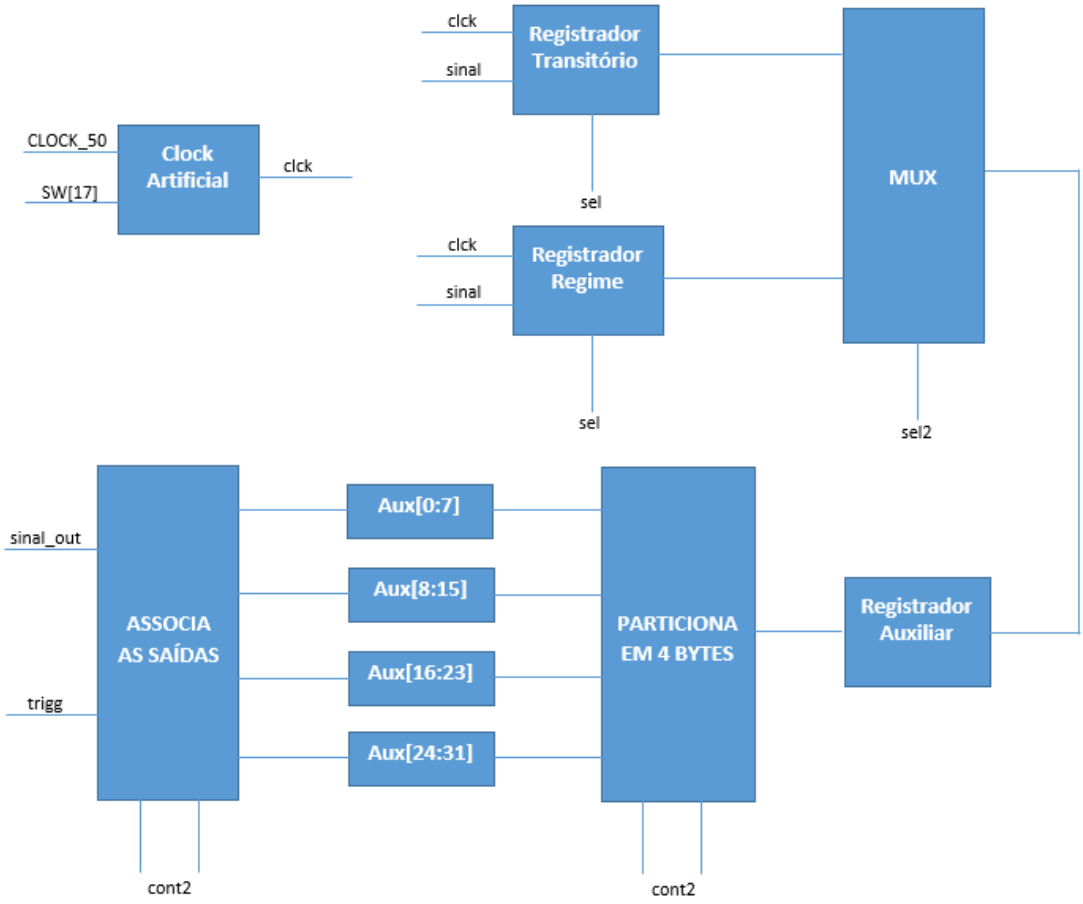
Fonte: Produzido pelo autor.

Outra funcionalidade necessária é a interface gráfica para observação dos pontos calculados, algo que é intrínseco a simuladores comerciais, como no caso do ATP e do PSIM por exemplo. Com relação ao FPGA é necessário que se transporte os dados obtidos para serem apresentados em um programa de computador ou para um osciloscópio. Neste trabalho optou-se pela execução da primeira opção. Para isso é necessário estabelecer uma conexão por meio da porta serial RS-232 do dispositivo FPGA com a de um computador. Por meio do programa Siow, que monitora a entrada e saída de dados da porta serial, são impressos em tela todos os dados que são transferidos do FPGA para o computador de forma contínua, visto que a simulação não cessa até que seja dado um comando para tal. Estes dados são impressos no formato hexadecimal.

Em termos de descrição de módulos, dois destes são utilizados para realizar o condicionamento dos dados para o processo de transmissão. Um dos módulos, denominado *serial,* realiza a coleta de 200 amostras representando um intervalo de tempo correspondente ao primeiro período de uma senóide de 60 Hz (transitório inicial) ou a trechos quaisquer da curva (regime permanente) com mesmo período, de aproximadamente 16,67 ms, selecionado por uma chave da placa. Os dados de regime transitório e de regime permanente são armazenados em conjuntos distintos de 200 registradores de 32 bits, cada. A quantidade de amostras foi definida dentro dos limites de capacidade de transmissão da porta serial que é de 115200 bits por segundo, ou 240

amostras por ciclo de 60 Hz, no máximo. Na Figura 18 é apresentado um esquema do funcionamento do módulo *serial.*

Figura 18. Diagrama esquemático do módulo serial.



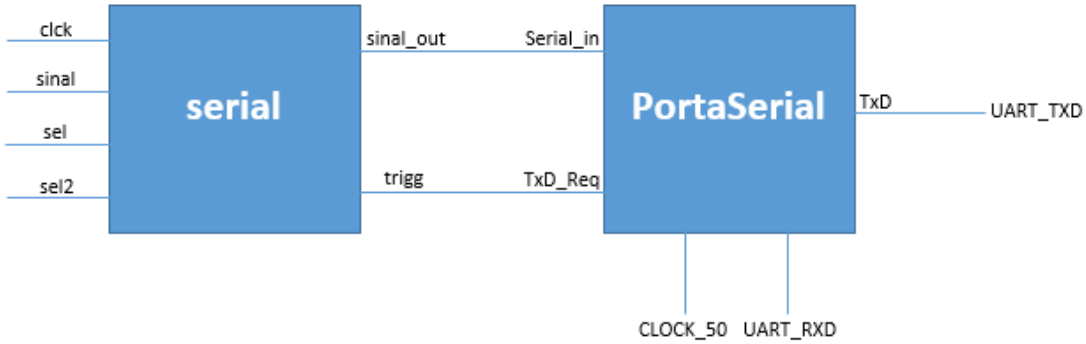
Fonte: Produzido pelo autor.

Para o processo de transmissão foi construído um outro clock artificial de frequência equivalente a taxa de amostragem definida, obtido novamente a partir do clock de 50 MHz. A transmissão, na prática, é realizada por meio de um outro módulo denominado *PortaSerial,* que recebe 1 byte por ciclo de clock artificial e transfere serialmente um bit por ciclo de clock de 50 MHz. Devido a esta condição o módulo *serial* também é responsável por enviar parcelas de cada amostra particionada em bytes ao outro módulo. O conteúdo dos registradores de armazenamento das amostras, por possuírem 32 bits, são associados a um registrador auxiliar e particionados em 4 bytes durante quatro ciclos, enviando assim 1 byte por ciclo do clock artificial para o módulo *PortaSerial*. A transferência dos dados do módulo *serial* para o módulo *PortaSerial* é

ativada mediante seleção de nível lógico alto de uma das chaves da placa FPGA. A estrutura de descrição dos dois módulos é apresentada no Anexo C.

O funcionamento do módulo *PortaSerial* é basicamente receber as parcelas de 1 byte das amostras e processar a lógica responsável por emitir um bit por ciclo. Dentro dessa lógica que pode ser vista no Anexo C, as amostras recebidas pelo módulo são associadas com bits de controle e é fornecido o caminho para a conexão física entre a placa e o meio de transmissão (cabo ou fio) através da saída UART_TXD do FPGA. Esta lógica de transmissão, entretanto, só é ativada mediante um sinal de trigger emitido pelo módulo *serial* junto com dados de 1 byte. Com este sinal ativado os dados passam a ser imediatamente impressos na tela do programa Siow. Na Figura 19 é mostrada a associação do módulo *serial* com o módulo *PortaSerial*.

Figura 19. Associação do módulo serial com o módulo PortaSerial.



Fonte: Produzido pelo autor.

4.2.2 Linhas de Transmissão e Transformadores

Nas simulações dos modelos de linhas de transmissão utilizou-se inicialmente placas DE2 com chip Cyclone II, funcionando para todos os testes realizados na aplicação de modelos monofásicos. A ideia inicial era utilizar apenas estas placas, porém o módulo para representação de linha trifásicas acarretou problemas de limitação de recursos ao ser instanciado mais de uma vez. Sendo assim se tornou necessária a utilização de um dispositivo de maior recurso e então optou-se por utilizar placas DE2-115 com chip Cyclone IV para implementar as simulações propostas neste trabalho.

Para descrição destes modelos de linha foram escritos dois módulos, um para representação de linhas monofásicas, denominado *Linhas,* e o outro para representação do modelo de linhas trifásicas, denominado *Linhas_Tri.* Um modelo de transformador

trifásico foi implementado no módulo denominado *Transformador*. Embora pouco aplicáveis na prática, optou-se por avaliar inicialmente sistemas monofásicos por possuírem modelos mais simples e posteriormente avançar com a avaliação de sistemas trifásicos, facilitando a identificação e correção de inconsistências nas implementações comum a ambos.

Tanto para as simulações do modelo monofásico quanto as do modelo trifásico variou-se cenários de avaliação modificando-se a quantidade de trechos instanciados, o sinal de entrada (senoidal ou degrau) por meio de uma chave seletora, conforme apresentado na Figura 14, e a carga acoplada no terminal receptor de cada linha, também por meio de uma chave seletora associada a uma lógica combinacional de seleção de dois possíveis estados, desenvolvida dentro dos próprios módulos de linhas: seleção de uma carga resistiva ou sistema em aberto (sem carga).

Alguns aspectos relativos à construção dos módulos de sintetização dos modelos de linhas e de transformadores valem ser destacados, pois podem se tornar fontes de erro nas simulações. Observou-se durante a descrição dos modelos a necessidade de uma maior atenção para a definição do tamanho dos registradores de armazenamento das variáveis intermediárias calculadas nos módulos. Percebeu-se que em equações em que há mais de uma operação os resultados parciais do cálculo da equação vão sendo armazenados durante o processamento no próprio registrador que armazena o resultado final. Uma escolha inadequada pode implicar em estouro de armazenamento e consequentemente erro nos cálculos se propagando indefinidamente a partir deste erro. Assim se torna necessário registradores capazes de armazenar o maior valor possível gerado por esses passos intermediários. Foram estimados, inicialmente de forma empírica, alguns valores para estes registradores e constatou-se que guardavam uma relação com a representação em ponto fixo adotada, na qual registradores com o dobro de bits desta representação se mostraram apropriados e atenderam tanto aos testes realizados quanto as demais simulações, não apresentarando problemas de estouro desta natureza.

Outro ponto que vale se ressaltar é com relação a precedência de operações aritméticas dentro de equações com mais de uma operação, especialmente quando há divisões. Deve-se escrever a ordem das operações de forma que um passo intermediário não ocasione a divisão de um numerador menor que um denominador implicando em um resultado parcial nulo. Um exemplo disso é no processamento da equação (50), em que ao invés de se dividir primeiro o valor da grandeza pelo valor base, gerando um

valor nulo por default do FPGA, realiza-se primeiro a multiplicação do valor da grandeza pelo formato, dividindo, em seguida, o resultado pelo valor base, conforme a equação (52).

$$N_{PFIXO} = \frac{N_{PFLUT} \cdot 2^B}{Base} \tag{52}$$

Também foi observado que na representação de circuitos abertos por meio de uma alta impedância, o software Quartus II não representava de forma correta o valor no formato decimal, a partir de valores de tamanhos superiores a 32 bits se tornando necessária representação do dado em formato binário.

Especificamente em relação aos sistemas trifásicos o equacionamento que o descreve apresenta matrizes que devem ser invertidas para o cálculo de tensões pelo método clássico adotado. Diferentemente do processamento em aritmética de ponto flutuante a inversão de matrizes em aritmética de ponto fixo, além de ser um processo não trivial, consome uma quantidade considerável de memória. Com os recursos disponíveis seria possível realizar este procedimento processando no próprio dispositivo FPGA, porém devido as restrições de placas disponíveis optou-se por alocar esses recursos dedicando-os ao processamento das equações dos modelos de elementos elétricos. Assim o cálculo das matrizes inversas é efetuado externamente ao ambiente construído em FPGA para alguns valores pré-definidos de carga, utilizando-se do Matlab, e os resultados são armazenados em um conjunto de registradores habilitados para receber estes dados.

Em termos de processamento das equações dos modelos, é utilizado o clock artificial de 1 MHz (1 μs) descrito na seção anterior para executar um bloco a*lways,* onde estão contidas estas equações, a cada subida deste clock. No módulo *Linhas* é processado o algoritmo do qual algumas equações são apresentadas a seguir e toda a estrutura de descrição do módulo pode ser vista no Anexo D.

$$V_{kpfixo} = Tens\_in \tag{53}$$

$$V_{mpfixo} = Z_{pfixo}(I_k[0] + I_{klinha}) \tag{54}$$

$$intermed1 = (Z_{base} \cdot (fat2 - 1)/Z_C) \cdot V_{kpfixo}/fat2 \tag{55}$$

$$intermed2 = (Z_{base} \cdot (fat2 - 1)/Z_C) \cdot V_{mpfixo}/fat2 \tag{56}$$

$$i_{kpfixo} = intermed1 - I_k[0] \tag{57}$$

$$i_{mpfixo} = intermed2 - I_m[0] \tag{58}$$

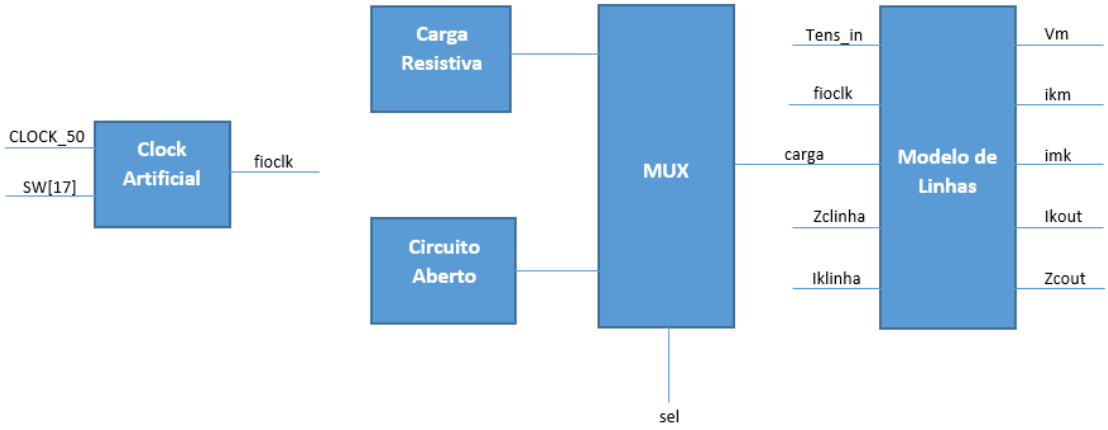
$$I_k[1000] = intermed2 + I_m[0] \tag{59}$$

$$I_m[1000] = intermed1 + I_k[0] \quad (60)$$

As variáveis I_k e $I_m$são janelas móveis onde o conteúdo de cada registrador é transferido para o registrador de índice anterior a cada passo de cálculo por meio de uma estrutura *for*. O tamanho da janela, ou seja, quantos registradores deverão haver para este processo, é definido pela razão entre o tempo de trânsito da linha e o passo de cálculo da simulação. Nas equações (59) e (60) as janelas móveis são de tamanho 1000, para um tempo de trânsito de 1 ms.

Na Figura 20 é apresentado um diagrama ilustrativo do funcionamento do módulo *Linhas.*

Figura 20. Diagrama esquemático do módulo Linhas.



Fonte: Produzido pelo autor.

No módulo *Linhas_Tri* a estrutura de descrição contida dentro do a*lways* é semelhante a forma de implementação do módulo *Linhas* com a inclusão das transformações modais que converte o sistema em modos desacoplados entre si e propicia a solução ser realizada como três sistemas monofásicos independentes, como é mostrado nas equações a seguir, retiradas do algoritmo completo apresentado no Anexo D.

$$int_a = I_m[0] + I_{klinha\_a} \quad (61)$$

$$int_b = I_m[1] + I_{klinha\_b} \quad (62)$$

$$int_c = I_m[2] + I_{klinha\_c} \quad (63)$$

$$V_m[0] = (Z[0][0] \cdot (int\_a) + Z[0][1] \cdot (int\_b) + Z[0][2] \cdot (int\_c))/fat2 \quad (64)$$

$$V_m[1] = (Z[1][0] \cdot (int\_a) + Z[1][1] \cdot (int\_b) + Z[1][2] \cdot (int\_c))/fat2 \quad (65)$$

$$V_m[2] = (Z[2][0] \cdot (int\_a) + Z[2][1] \cdot (int\_b) + Z[2][2] \cdot (int\_c))/fat2 \quad (66)$$

$$V_{mmod}[0] = V_m[0] + V_m[1] + V_m[2]/3 \quad (67)$$

$$V_{mmod}[1] = V_m[0] - V_m[1]/3 \quad (68)$$

$$V_{mmod}[2] = V_m[0] - V_m[2]/3 \quad (69)$$

$$intermed[0] = (Z_{base} \cdot ((fat2) - 1)/Z0) \quad (70)$$

$$intermed[1] = (Z_{base} \cdot ((fat2) - 1)/Z1) \quad (71)$$

$$intermed[2] = (Z_{base} \cdot ((fat2) - 1)/Z2) \quad (72)$$

$$J_{km}[0] = intermed[0] \cdot V_{kmod}[0]/(fat2) - J_{k0}[0] \quad (73)$$

$$J_{km}[1] = intermed[1] \cdot V_{kmod}[1]/(fat2) - J_{k1}[0] \quad (74)$$

$$J_{km}[2] = intermed[2] \cdot V_{kmod}[2]/(fat2) - J_{k2}[0] \quad (75)$$

$$J_{mk}[0] = intermed[0] \cdot V_{mmod}[0]/(fat2) - J_{m0}[0] \quad (76)$$

$$J_{mk}[1] = intermed[1] \cdot V_{mmod}[1]/(fat2) - J_{m1}[0] \quad (77)$$

$$J_{mk}[2] = intermed[2] \cdot V_{mmod}[2]/(fat2) - J_{m2}[0] \quad (78)$$

$$J_{k0}[2000] = intermed[0] \cdot V_{mmod}[0]/(fat2) + J_{mk}[0] \quad (79)$$

$$J_{k1}[1000] = intermed[1] \cdot V_{mmod}[1]/(fat2) + J_{mk}[1] \quad (80)$$

$$J_{k2}[1000] = intermed[2] \cdot V_{mmod}[2]/(fat2) + J_{mk}[2] \quad (81)$$

$$J_{m0}[2000] = intermed[0] \cdot V_{kmod}[0]/(fat2) + J_{km}[0] \quad (82)$$

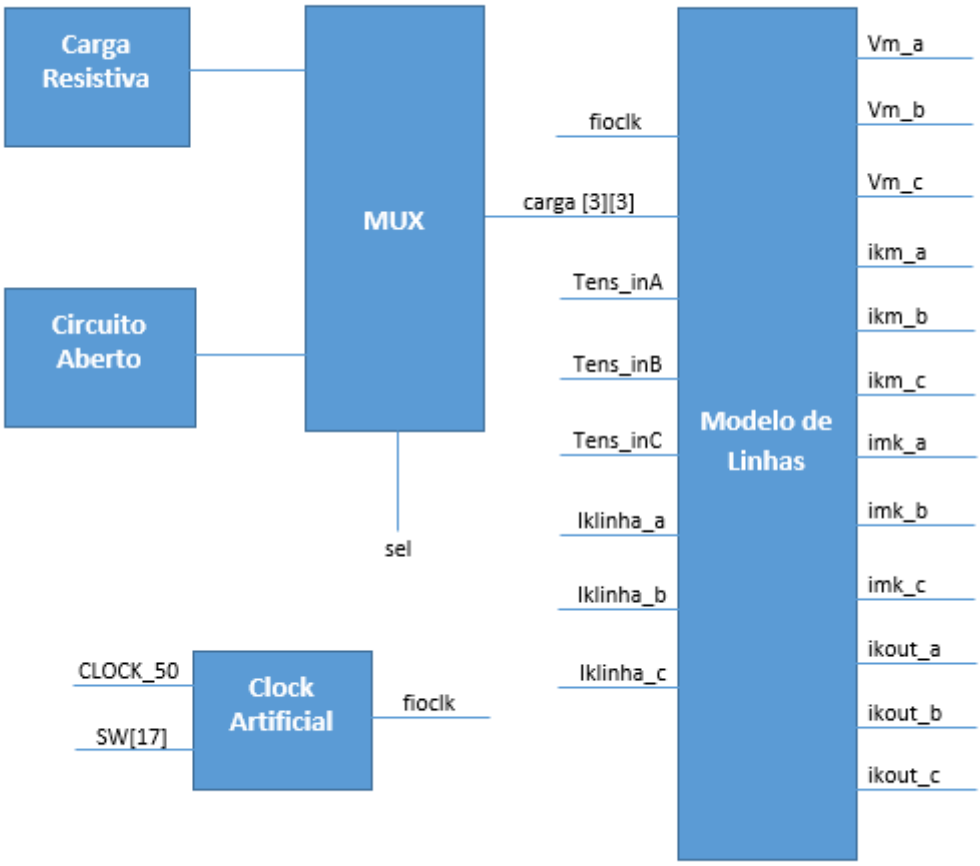
$$J_{m1}[1000] = intermed[1] \cdot V_{kmod}[1]/(fat2) + J_{km}[1] \quad (83)$$

$$J_{m2}[1000] = intermed[2] \cdot V_{kmod}[2]/(fat2) + J_{km}[2] \quad (84)$$

Em que os índices 0, 1 e 2 caracterizam os 3 modos nas variáveis calculadas da equação (61) a equação (78) e os índices 2000 e 1000 da equação (79) a equação (84) caracterizam o tamanho da janela móvel relacionados aos tempos de transito de 2 ms no modo zero e 1 ms nos modos positivo e negativo (1 e 2). O diagrama referente ao funcionamento do módulo é exibido na Figura 21.

Um modelo de transformador trifásico foi implementado no módulo denominado *Transformador*. Foi definida a utilização de um modelo em Y-Y aterrado para aplicação deste elemento na simulação do sistema de testes apresentado no capítulo anterior e referenciado em IEEE Power System Relaying Committee (2004). Dentro do bloco *always* do módulo são calculadas as tensões e correntes do lado secundário mediante dados do lado primário fornecidos como variáveis de entrada do módulo. O algoritmo utilizado neste processo de cálculo é apresentado no Anexo D. Na Figura 22 é apresentado o diagrama com as entradas e saídas definidas no módulo.

Figura 21. Diagrama esquemático do módulo Linhas_Tri.



Fonte: Produzido pelo autor.

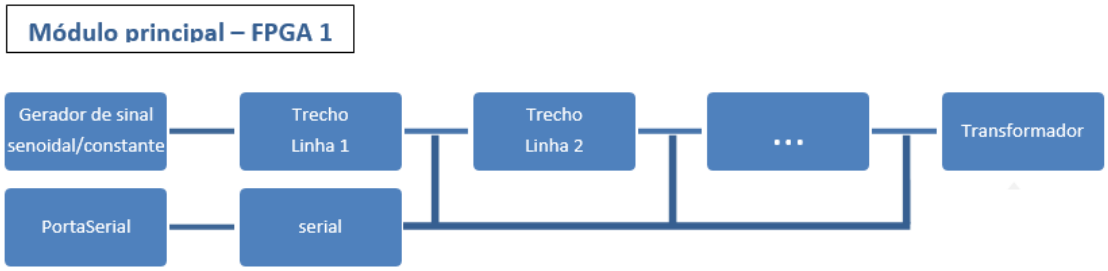
Figura 22. Diagrama do módulo Transformador.

Fonte: Produzido pelo autor.

Para a instanciação dos módulos a serem utilizados em cada simulação é definido um módulo principal denominado *Mod_Teste* que realiza a associação entre eles por meio de variáveis *wire,* sendo também responsável por associar os dados a

serem transmitidos aos pinos de conexão com meios externos à placa. No *Mod_Teste* os módulos também são associados as chaves seletoras e ao clock nativo da placa de 50 MHz. Um diagrama esquemático pode ser visto na Figura 23, onde é apresentada a ideia de como os módulos de representação de modelos e de condicionamento de dados são associados no *Mod_Teste*. No anexo E é exibido o algoritmo escrito para descrição deste módulo.

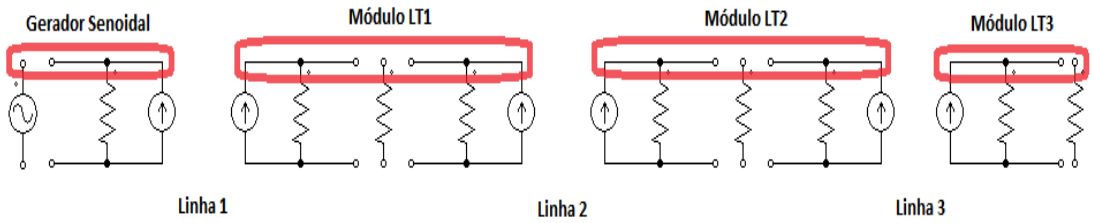
Figura 23. Diagrama esquemático do módulo principal.



Fonte: Produzido pelo autor.

Na prática a instanciação de cada trecho de linha faz com que haja conexões em cascata do modelo apresentado na Figura 2 do Capítulo 2. Em termos de funcionamento dos módulos há um processo paralelo de cálculo das tensões em cada um dos nós formados pela junção do terminal receptor de uma das linhas com o terminal emissor da outra que está em conexão, além de uma carga acoplada a este nó. A Figura 24 ilustra este processo.

Figura 24. Processo de instanciação de linhas em cascata.



Fonte: Produzido pelo autor.

4.2.3 Capacidade de Uma Placa FPGA

Dispositivos FPGA são produzidos comercialmente por diferentes fabricantes e dentro de mesmos fabricantes há diferentes modelos de placas e de chips, implicando em alternativas de escolha definidas pelo compromisso entre o custo (quanto mais

robusta for a placa maior o custo) e a aplicação que vai ser dada ao FPGA (estimativa inicial de quanto as aplicações consomem de recursos). Dentre os dispositivos utilizados neste trabalho avaliou-se a capacidade total de processamento da placa DE2-115 com chip Cyclone IV e como são alocados recursos oferecidos pelo FPGA para operações aritméticas, equações de modelos e o quão os módulos que vão sendo instanciados consome desses recursos.

Na Tabela 1 são apresentados dados de quantidades de elementos lógicos, pinos, bits de memória e multiplicadores embarcados, que definem a capacidade total de armazenamento de dados e de processamento para o FPGA avaliado.

Tabela 1. Capacidade total de processamento de um dispositivo FPGA – Cyclone IV.

Total de elementos lógicos	114480
Total de pinos	529
Total de bits de memória	3981312
Total de multiplicadores embarcados	532

Fonte: Quartus II.

Em todas as simulações efetuadas observou-se que a quantidade de pinos utilizada está relacionada com associações a variáveis do tipo *output* e não refletem diretamente ao consumo de recursos de memória do dispositivo. Diante disto foram avaliados para operações aritméticas o consumo de elementos lógicos, bits de memória e número de registradores, conforme registrado na Tabela 2 em números absolutos e em percentuais relacionados ao total indicado na Tabela 1. Para isso foi construído um módulo que tem como entrada dois operandos de tamanho máximo de 16 bits e dentro do módulo é processada apenas a operação específica, sendo armazenada em um registrador também de 16 bits.

Tabela 2. Consumo de recursos para uma placa FPGA em operações aritméticas elementares.

Operação	Elementos lógicos	Registradores	Bits de memória
Adição/Subtração	519 (< 1%)	224	5628 (< 1%)
Multiplicação	528 (<1%)	224	5628 (< 1%)
Divisão	462 (< 1%)	191	402 (< 1%)

Fonte: Autor, via Quartus II

Percebeu-se que a quantidade de bits de memória, bem como a de registradores utilizados nos módulos de condicionamento dos dados para transmissão serial varia de acordo com o tamanho dos operandos e também do resultado final da operação aritmética, tendo sido observado também este comportamento em testes com operandos com quantidade de bits maiores e menores que 16. Com isso comprovou-se que o tamanho das variáveis de registro tem influência no consumo de recursos de um dispositivo FPGA, assim como as operações aritméticas presentes nas equações a serem processadas também estão relacionadas diretamente a este consumo da capacidade do dispositivo.

Também foi avaliada a execução de equações genéricas combinando operações elementares, representando algumas das equações presentes nos módulos de sintetização do ambiente de simulação. Para esta verificação adaptou-se o módulo construído para as operações elementares de forma que recebesse mais de dois operandos e processasse combinações destas operações, de acordo com o apresentado na Tabela 3.

Tabela 3. Consumo de recursos para uma placa FPGA em equações com múltiplas operações elementares

Equação	Elementos lógicos	Registradores	Bits de memória
a·(b+c)	528 (< 1%)	224	5628 (<1%)
a·b + c·d + e·f	535 (< 1%)	226	6030 (< 1%)
a·b·c/d	573 (<1%)	240	8844 (< 1%)
a·b/c - d	551 (< 1%)	228	6432 (< 1%)

Fonte: Autor, via Quartus II

Observou-se condições semelhantes ao fato descrito anteriormente em relação a influência da quantidade de bits dos operandos e do resultado final no número total de registradores e de bits de memória utilizados. Além disso constatou-se que a quantidade de elementos lógicos usados no processamento de apenas uma equação é próxima, sendo um pouco maior, à execução de uma única operação elementar. Pressupôs-se que o FPGA reutiliza circuitos lógicos que são construídos para processar determinadas operações e os readapta, sendo válida uma análise futura mais minuciosa a fim de atestar esta hipótese. Outra observação é feita no caso das equações específicas nos módulos do ambiente de simulação, cujos operandos são, para uma parte considerável destas equações, variáveis a cada passo de cálculo. Isto implica em maior consumo dos

elementos lógicos, conforme pode ser verificado na Tabela 4, a qual apresenta o que é consumo que é dispendido para a implementação das funcionalidades do ambiente de simulação: gerador senoidal, clock artificial, transmissão serial para interface gráfica e mecanismo de comunicação entre múltiplos dispositivos.

Tabela 4. Consumo de recursos para uma placa FPGA com as funcionalidades do ambiente de simulação.

Total de elementos lógicos	8666 (8%)
Total de registradores	333
Total de bits de memória	12864 (< 1%)
Total de multiplicadores embarcados	18 (3%)

Fonte: Autor, via Quartus II

Outro recurso que não vinha sendo utilizado pelo FPGA durante as análises anteriores foram os multiplicadores embarcados, passando a ser usados, por definições intrínsecas ao processo de sintetização do Quartus, em módulos nos quais os operandos tendem a variar a cada passo de cálculo.

Após estes levantamentos foram incluídos nas simulações os modelos de linhas e do transformador e foi realizada a avaliação do consumo de recursos ocasionado nestes cenários. Na Tabela 5 é apresentado o consumo determinado pela instanciação de um a quatro trechos de linhas monofásicas. Vale ressaltar que as funcionalidades necessárias ao ambiente de simulação suguem inclusas nestas simulações e há uma composição entre o consumo delas e o consumo individual de cada instancia dos módulos com os modelos de linhas. Isto explica-se, pois, o consumo apresentado pelo Quartus não reflete a realidade quando as entradas e saídas do módulo não estão associadas a entradas e saídas do módulo principal. Assim se torna impraticável avaliar apenas um módulo individualmente.

Da Tabela 5 verificou-se que a partir da simulação com instanciação de duas linhas o aumento da quantidade de recursos consumidos segue uma tendência que se aproxima de uma progressão aritmética, e em alguns casos até se iguala. Observou-se que a instanciação de uma única linha consome, no geral, mais recursos que a inclusão de linhas adicionais, algo semelhante ao discutido para equações genéricas compostas de operações elementares, onde se notou que a sintetização é capaz de otimizar o consumo reutilizando elementos utilizados em instanciações semelhantes. Na Tabela 6 são apresentadas as estimativas de consumo considerando apenas o modelo de linhas

monofásicas, calculadas subtraindo o consumo apresentado na Tabela 4 para a primeira linha e considerando, para a segunda linha e as demais, uma razão aproximada de progressão aritmética em cada um dos itens da tabela.

Tabela 5. Consumo de recursos para uma placa FPGA com módulos de linhas monofásicas.

Instâncias	Elementos lógicos	Registradores	Bits de memória	Multiplicadores embarcados
1 Linha	9324 (8%)	527	106441 (3%)	40 (8%)
2 Linhas	9823 (9%)	633	200064 (5%)	56 (11%)
3 Linhas	10520 (9%)	782	293656 (7%)	72 (14%)
4 Linhas	11154 (10%)	931	387248 (10%)	88 (17%)

Fonte: Autor, via Quartus II

Tabela 6. Estimativa de consumo de recursos para modelo de linha monofásica.

Instâncias	Elementos lógicos	Registradores	Bits de memória	Multiplicadores embarcados
1ª Linha	658 (< 1%)	194	93757 (2%)	22 (4%)
2ª Linha e demais	697 (< 1%)	149	93592 (2%)	16 (3%)

Fonte: Autor, via Quartus II

Em termos de capacidade total do FPGA percebe-se que os multiplicadores embarcados tendem a ser os primeiros elementos do dispositivo a atingir seu limite conforme vai se aumentando o número de instanciações. Nesta tendência observada projeta-se a chegada a este limite com 27 instanciações do módulo de linhas monofásicas.

Na Tabela 7 são apresentadas as mesmas observações feitas na Tabela 5, sendo agora considerados sistemas trifásicos com instanciações do módulo com o modelo de linha e do módulo com o modelo de transformador. Também estão inclusas nas simulações as funcionalidades do ambiente de simulação, conforme já explanado, havendo uma composição entre o consumo delas (Tabela 4) e o consumo individual de cada instancia dos módulos.

Tabela 7. Consumo de recursos para uma placa FPGA simulando modelos de sistemas trifásicos.

Instâncias	Elementos lógicos	Registradores	Bits de memória	Multiplicadores embarcados
1 Linha	31716 (28%)	1009	435751 (11%)	150 (28%)
2 Linhas em série	39004 (34%)	2050	858446 (22%)	246 (46%)
3 Linhas em série	46254 (40%)	3112	1281141 (32%)	342 (64%)
4 Linhas em série	53441 (47%)	4153	1703836 (43%)	438 (82%)
1 Transformador	9130 (8%)	365	12864 (< 1%)	17 (3%)
Sistema do IEEE	51928 (45%)	3959	616464 (15%)	531(100%)

Fonte: Autor, via Quartus II

Percebe-se claramente que o algoritmo de solução para as equações de modelos trifásicos consome uma quantidade de recursos bem maior e seu crescimento se aproxima mais de um comportamento exponencial, em comparação com as simulações envolvendo modelos monofásicos, que de um comportamento linear proporcional a 3 linhas monofásicas. A complexidade maior deste algoritmo e a necessidade de calcular grandezas em dois domínios (fase e modal) contribui para se necessite de mais alta capacidade de processamento dentro do hardware. Em relação a instanciação dos módulos de modelos trifásicos observou-se que foi novamente seguida a tendência de aproximação a uma progressão aritmética ao se aumentar o número de trechos do sistema e da instancia do primeiro trecho consumir mais recursos que os demais, conforme a estimativa apresentada na Tabela 8, calculada seguindo o mesmo procedimento usado para a construção da Tabela 6.

Tabela 8. Estimativa de consumo de recursos para sistemas trifásicos.

Instâncias	Elementos lógicos	Registradores	Bits de memória	Multiplicadores embarcados
1ª Linha	23050 (20%)	676	422887 (11%)	132 (25%)
2ª Linha e demais	7288 (6%)	1041	422695 (11%)	96 (18%)

Fonte: Autor, via Quartus II

A respeito da capacidade total do FPGA, estas estimativas indicam novamente que os multiplicadores embarcados são os primeiros a atingir seu limite conforme vai se

aumentando o número de instancias dos módulos. Percebeu-se que foi atingida sua capacidade máxima na simulação do sistema do IEEE Power System Relaying Committee (2004) e estima-se que uma simulação com 5 linhas trifásicas em série extrapole esta capacidade.

4.2.4 ASSOCIAÇÕES ENTRE PLACAS FPGA

Além da associação entre módulos instanciados para processamento em uma mesma placa, os dispositivos FPGA possuem a capacidade de realizar a mesma simulação com placas distintas conectadas entre si. Esta possibilidade se torna relevante tendo em vista as observações da seção anterior em que as placas possuem limites de processamento, podendo se tornar necessário um aumento na capacidade total de representação dos sistemas.

Neste trabalho se dispunha de duas placas e foi realizada a conexão destas, embora seja possível realizá-la com quantos dispositivos se desejar ou se tiver disponível. Como conexão física entre os dois dispositivos foi utilizado um cabo IDE de 40 vias associando os pinos de GPIO (General-purpose input/output) de ambas as placas. A conexão física descrita pode ser observada na Figura 25.

Devido ao fato de ser possível transmitir apenas um bit por pino, construiu-se um modulo denominado *transmissao* que particiona os dados a serem enviados, de 32 bits, em conjuntos de 8 bits a serem associados com os pinos de GPIO pelo módulo principal, e transmitidos junto com bits de controle e de sincronização. Além disso, este módulo também recebe conjuntos de 8 bits transmitidos pela placa adjacente e reconstrói cada um dos dados de 32 bits mediante uma lógica definida para este processo, baseada em um código transferido junto com o dado (index_out) e um bit de habilitação da leitura deste dado (sinal). A transmissão é efetuada de forma bidirecional, ou seja, no mesmo ciclo de clock de 50 MHz dados são emitidos de forma simultânea de uma placa para outra. A reconstrução e envio dos dados recebidos para os módulos onde serão utilizados nos cálculos é realizada de forma paralela.

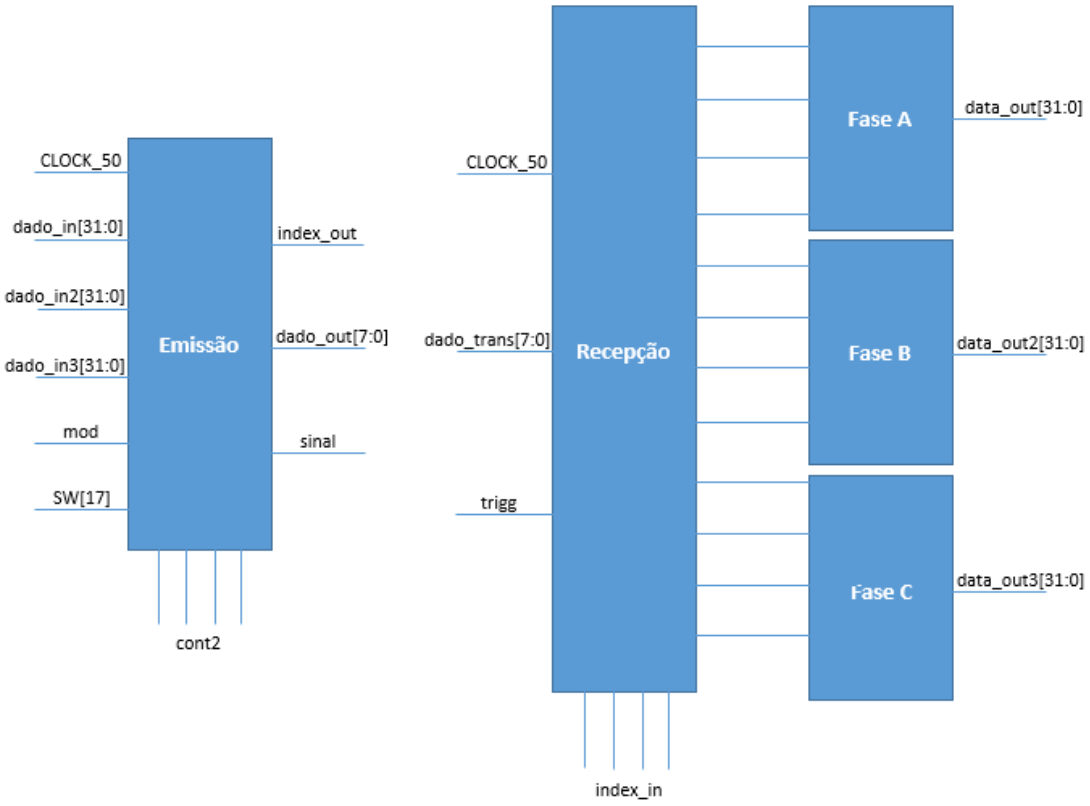
Figura 25. Dispositivos FPGA conectados via GPIO por cabo IDE.

Fonte: Autor.

Como as simulações propostas estão sendo realizadas em passos de cálculo de 1 µs, ou seja, a 1 MHz, e a transmissão é realizada enviando cada conjunto de 8 bits em uma frequência de 50 MHz, se dispõe de 50 ciclos, gastando-se 4 ciclos para enviar e receber dados de sistemas monofásicos (um dado de 32 bits emitido e um dado de 32 bits recebido) e 12 ciclos para enviar e receber dados de sistemas trifásicos (três dados de 32 bits emitidos e três dados de 32 bits recebidos). Para o funcionamento do módulo *transmissao* foi realizada uma adaptação no módulo do clock artificial de 1 MHz em que ele transmite um sinal denominado *mod,* o qual, possuindo valor zero, significa que está ocorrendo a borda de descida. Com isso, transmitindo na borda de descida, ao invés de se ter efetivamente 50 ciclos dispõe-se de 25 ciclos que ainda assim são suficientes, conforme já descrito. A escolha pela borda de descida se deu pelo fato dos cálculos estarem sendo processados na borda de subida e o envio de dados no mesmo instante de

tempo em que se processa os cálculos poderia se tornar uma fonte de erros, motivados por atrasos. O algoritmo completo de descrição do módulo *transmissao* está contido no Anexo F. Na Figura 26 é apresentado um diagrama esquemático expondo de forma ilustrativa o funcionamento deste módulo.

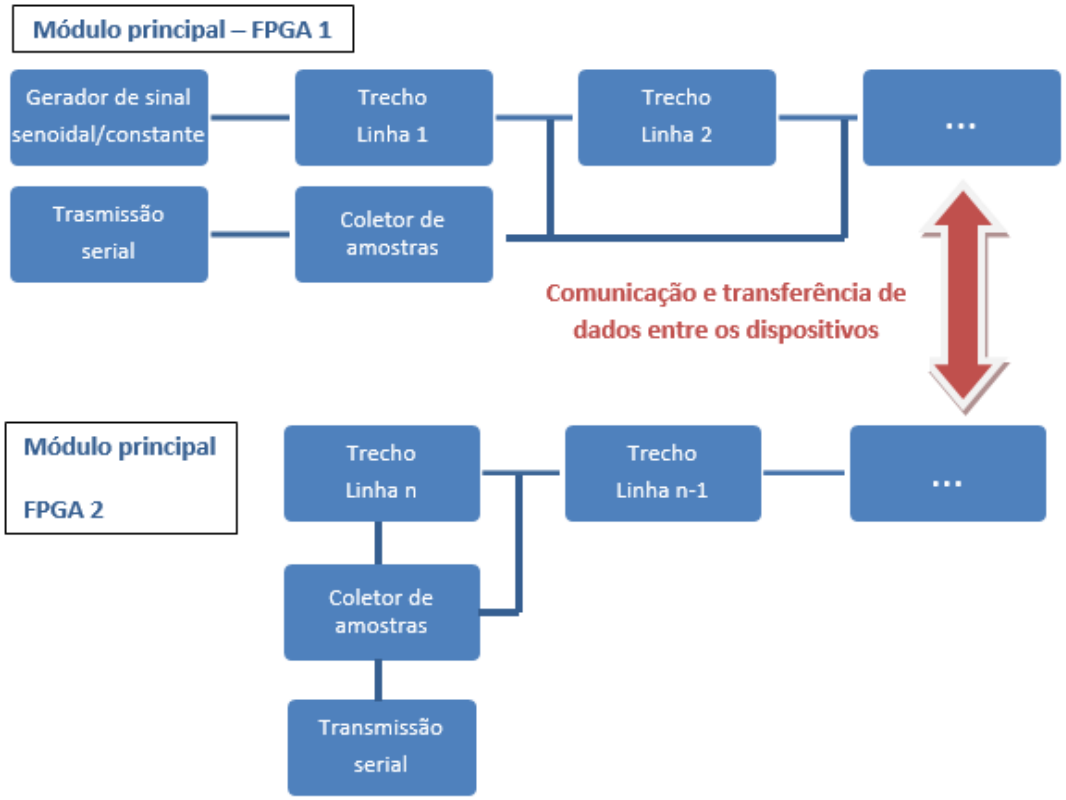
Figura 26. Diagrama esquemático do módulo transmissao.



Fonte: Produzido pelo autor.

Para avaliação deste mecanismo de transmissão foram simulados novamente sistemas contendo modelos de elementos elétricos monofásicos e trifásicos, com as mesmas possibilidades de variação de cenários apresentadas anteriormente nas simulações com uma só placa. No diagrama da Figura 27 é apresentado o esquema da instanciação de módulos no módulo principal de cada placa e a conexão entre os módulos principais para a configuração com duas placas interligadas.

Figura 27. Diagrama esquemático da segmentação das simulações em duas placas FPGA.

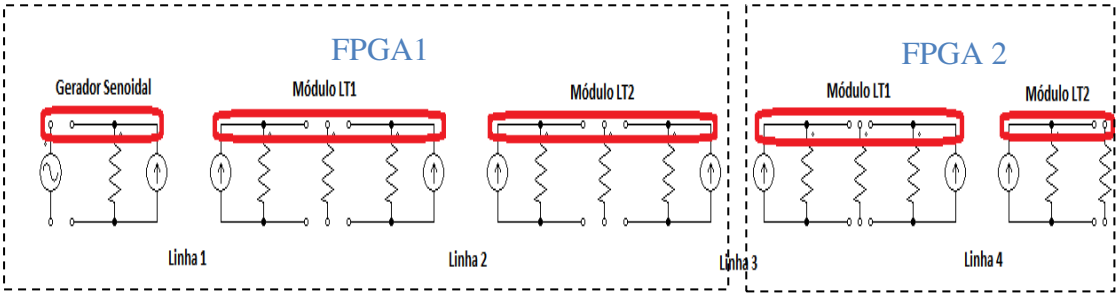


Fonte: Produzido pelo autor.

O barramento responsável pela junção de duas linhas onde o sistema foi particionado é o ponto no qual ocorre a troca bidirecional de informações, em que uma placa calcula e transmite para outra as tensões neste barramento e a segunda placa calcula e transmite para a primeira os dados correspondentes a fontes de tensão fictícias atualizadas. Esta transferência bidirecional é equivalente a que é executada por dois diferentes módulos em uma mesma placa, permitindo assim que módulos instanciados em dispositivos distintos operem da mesma forma como se pertencessem a um único dispositivo.

Na prática, em termos de processamento, a partição ocorre fazendo uso do desacoplamento entre os circuitos do modelo de linhas, de forma semelhante ao procedimento apresentado na Figura 24 para uma única placa. O funcionamento com o uso de duas placas é exemplificado na Figura 28.

Figura 28. Processo de instanciação de linhas simuladas em duas placas FPGA.



Fonte: Produzido pelo autor.

Durante a avaliação deste processo de transmissão observou-se inconsistências em algumas simulações realizadas na transferência de dados gerados pelos modelos de linhas trifásicas, processados com aplicação de um sinal senoidal, diferentemente da aplicação de um sinal constante em todas as fases das linhas que funcionou adequadamente. Essas inconsistências são constituídas de corrupções esporádicas nos dados que são entregues ao módulo de descrição do modelo de linhas, e que comprometem todo o restante da simulação a partir do instante em que o primeiro erro ocorre. Buscou-se realizar um rastreamento de onde está se iniciando o problema, porém com o passo de cálculo utilizado nas simulações, de 1 µs, se tem 1 milhão de amostras por segundo e não haveria armazenamento suficiente para transferência e observação dos dados na interface gráfica utilizada neste trabalho. Devido a esta adversidade ter surgido na fase final do trabalho não houve mais tempo hábil de busca de uma nova solução de rastreio. As simulações com modelos de linhas monofásicas foram executadas de forma satisfatória, em todos os cenários propostos e avaliados, e seus resultados, assim como as simulações de sinais constantes em linhas trifásicas, são apresentados no capítulo seguinte.

5 RESULTADOS E ANÁLISES

Na intenção de avaliar o ambiente de simulação projetado, foi simulado e observado o comportamento de tensões e correntes em sistemas contendo os modelos implementados, observando-se diferentes cenários com variações das condições de carregamento e do sinal de tensão aplicado ao sistema, comparando os dados gerados pela simulação em tempo real com simulações realizadas nas mesmas condições no ATP e no PSIM.

5.1 ESCOLHA DA REPRESENTAÇÃO EM PONTO FIXO

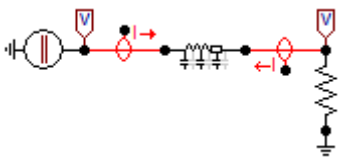
A fim de executar testes para análise da melhor representação em ponto fixo para as simulações a serem realizadas e de possíveis problemas de uma escolha inadequada, optou-se por avaliar um cenário de simulação contendo uma linha monofásica em aberto sendo aplicado um sinal constante, embora seja uma situação hipotética, porém importante para as conclusões obtidas na observação deste cenário. Na Tabela 9 são apresentados os parâmetros utilizados nas simulações e na Figura 29 é apresentado o diagrama unifilar do sistema simulado construído no ATPDraw.

Tabela 9. Parâmetros utilizados na simulação de uma linha monofásica.

Nível de tensão (pu)	1
Comprimento da linha (km)	300
Impedância característica (Ω)	500
Impedância terminal (GΩ)	1000

Fonte: Produzido pelo autor.

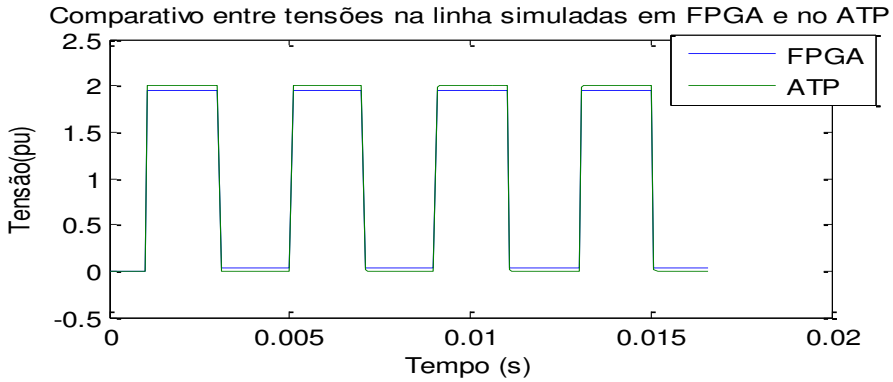
Figura 29. Diagrama unifilar representado uma linha monofásica em aberto.



Fonte: Produzido pelo autor, via ATP.

Inicialmente foi selecionada uma quantidade de 16 bits para representar dados de tensão e corrente em ponto fixo e realizou-se a simulação em FPGA com passo de cálculo de 1 µs. Os resultados foram comparados com uma simulação no ATP, em ponto flutuante, com passo de cálculo de 0,1 µs. Observa-se na Figura 30 que este número de bits escolhido não foi suficiente para minimizar a perda de informações durante o processamento em ponto fixo, ocasionando uma imprecisão considerável.

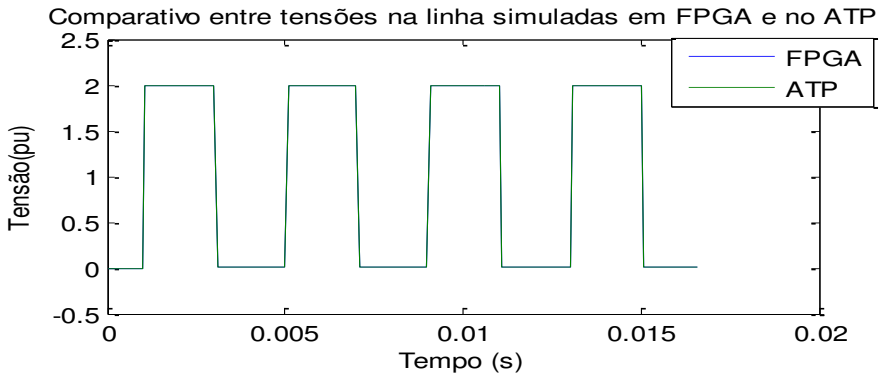
Figura 30. Tensão no terminal receptor de uma linha monofásica, com simulação em FPGA representada em 16 bits.



Fonte: Produzido pelo autor.

Partindo destes 16 bits foram realizados mais alguns testes com uma quantidade crescente, onde foi constatado que a partir da representação em 20 bits os resultados se tornavam mais satisfatórios e o erro apresentado na simulação implementada em 16 bits se tornava desprezível dentro deste cenário, conforme pode ser visto na Figura 31 para uma representação em 23 bits. Adotou-se a representação em 23 bits para as simulações posteriores, por ter sido considerado o valor mais conveniente dentro do compromisso entre exatidão e eficiência no consumo de recursos do dispositivo FPGA.

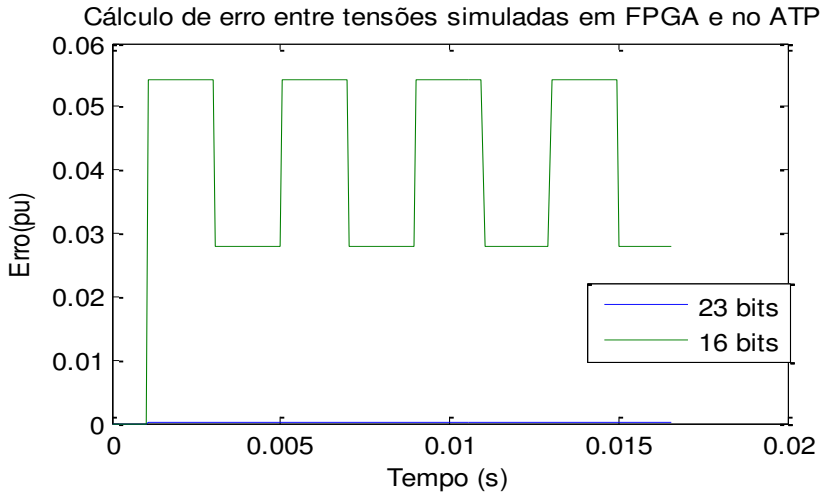
Figura 31. Tensão no terminal receptor de uma linha monofásica, com simulação em FPGA representada em 23 bits.



Fonte: Produzido pelo autor.

Um gráfico comparativo do erro entre as tensões para as duas representações avaliadas é apresentado na Figura 32.

Figura 32. Erro entre as tensões representadas em 16 bits e em 23 bits.



Fonte: Produzido pelo autor.

5.2 AVALIAÇÃO DOS MODELOS DE LINHAS

Para avaliação dos modelos de linhas implementados foi observado o comportamento individual de cada um deles, conforme mencionado no capítulo anterior, em pequenos sistemas contendo de um a quatro trechos de linha. Assim apresenta-se nesta seção os resultados para a simulação correspondente a energização de um sistema com 4 linhas monofásicas e de um sistema com 2 linhas trifásicas, realizadas com um passo de cálculo de 1 μs.

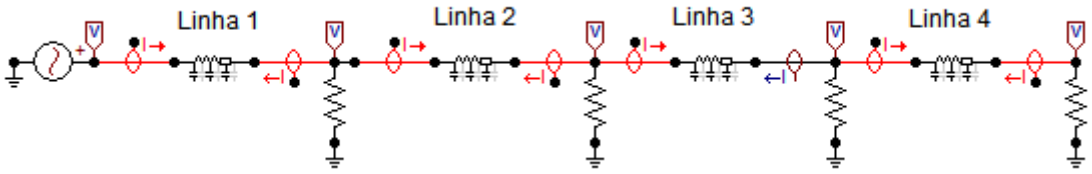
Na simulação do sistema monofásico as linhas são idênticas e conectadas em série cujos parâmetros são apresentados na Tabela 10. No terminal receptor de cada linha foi inserida uma carga resistiva, sendo de 2 kΩ nas linhas 1 e 3 e 1 kΩ nas linhas 2 e 4 conforme pode ser observado na Figura 33.

Tabela 10. Parâmetros utilizados na simulação de linhas monofásicas.

Nível de tensão (kV)	500
Comprimento da linha (km)	300
Impedância característica (Ω)	500
Tempo de trânsito (ms)	1

Fonte: Produzido pelo autor.

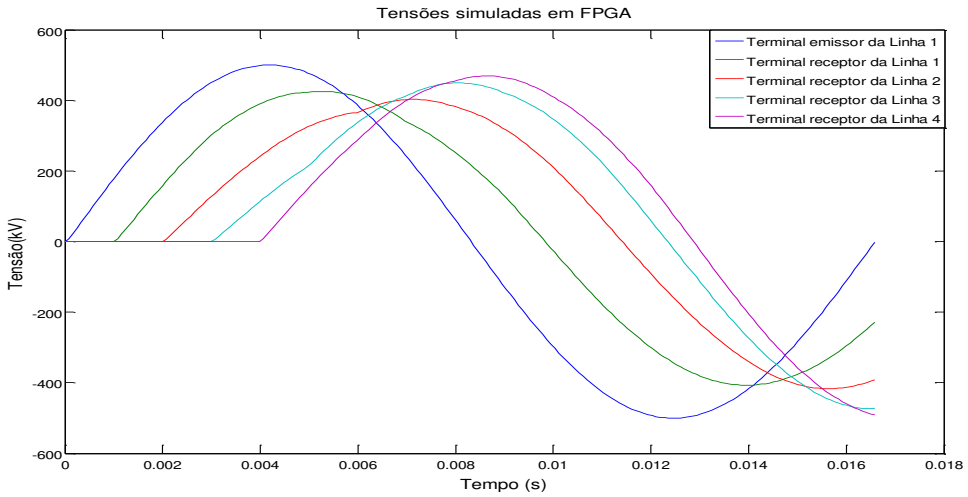
Figura 33. Diagrama unifilar de sistema monofásico com 4 linhas em série.



Fonte: Produzido pelo autor, via ATP.

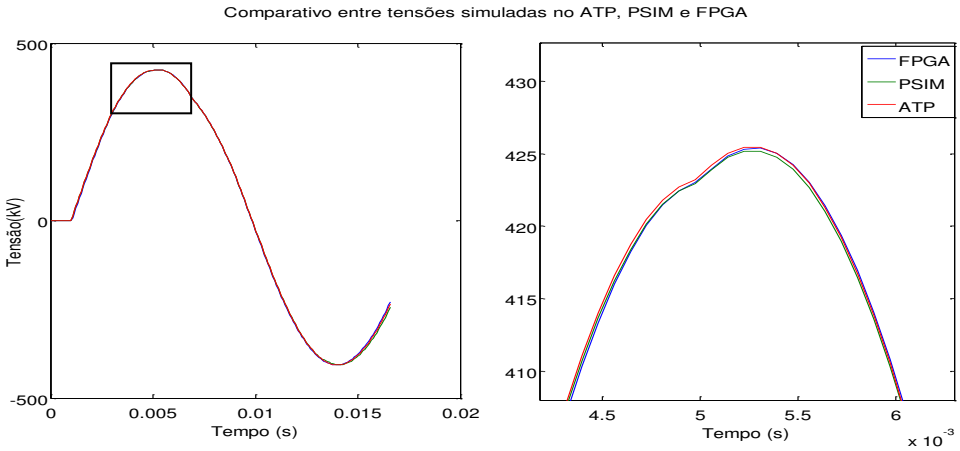
Posteriormente são apresentadas da Figura 35 à Figura 39 as curvas de tensão no terminal receptor de cada linha e a corrente fornecida pela fonte, comparando-se as curvas simuladas em FPGA com as curvas simuladas no ATP e no PSIM, durante o transitório inicial no processo de energização.

Figura 34. Tensões simuladas em FPGA nos terminais de cada linha monofásica.



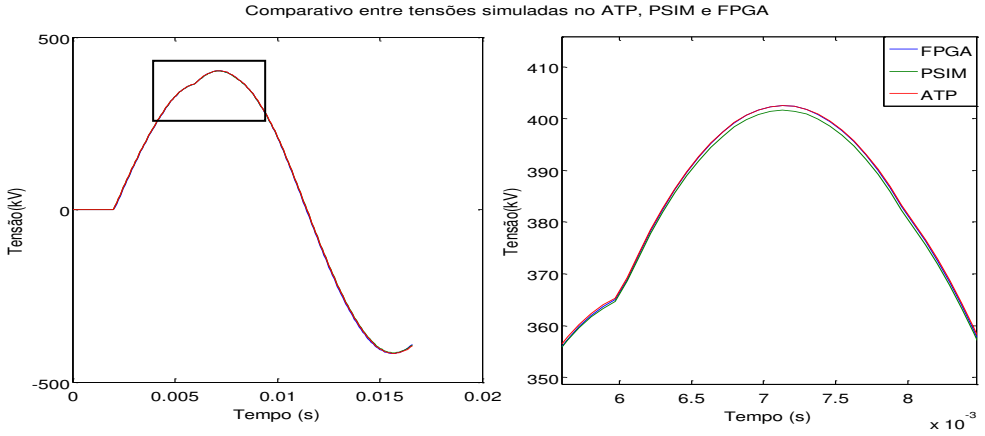
Fonte: Produzido pelo autor.

Figura 35. Comparativo entre tensões no terminal receptor da linha 1.



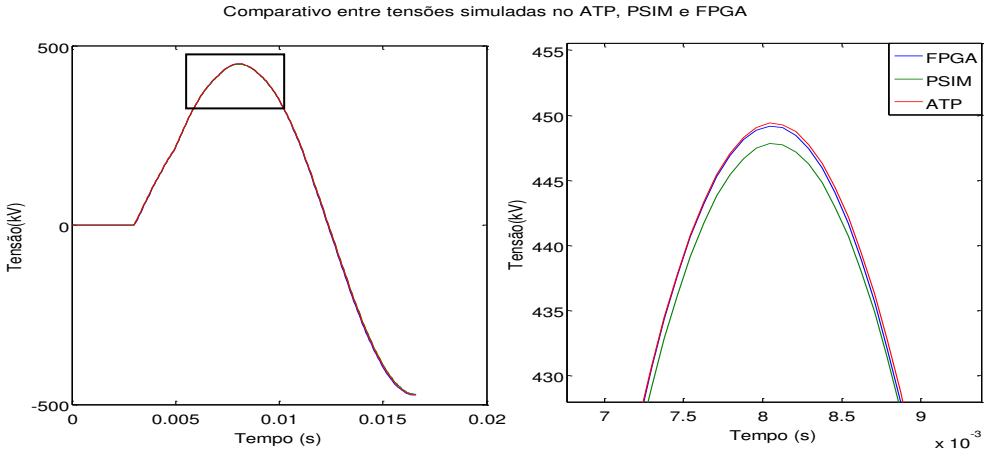
Fonte: Produzido pelo autor.

Figura 36. Comparativo entre tensões no terminal receptor da linha 2.



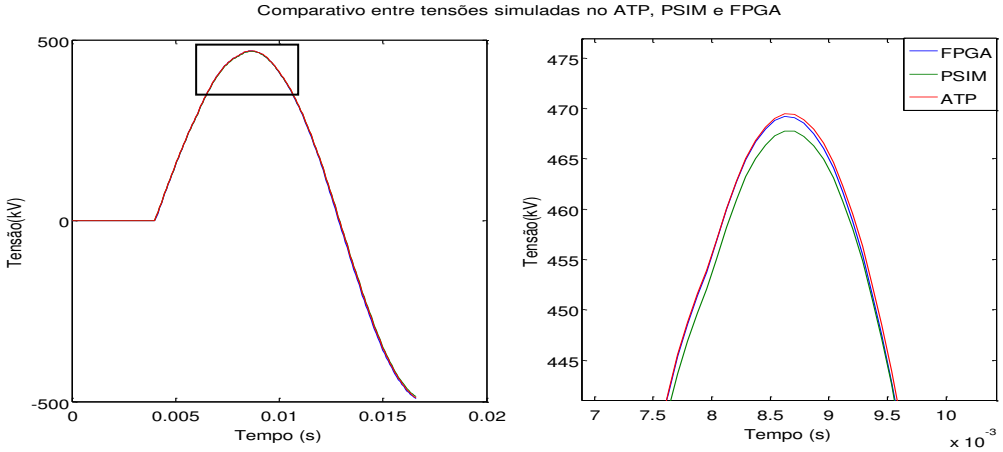
Fonte: Produzido pelo autor.

Figura 37. Comparativo entre tensões no terminal receptor da linha 3.



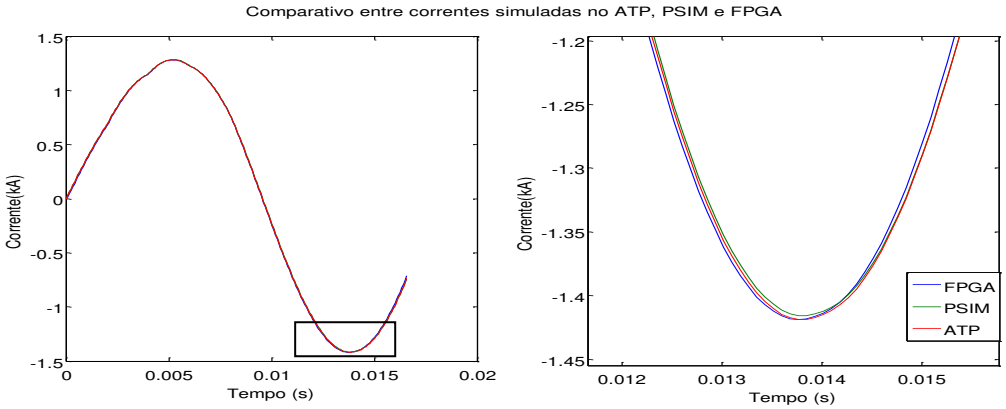
Fonte: Produzido pelo autor.

Figura 38. Comparativo entre tensões no terminal receptor da linha 4.



Fonte: Produzido pelo autor.

Figura 39. Comparativo entre curvas da corrente fornecida pelo gerador.

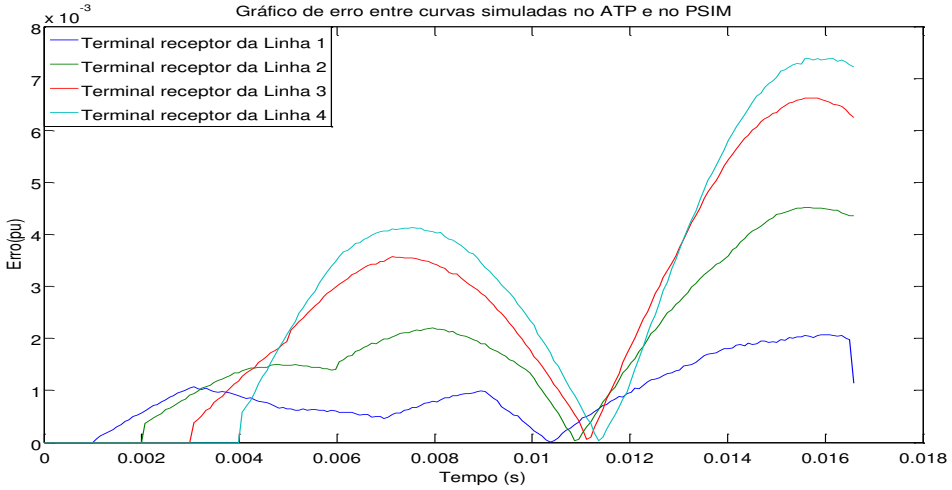


Fonte: Produzido pelo autor.

Destes gráficos observou-que as formas de onda se distanciam, nos primeiros períodos, de uma senóide pura, devido aos fenômenos de reflexão e refração de ondas ao se atingir a extremidade de uma linha, cujos efeitos vão se estabelecendo nas condições de regime. Nota-se também que esses efeitos causam elevação inicial na tensão do sistema com relação a amplitude da tensão do gerador conectado ao terminal emissor da Linha 1. O deslocamento de uma onda em distâncias da ordem de quilômetros, como é o caso do comprimento de uma linha, também ocasiona atrasos da ordem de milissegundos, de acordo com o quão mais distante estiver o terminal receptor observado em relação ao gerador.

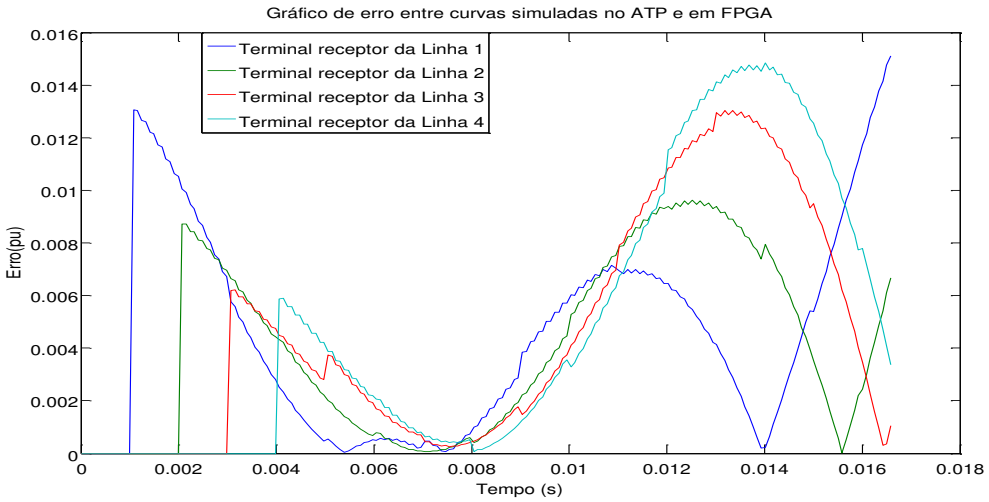
Comparando as curvas entre os diferentes simuladores observou-se que estas se aproximam, percebendo-se pequenas diferenças nas aproximações. Tomando as curvas geradas no ATP, processadas em ponto flutuante, como referência, percebeu-se que em partes das curvas geradas no PSIM e em FPGA, em ponto fixo, vão se distanciando, em termos de amplitude, dos gráficos obtidos por meio do ATP. A ocorrência destas discrepâncias ocorre devido às perdas intrínsecas ao processamento em ponto fixo abordado no capítulo 2. Entretanto em todas as curvas apresentadas na Figura 40 e na Figura 41, o erro em pu, calculado com bases de 500 kV e 1 kA se mantém na maior parte dos pontos em torno de 1%.

Figura 40. Erro nas tensões entre simulações no ATP e no PSIM.



Fonte: Produzido pelo autor.

Figura 41. Erro nas tensões entre simulações no ATP e em FPGA.



Fonte: Produzido pelo autor.

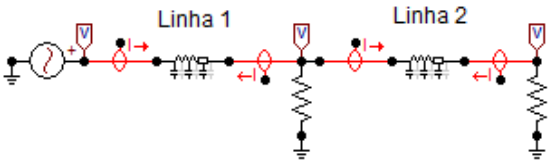
Na simulação do sistema trifásico utilizou-se também linhas idênticas e conectadas em série cujos parâmetros são apresentados na Tabela 11. Para representação destas linhas trifásicas no domínio modal foram utilizadas matrizes de Karrenbauer para n = 3, de acordo com a equação (15). No terminal receptor de cada linha foi inserida uma carga resistiva de 2 kΩ. O diagrama unifilar pode ser observado na Figura 42.

Tabela 11. Parâmetros utilizados na simulação de linhas trifásicas.

Nível de tensão (kV)	500
Comprimento da linha (km)	300
Impedância de sequência zero (Ω)	648
Impedância de sequência positiva (Ω)	290
Tempo de trânsito de sequência zero (ms)	2
Tempo de trânsito de sequência positiva (ms)	1

Fonte: Autor.

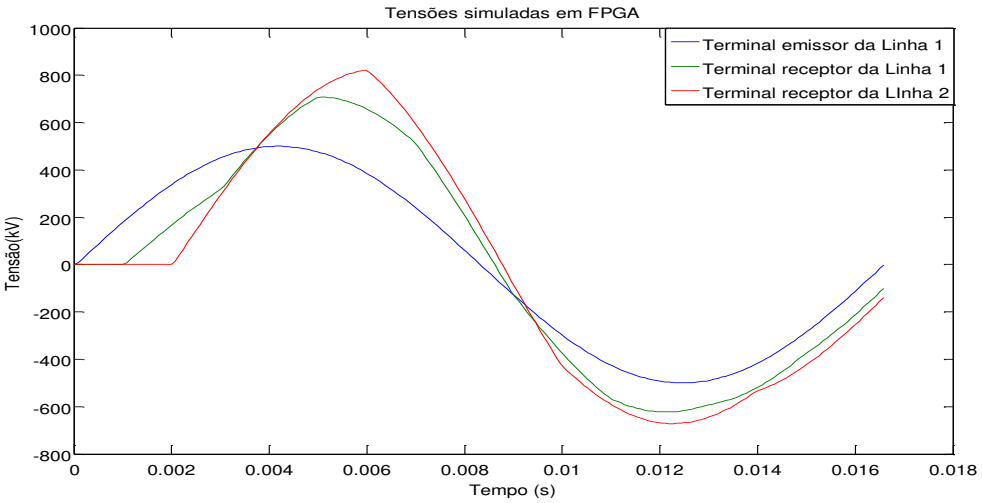
Figura 42. Diagrama unifilar de sistema trifásico com 2 linhas em série.



Fonte: Produzido pelo autor, vi ATP.

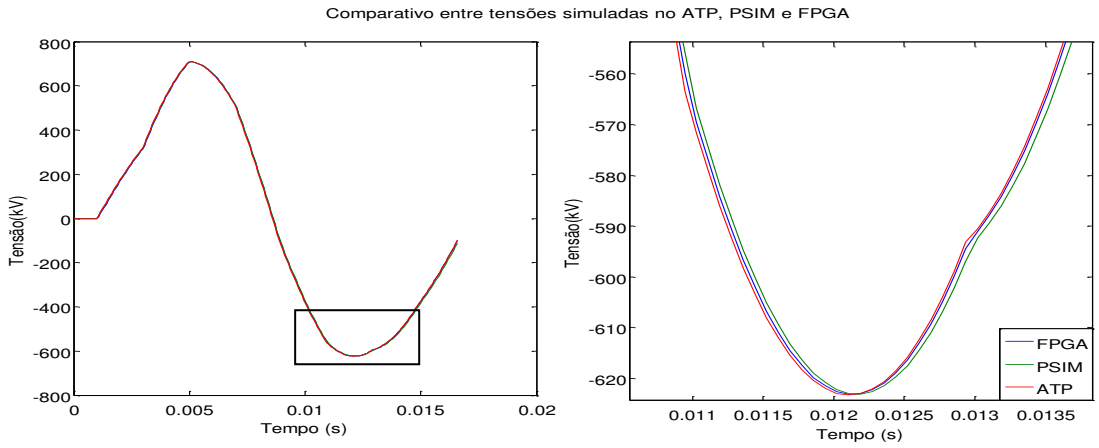
Foram construídas e observadas as curvas de tensão de uma das fases do sistema, geradas por meio dos dados obtidos nas simulações no ATP, PSIM e FPGA, durante o transitório inicial no processo de energização. Também se observou a curva da corrente fornecida pelo gerador. Estes gráficos são apresentados da Figura 43 a Figura 46.

Figura 43. Tensões simuladas em FPGA nos terminais de uma fase em cada linha trifásica.



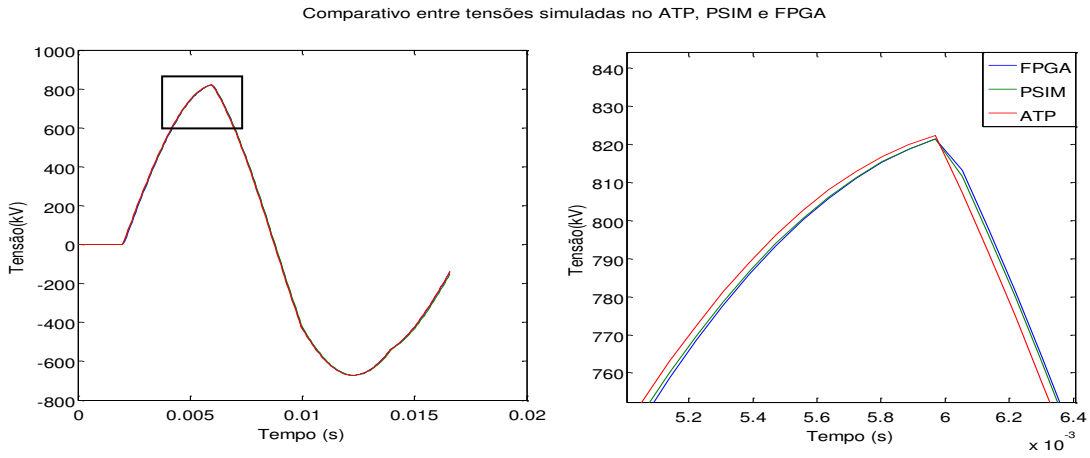
Fonte: Produzido pelo autor.

Figura 44. Comparativo entre curvas de tensão no terminal receptor da linha 1.



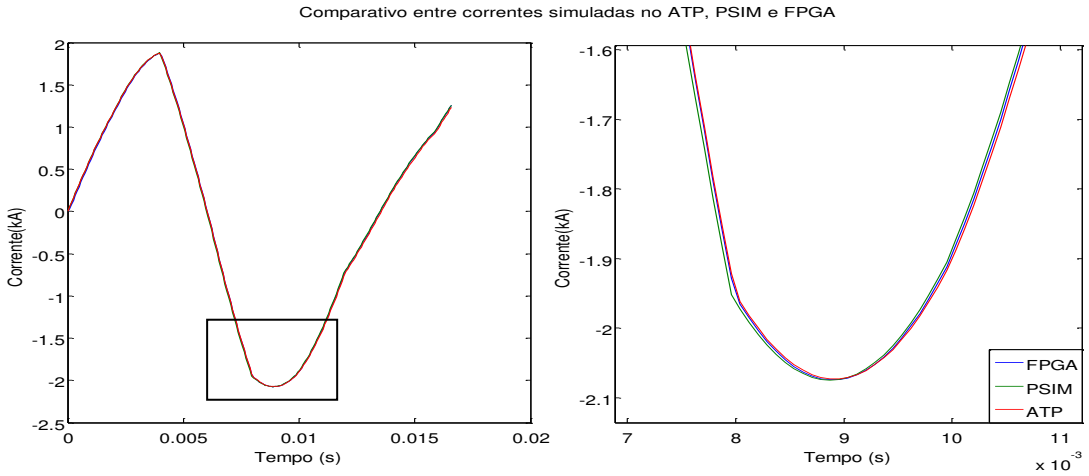
Fonte: Produzido pelo autor.

Figura 45. Comparativo entre curvas de tensão no terminal receptor da linha 2.



Fonte: Produzido pelo autor.

Figura 46. Comparativo das curvas da corrente fornecida pelo gerador trifásico.

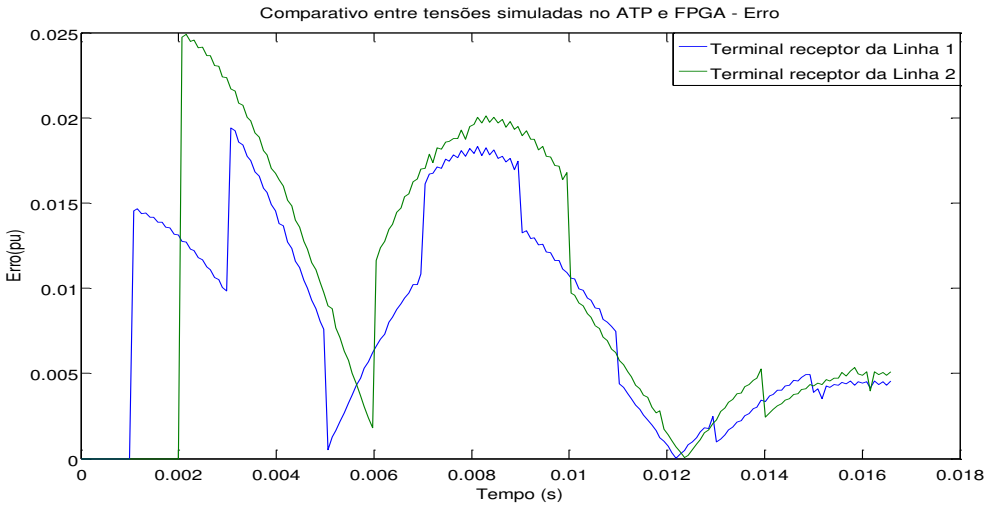


Fonte: Produzido pelo autor.

Os efeitos causados pelos fenômenos da reflexão e refração das ondas nas extremidades das linhas é novamente observado, ocasionando sobretensões que chegam a aumentar os níveis de tensão em até 50% do valor nominal. Para minimização destes efeitos utiliza-se alguma técnica para redução de transitórios, dentre as mais comuns estão elementos de pré inserção (BICKFORD; MULLINEUX; REID, 1980) e chaveamento controlado (DANTAS et al., 2011). A ampliação no uso de tecnologias de simulações em tempo real poderá se tornar uma ferramenta de auxílio à aplicação destas técnicas, como por exemplo, para escolha de instantes ótimos de inserção/remoção do elemento de pré inserção ou abertura/fechamento de uma chave.

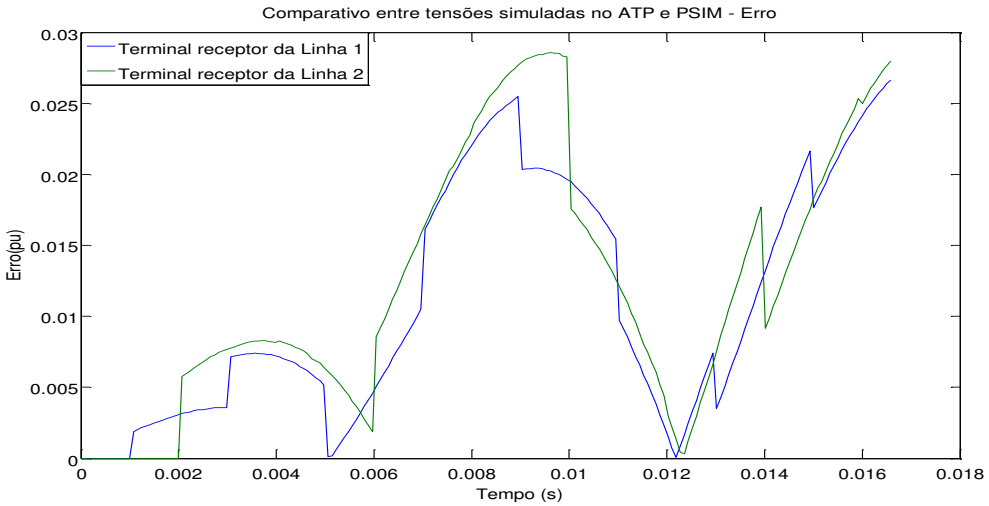
Analisando comparativamente as curvas percebeu-se um comportamento semelhante ao observado na simulação das linhas monofásicas, estando elas aproximadamente sobrepostas, com parte dos pontos se distanciando mais que outros da curva de referência simulada no ATP, processada em ponto flutuante. Nos gráficos de erro em pu apresentados na Figura 47 e na Figura 48, é visto que estes erros se mantêm abaixo de 2% para a maior parte dos pontos.

Figura 47. Gráfico de erros entre simulações no ATP e em FPGA.



Fonte: Produzido pelo autor.

Figura 48. Gráfico de erros entre simulações no ATP e no PSIM.

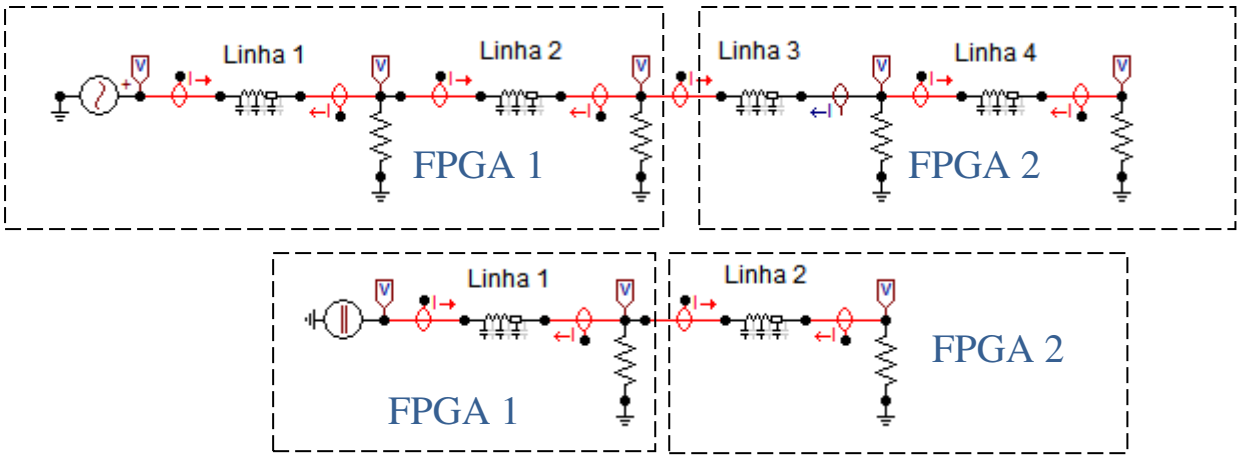


Fonte: Produzido pelo autor.

5.3 AVALIAÇÃO DA SIMULAÇÃO PROCESSADA EM DUAS PLACAS FPGA

A fim de avaliar o funcionamento das simulações processadas em dois dispositivos distintos foram simulados os mesmos sistemas apresentados na sessão anterior, com parte dos elementos simulados em uma placa e outra parte destes elementos na segunda placa. A figura 49 exibe os diagramas unifilares para o sistema monofásico com 4 linhas e o sistema trifásico com 2 linhas, baseado no esquema da Figura 28 do Capítulo 4 a respeito de como ocorre esta segmentação.

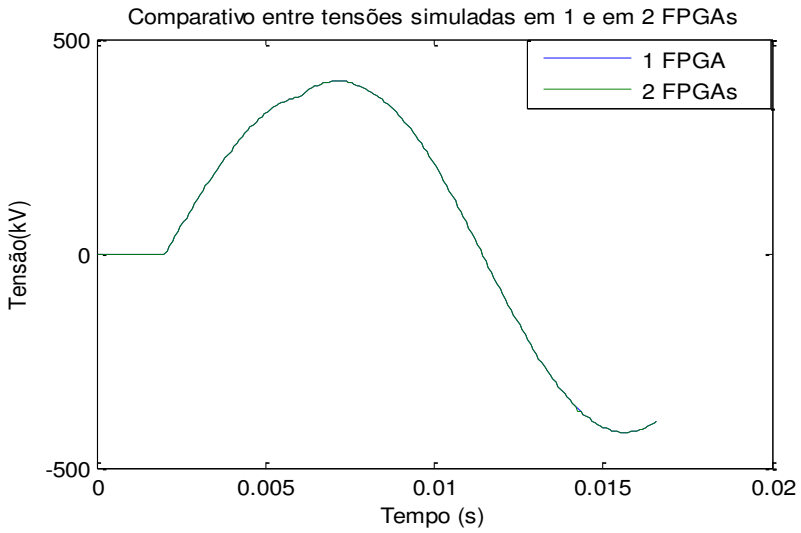
Figura 49. Segmentação da simulação em duas placas FPGA.



Fonte: Produzido pelo autor, via ATP.

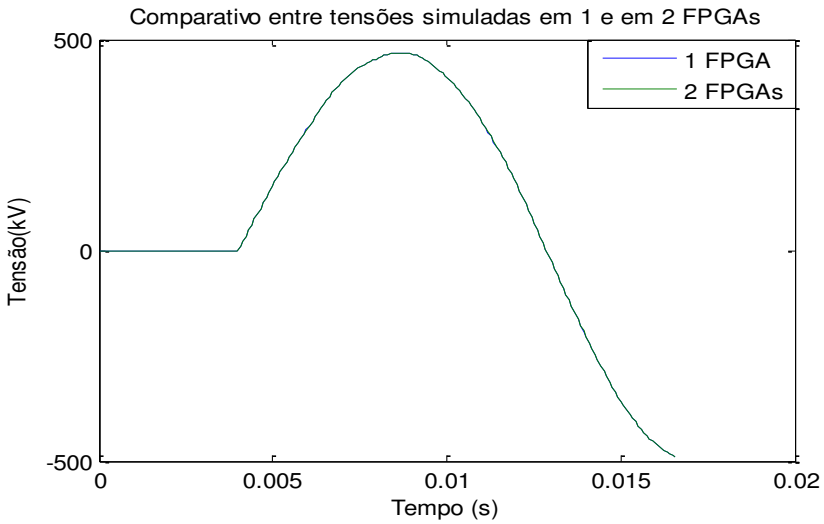
Os parâmetros utilizados na simulação do sistema monofásico são os mesmos apresentados na Tabela 9, com cargas resistivas de 2 kΩ nas linhas 1 e 3 e 1 kΩ nas linhas 2 e 4. Na Figura 50 e na Figura 51 são mostradas as curvas do transitório inicial de energização no terminal receptor das linhas 2 e 4, comparando-se a simulação processada em um único dispositivo com a simulação processada em duas placas.

Figura 50. Curvas de tensão no terminal receptor da linha 2.



Fonte: Produzido pelo autor.

Figura 51. Curvas de tensão no terminal receptor da linha 4.

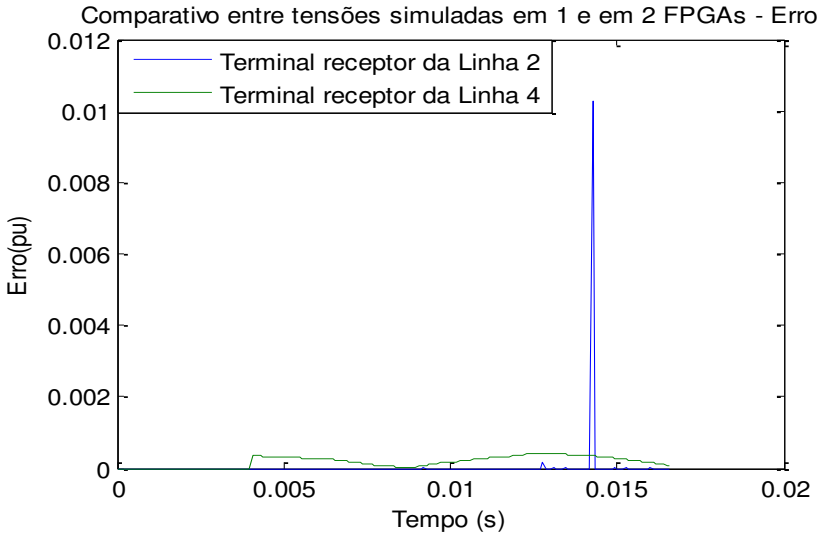


Fonte: Produzido pelo autor.

Os pontos que foram amostrados para construção das curvas estiveram, em sua maioria, muito próximos ou iguais entre os dois dispositivos, ilustrado pelas curvas sobrepostas. O gráfico de erros da Figura 52 comprova que esta diferença entre os pontos é praticamente nula, com exceção de um único ponto nas curvas de tensão do

terminal receptor da Linha 2 que apresentou uma divergência maior, em torno de 1%, mas ainda dentro de limites aceitáveis, ocasionado possivelmente por algum atraso durante a transmissão, não comprometendo os demais cálculos da simulação.

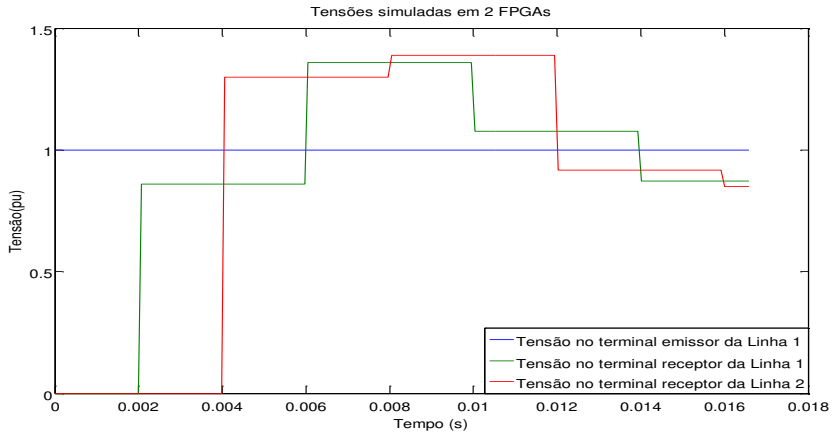
Figura 52. Gráfico de erro no cálculo das tensões simuladas em 1 e em 2 dispositivos FPGA.



Fonte: Produzido pelo autor.

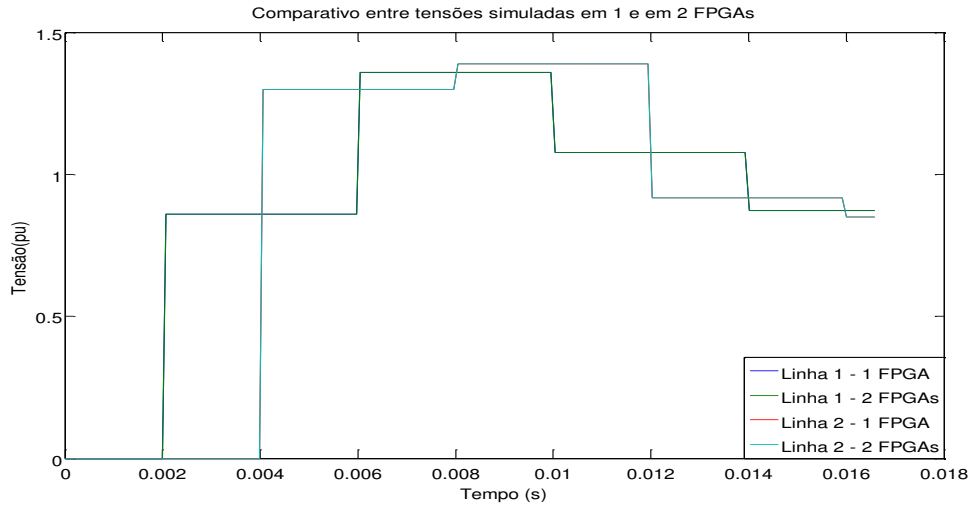
Na simulação do sistema trifásico foi aplicada uma tensão constante de valor unitário no terminal emissor da Linha 1. Os parâmetros de cada linha são os mesmos apresentados na Tabela 10, com uma carga de 2 kΩ acoplada ao terminal receptor de ambas. Desta simulação construíram-se os gráficos da tensão em cada terminal, ilustrados na Figura 53, e comparou-se com as curvas obtidas simulando o mesmo sistema em uma única placa, conforme pode ser visto na Figura 54. Pelo fato do sinal ser constante e unitário, as curvas de tensão são as mesmas nas três fases.

Figura 53. Tensões simuladas em duas placas FPGA.



Fonte: Produzido pelo autor.

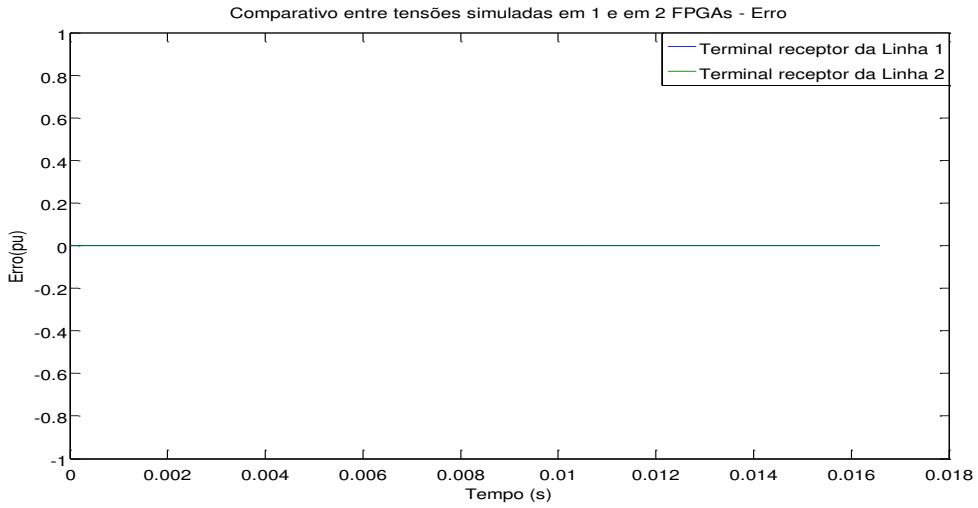
Figura 54. Comparativo entre tensões simuladas em uma e em duas placas FPGA.



Fonte: Produzido pelo autor.

Neste cenário simulado as curvas se sobrepuseram totalmente gerando um gráfico de erro nulo, conforme a Figura 55.

Figura 55. Gráfico de erro no cálculo das tensões simuladas em 1 e em 2 dispositivos FPGA.



Fonte: Produzido pelo autor.

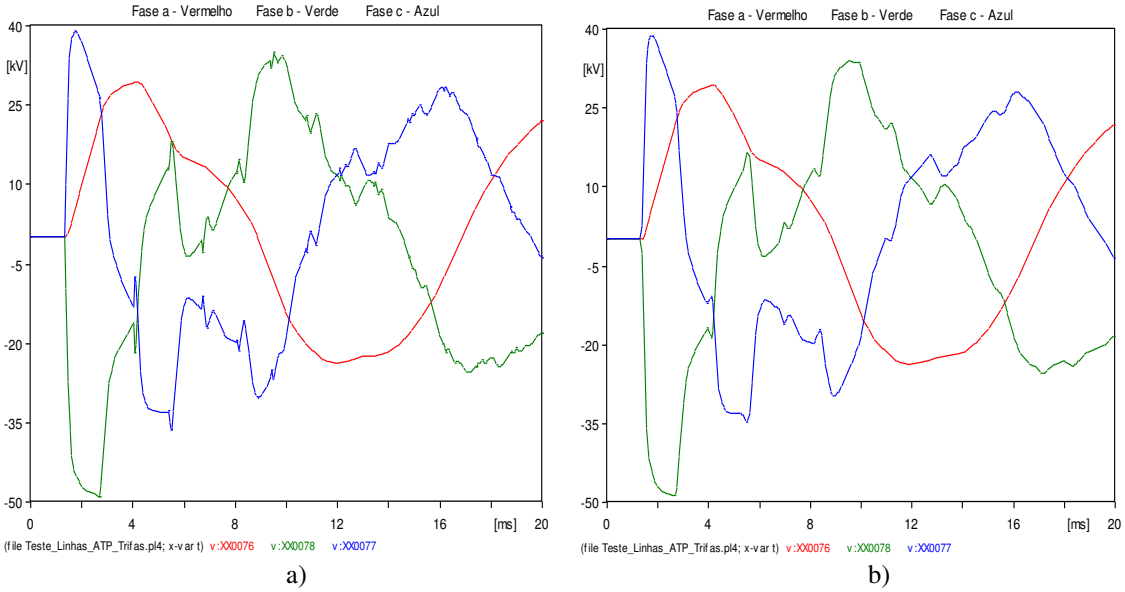
Assim, comprova-se que esta ferramenta de simulação proposta envolvendo múltiplos dispositivos, embora ainda restrita para aplicação a qualquer cenário desejado, pode apresentar resultados tão consistentes quanto o uso de uma única placa, podendo ser uma alternativa a eventuais necessidades de uso de hardwares de maiores recursos, procedendo-se com a comunicação entre dispositivos disponíveis.

5.4 SIMULAÇÃO DO SISTEMA DO IEEE POWER SYSTEM RELAYING COMMITTEE

A simulação deste sistema permitiu uma análise mais completa dos modelos implementados no simulador em tempo real processados em conjunto, e não de forma individual como nas outras simulações. Os parâmetros estão descritos no Anexo A. O diagrama unifilar do sistema está exposto na Figura 13, apresentada no Capítulo 3.

Foram simuladas todas as tensões e correntes do sistema, sendo apresentadas a seguir as curvas destas grandezas observadas no lado de baixa do transformador, aplicadas a carga. Foi realizada no ATP a mesma simulação com dois passos de cálculo distintos, um de 50 µs, típico do RTDS, e o outro de 1 µs. Da Figura 56 observa-se que as curvas perderam informações relevantes para reprodução mais fiel destas, utilizando-se um passo de cálculo maior. Com isso ressalta-se a importância de se ter alternativas para simulações em tempo real que possibilitem a utilização de passos de cálculo que permitam uma reprodução mais exata de sinais em aplicações onde estas condições sejam significativas.

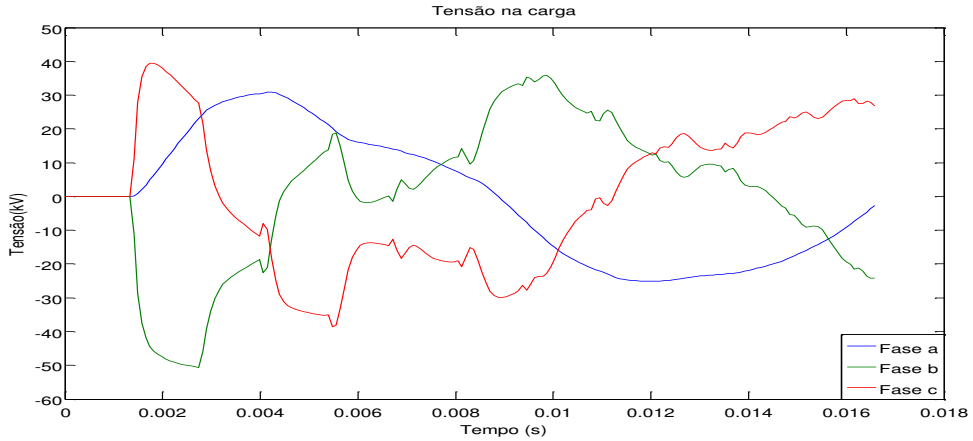
Figura 56. Simulação das tensões no ATP com passo de cálculo de: a) 1 µs. b) 50 µs.



Fonte: Produzido pelo autor.

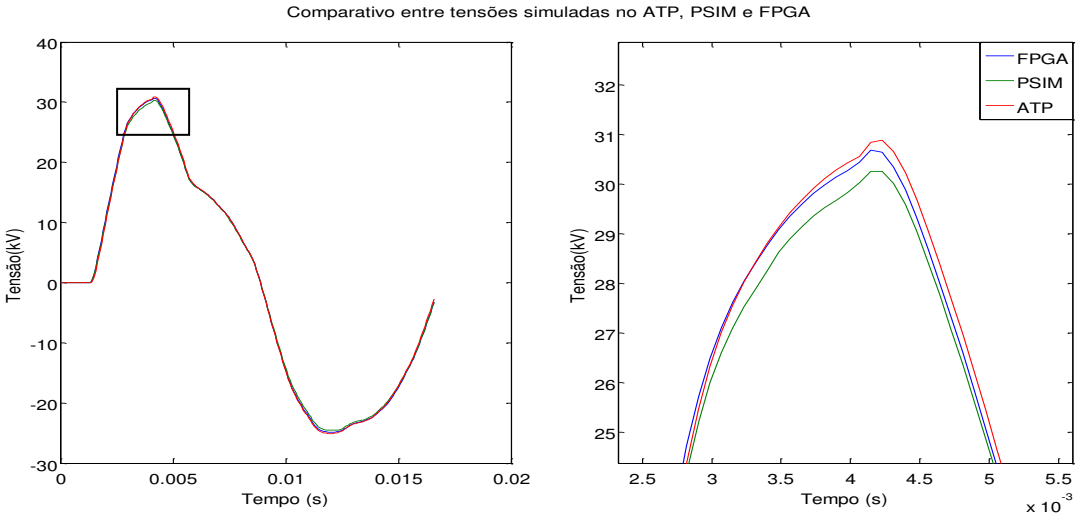
As curvas correspondentes ao transitório inicial da tensão simulada em tempo real, processadas com passo de cálculo de 1 µs podem ser vistas da Figura 57 à Figura 60.

Figura 57. Tensão aplicada as três fases da carga.



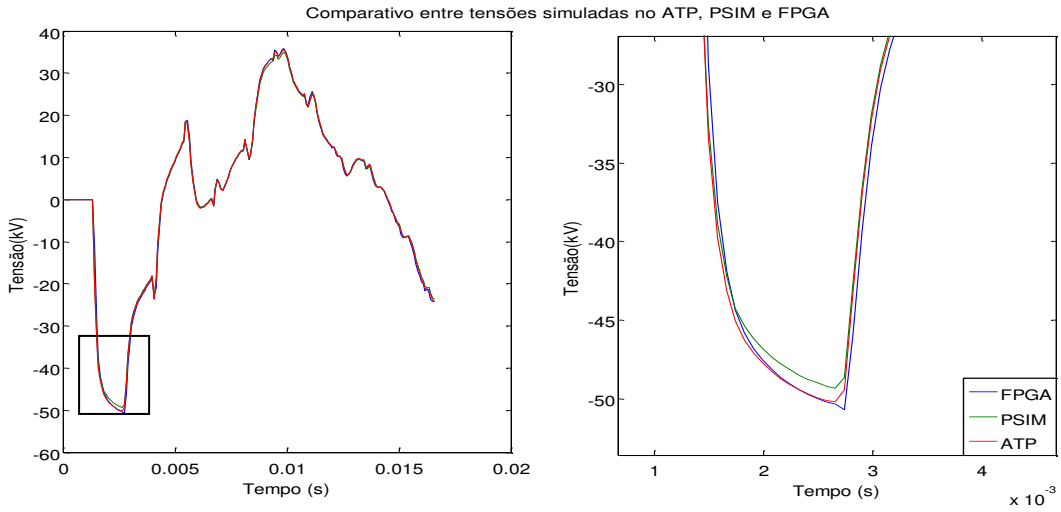
Fonte: Produzido pelo autor.

Figura 58. Comparativo entre curvas de tensão na fase a.



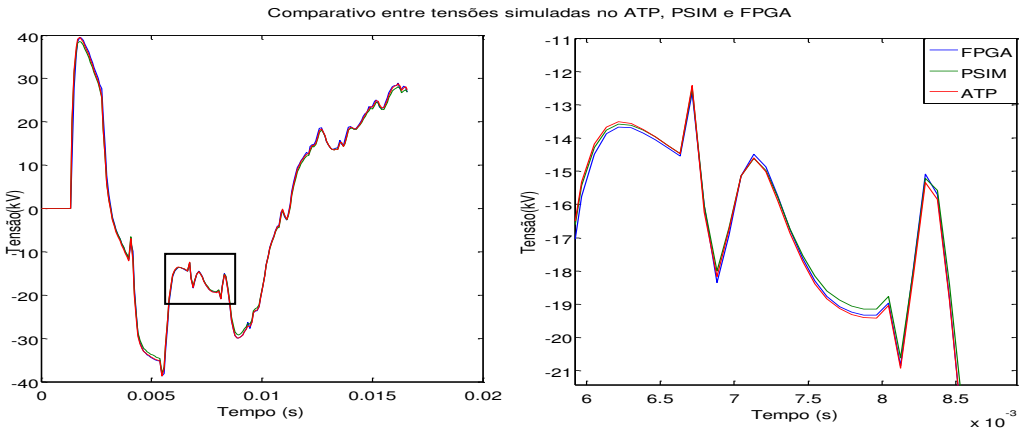
Fonte: Produzido pelo autor.

Figura 59. Comparativo entre curvas de tensão na fase b.



Fonte: Produzido pelo autor.

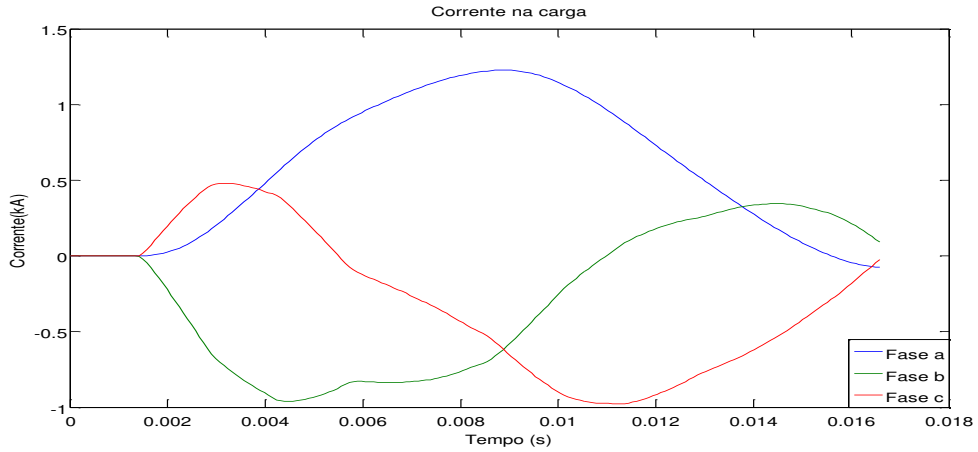
Figura 60. Comparativo entre curvas de tensão na fase c.



Fonte: Produzido pelo autor.

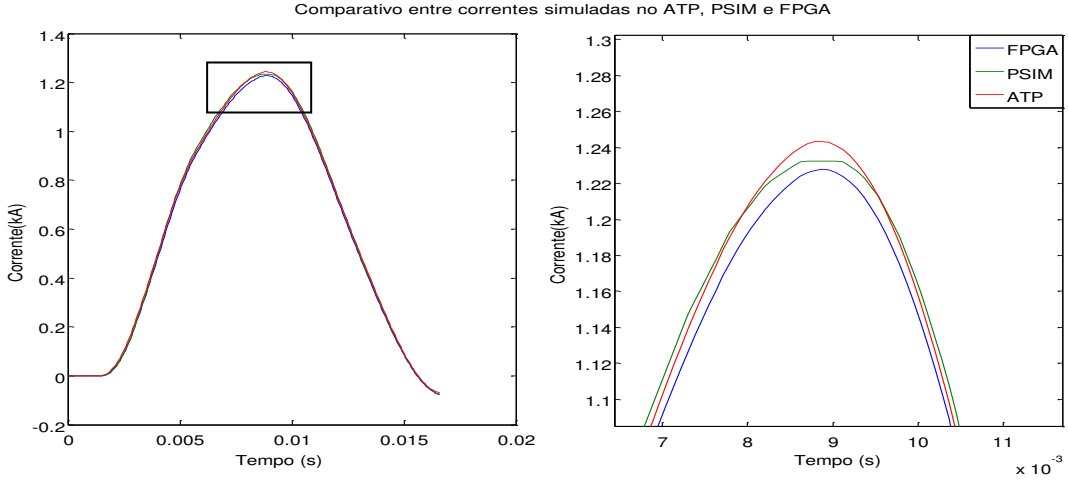
Os gráficos da corrente na carga são apresentados da Figura 48 à Figura 51.

Figura 61. Corrente nas três fases da carga.



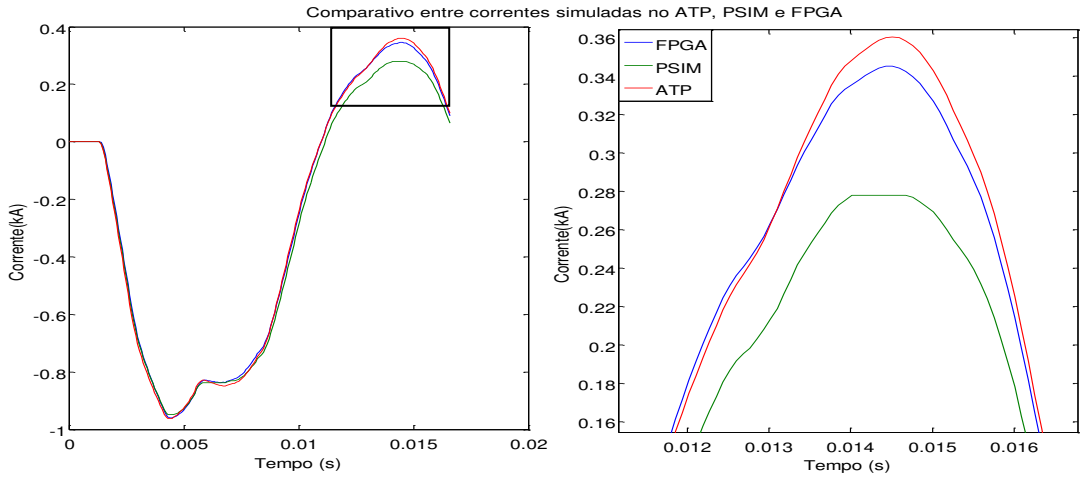
Fonte: Produzido pelo autor.

Figura 62. Comparativo entre curvas de corrente na fase a.



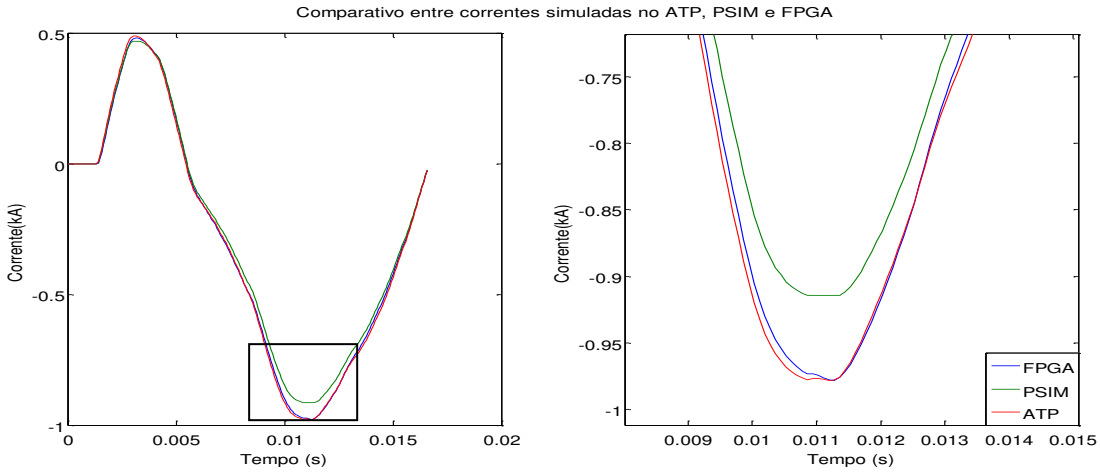
Fonte: Produzido pelo autor.

Figura 63. Comparativo entre curvas de corrente na fase b.



Fonte: Produzido pelo autor.

Figura 64. Comparativo entre curvas de corrente na fase c.



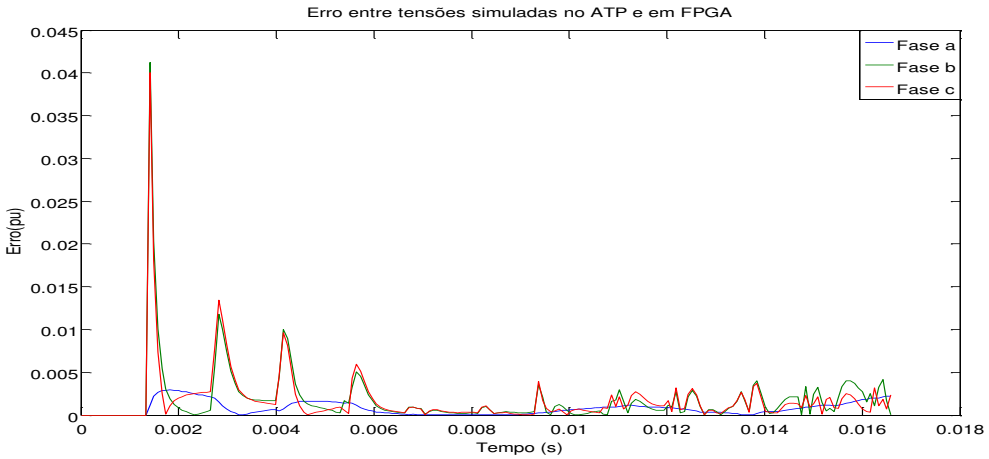
Fonte: Produzido pelo autor.

Estes transitórios observados nos gráficos são comuns em sistemas que são ligados sem mecanismos de controle, como foi efetuado na simulação deste sistema avaliado, cessando naturalmente após alguns segundos ao serem atingidas as condições de regime.

Observou-se que as curvas de tensão geradas pela simulação em tempo real apresentaram exatidão próxima das curvas de referências obtidas pela simulação no ATP, diferentemente das curvas obtidas por meio do PSIM que em vários trechos apresentou um grande distanciamento das demais, ocasionado possivelmente pelos limites de representação de variáveis em ponto fixo neste software, que é menor em relação a um dispositivo FPGA. Gráficos de erro em pu, calculados com bases de 22.8

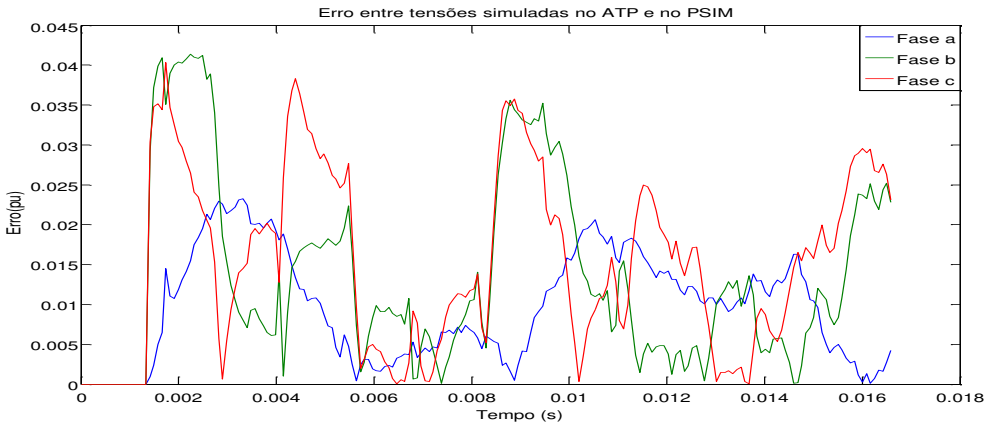
kV e 1 kA, são apresentados da Figura 65 a Figura 68, comprovando numericamente estas impressões mencionadas.

Figura 65. Gráfico de erro nas tensões entre simulações no ATP e em FPGA.



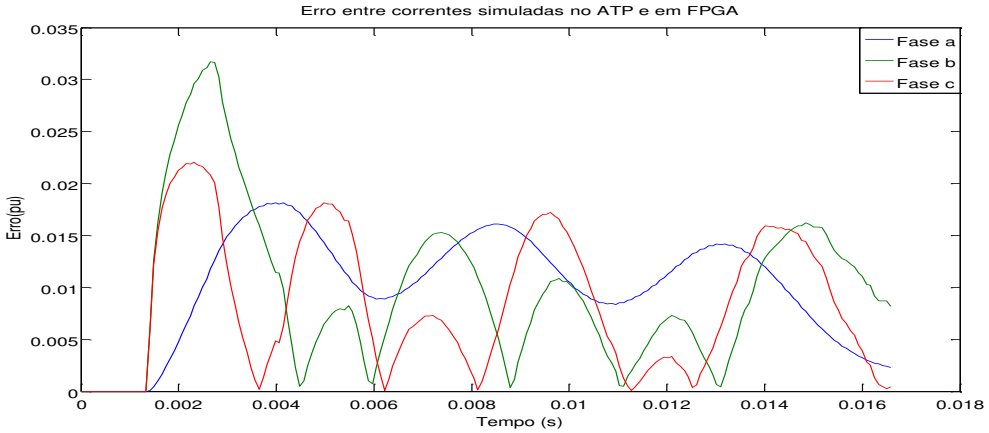
Fonte: Produzido pelo autor.

Figura 66. Gráfico de erro nas tensões entre simulações no ATP e no PSIM.



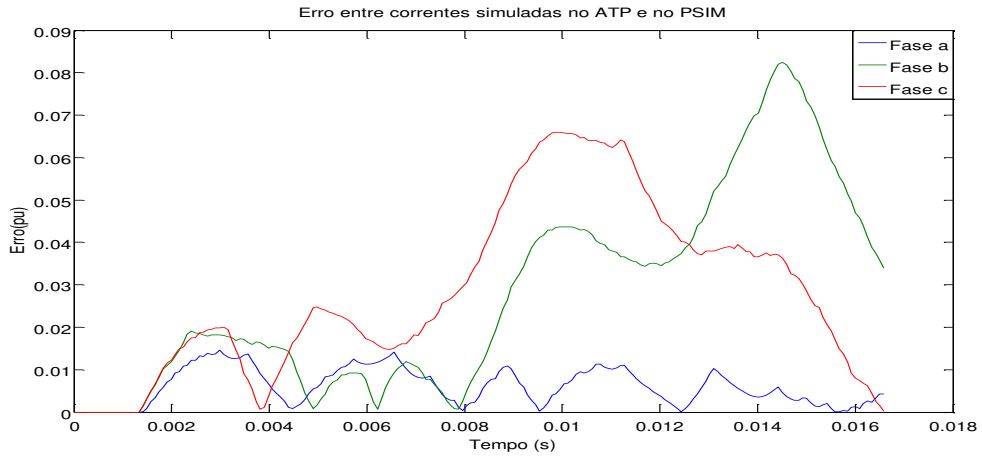
Fonte: Produzido pelo autor.

Figura 67. Gráfico de erro nas correntes entre simulações no ATP e em FPGA.



Fonte: Produzido pelo autor.

Figura 68. Gráfico de erro nas correntes entre simulações no ATP e no PSIM.



Fonte: Produzido pelo autor.

6 CONCLUSÕES

Em suma esta Dissertação buscou apresentar motivações para implementação de simulações em tempo real baseadas em dispositivos FPGA, tendo este hardware como processador principal, apresentando uma arquitetura projetada para o processamento destas simulações, embasadas em trabalhos que serviram de referência para a metodologia adotada dentro desta linha de pesquisa. Dentre as motivações apresentadas, a aplicabilidade deste tipo de hardware se destacou, como por exemplo a possibilidade inserção do simulador em condições de hardware-in-the-loop para testes de comissionamento de equipamentos de maneira mais prática, podendo gerar números de amostras em faixas consideravelmente acima do mínimo permitido em normas internacionais de proteção e qualidade de energia.

Estruturalmente foram abordados alguns detalhes que influenciam diretamente no funcionamento de forma satisfatória do simulador, como a definição de tamanho e representação de variáveis para implementação em aritmética de ponto fixo, além de estimativas para escolha da quantidade de bits, tendo em vista os impactos apresentados de uma escolha inadequada, e para definição do tamanho dos registradores de armazenamento das variáveis intermediárias. A partir destas estimativas e dos dados de síntese fornecido pelo Quartus foi possível também estimar o consumo de recursos dos modelos utilizados e a capacidade total de uma das placas ao simular sistemas contendo estes modelos.

Dentro dos testes realizados o simulador implementado apresentou, em quase sua totalidade, resultados próximos aos obtidos por meio de softwares de simulação off-line, ao comparar-se ambos. Devido ao fato das perdas intrínsecas no processamento em ponto fixo em relação a um processamento em ponto flutuante observou-se crescimentos pontuais nos gráficos de erro em comparações entre algumas amostras, porém dentro de limites toleráveis onde não há comprometimento dos demais cálculos efetuados no decorrer da simulação.

Como contribuição para o estado da arte buscou-se implementar um mecanismo de comunicação para simulação com múltiplos dispositivos que funcionou parcialmente dentro dos cenários avaliados. Na aplicação de sinais periódicos a sistemas trifásicos observou-se inconsistências em que se percebeu a necessidade da implementação de

uma estrutura de rastreamento mais precisa para identificação de erros no processo de transmissão de dados entre dispositivos. Isto pode permitir identificar também eventuais atrasos na comunicação que afetam parcialmente o sincronismo entre a transferência dos dados, acarretando em erros de menor impacto, porém passíveis de serem otimizados.

Sugere-se para trabalhos futuros para esta pesquisa:

- Implementação de outros mecanismos de comunicação, utilizando-se outras portas de comunicação disponíveis em dispositivos FPGA, tanto para transferência de dados entre placas, quanto para visualização dos dados gerados em interfaces gráficas;
- Conexão e transferência automática de dados de uma unidade de processamento externa para cálculos de inversão de matrizes realizadas em aritmética de ponto fixo;
- Implementação e avaliação de modelos mais completos em relação aos avaliados neste trabalho, por exemplo modelos de linhas dependentes da frequência e de outros elementos de sistemas elétricos não avaliados, como por exemplo modelos de chave de impedância constante.

BIBLIOGRAFIA

ALTERA. ftp://ftp.altera.com/up/pub/Webdocs/DE2_UserManual.pdf, acessado em setembro de 2017. DE2 Development and Education Board, 2006.

ALTERA. ftp://ftp.altera.com/up/pub/Altera_Material/Boards/DE2-115/DE2_115_User_Manual.pdf, acessado em janeiro de 2018. DE2-115 User Manual, 2010.

ARAÚJO, A.; NEVES, W. Cálculo de Transitórios Eletromagnéticos em Sistemas de Energia, Editora UFMG, Belo Horizonte – MG, Brasil: 2005

BICKFORD, J.; MULLINEUS, N.; REDD, J. Computation of Power-System Transients, Stevenage/UK: IEE Monograph Series 18, Peter Peregrinus Ltd., 1980.

BAHRI, I.; NAOUAR, M.; MONMASSON, E.; SLAMA-BELKHODJA I.; CHAARABI, L. Design of an FPGA-based Real-Time Simulator for electrical system, In: *2008 13th International Power Electronics and Motion Control Conference*, Poznan, 2008, pp. 1365-1370.

CHEN, Y.; DINAVAHI, V. FPGA-Based Real Time EMTP. In: *IEEE Transactions on Power Delivery*, v. 24, n. 02, p. 892-901, Apr 2009, ISSN 0885-8977.

CHEN, Y.; DINAVAHI, V. Large-Scale Real-Time Electromagnetic Transient Simulation of Power Systems Using Hardware Emulation on FPGAs. In: *Tese de Doutorado,*Edmonton-AB, Canadá, 2012.

CHEN, Y.; DINAVAHI, V. Multi-FPGA Digital Hardware Design for Detailed Large-Scale Real Time Electromagnetic Transient Simulation of Power Systems. In: *IET Generation, Transmission & Distribution*, v. 7, n. 05, p. 451-463, Jun 2013, ISSN 1751-8687.

CHENGDI, D.; PENG, L.; CHENGSHAN, W.; ZHIYING, W.; DAWEI, Y.; JIN, Z.; ZHENG, L. A Design and Implementation of FPGA-Based Real-Time Simulator for Distribution System with DG Integration. In: *China International Conference on Eletricity Distribution,* Paper No. CP0467, Xi'an - China, Aug 2016, p. 1-6.

DANTAS, K.; NEVES, W.; FERNANDES JR, D.; CARDOSO, G.; FONSECA, L. Chaveamento Controlado de Linhas de Transmissão: Uma Abordagem em Tempo Real via RTDS, In: *Revista Controle e Automação*, v. 22, n. 02, p.151-168, Março/Abril 2011.

DOMMEL, H. W. Digital Computer Solution of Electromagnetic Transients in Single and Multiphase Networks, In: *IEEE Summer Power Meeting*, Chicago - Ill, USA: 1968.

HADIZADEH, A.; HASHEMI, M.; LABBAF, M.; PARNIANI, M. A Matrix-Inversion Technique for FPGA-Based Real-time EMT Simulation of Power Converters, In: *IEEE Transactions on Industrial Electronics*.

HERRERA, I; ORTEGA, S.; MORENO, P.; PANDURO, J. J. R. Methodology for the Implementation of a Simulator of Electric Transients on FPGA. In: *IEEE 4th Colombian Workshop on Circuits and Systems*, Barranquilla- Colômbia, Nov 2012.

IEEE POWER SYSTEM RELAYING COMMITEE. *EMTP reference models for transmission line relay testing,* 2004. Disponível em: < http://www.pes-psrc.org>

JALILI-MARANDI, V.; AYRES, F. J.; GHAHREMANI, E.; BÉLANGER, J.; LAPOINTE, V. A real-time dynamic simulation tool for transmission and distribution power systems, In: *2013 IEEE Power & Energy Society General Meeting*, Vancouver, BC, 2013, pp. 1-5.

KILTS, S. Advanced FPGA Design: Architecture, Implementation and Optimization. Published by: John Wiley & Sons, Inc., Hoboken - New Jersey, USA: 2007.

KUMAR, S.; SUBBIAH, V.; KANDASWARAY, A.; KUMAR, G.; SUAJAY, R.; MANOHARAN, S. "A state of the art STATCON for instantaneous VAr compensation and harmonic suppression to enhance power quality," *CIGRE/IEEE PES International Symposium Quality and Security of Electric Power Delivery Systems, 2003. CIGRE/PES 2003.*, Montreal, Quebec, Canada, 2003, pp. 86-90.

KUON, I.; TESSIER, R.;ROSE, J. FPGA Architeture: Survey and Challenges, F*oundation and Trends in Electronic Design Automation*, Published by: Now Publishers Inc., v. 02, n. 02, p. 135-253, Hanover – MA, USA: 2008.

LE-HUY, P.; GUÉRETTE, S.; DESSAINT, L. A.; LE-HUY, H. Real-time Simulation of Power Electronics in Power Systems Using an FPGA. In: *Canadian Conference on Electrical and Computer Engineering*, Ottawa-Ont, Canadá, May 2006, p. 873-877.

LI, P.; WANG, Z.; WANG, C.; FU, X.; YU, H.; WANG, L. Synchronisation mechanism and interfaces design of multi-FPGA-based real-time simulator for microgrids, In: *IET Generation, Transmission & Distribution*, vol. 11, no. 12, pp. 3088-3096, 24 8 2017.

LOPES, F. V.; NEVES, W. L. A.; FERNANDES JR, D. Localização de faltas em tempo real baseada na teoria de ondas viajantes usando dados não sincronizados de dois terminais. In: *Tese de Doutorado* Campina Grande-PB, Brasil, Mai 2014.

LU, C.; LIOA, H.; HSIUNG, P. Multi-objective Placement of Reconfigurable Hardware Tasks in Real-Time System, In: *2009 International Conference on Computational Science and Engineering*, Vancouver, BC, 2009, pp. 921-925.

MATAR, M.; ABDEL-RAHMAN, M.; SOLIMAN, A. Developing an FPGA Coprocessor for Real-Time Simulation of Power Systems. In: *International Conference on Electrical, Electronic and Computer Engineering*, Cairo, Egypt, Sep 2004, p. 791-794.

MATAR, M.; IRAVANI, R. An FPGA-Based Real-Time Digital Simulator for Power Electronic Systems. In: *International Conference on Power Systems Transients*, Lyon, France, Jun 2007.

MATAR, M.; IRAVANI, R. FPGA Implementation of a Modified Two-Layer Network Equivalent for Real-Time Simulation of Electromagnetic Transients. In: *International Conference on Power Systems Transients*, Kyoto, Japan, Jun 2009.

MEKA, R.; SLODERBECK, M.; FARUQUE, O.; LANGSTON, J.; STEURER, M.; DEBRUNNER, L. S. FPGA Model of a High-Frequency Power Electronic Converter in an RTDS Power System Co-Simulation, In: *2013 IEEE Electric Ship Technologies Symposium (ESTS)*, Arlington, VA, 2013, pp. 71-75.

MILTON, M.; BENIGNI, A.; BAKOS, J. System-Level, FPGA-Based, Real-Time Simulation of Ship Power Systems, In: *IEEE Transactions on Energy Conversion*, vol. 32, no. 2, pp. 737-747, June 2017.

MYAING. A.; DINAVAHI, V. FPGA-Based Real Time Emulation of Power Electronic Systems with Detailed Representation of Device Charachteristics, In: *IEEE Power and Energy Society General Meeting*, Detroit-MI, USA, Jul 2011.

OPAL-RT TECHNOLOGIES. https://www.opal-rt.com/systems-hypersim/, acessado em março de 2017. Hypersim Real-Time Simulation, 2017.

OU, K. *et al*., "Research and application of small time-step simulation for MMC VSC-HVDC in RTDS," *2014 International Conference on Power System Technology*, Chengdu, 2014, pp. 877-882.

OULD-BACHIR, T.; MERDASSI, A.; CENSE, S.; BLANCHETTE, H. F.; BÉLANGER, J. FPGA-Based Real-Time Simulation of a PSIM Model: An Indirect Matrix Converter Case Study, In: *IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society*, Yokohama, 2015, pp. 003336-003340.

PALNITKAR, S. Verilog HDL: A Guide to Digital Design and Synthesis, 2nd ed. Upper Saddle River, NJ: Pearson Prentice Hall, 2003.

PEJOVIC, P.; MAKSIMOVIC, D. "A method for fast time-domain simulation of networks with switches" in *IEEE Transactions on Power Electronics*, vol. 9, no. 4, pp. 449-456, July 1994.

QI, L.; LANGSTON, J.; STEURER, M.; SUNDARAM, A. Implementation and validation of a Five-Level STATCOM Model in the RTDS small time-step environment, *2009 IEEE Power & Energy Society General Meeting*, Calgary, AB, 2009, pp. 1-6.
doi: 10.1109/PES.2009.5275691.

QI, L.; WODDRUFF, S.; STEURER, M. Study of Power Loss of Small Time-Step VSC Model in RTDS, *2007 IEEE Power Engineering Society General Meeting*, Tampa, FL, 2007, pp. 1-7.

RAZZAGHI, R.; PAOLONE, M.; RACHIDI, F. A Genereal Purpose FPGA-Based Real-Time Simulation for Power Systems Aplications. *In: 4th IEEE/PES Innovative Smart Grid Technologies Europe,* Lyngby, Denmark, Oct 2013, p. 1-5.

RAZZAGHI, R.; PAOLONE, M.; RACHIDI, F. Fast Simulation of Electromagnetic Transients in Power Systems: Numerical Solvers and Their Coupling with the Electromagnetic Time Reversal Process. In: *Tese de Doutorado,* Lausanne, Switzerland, Mar 2016.

RTDS TECHNOLOGIES. https://www.rtds.com/real-time-power-system-simulation/, acessado em março de 2017. Real time power system simulation, 2017.

RTDS TECHNOLOGIES. https://www.rtds.com/wp-content/uploads/2014/11/ RTDSNEWS_2005_summer.pdf, acessado em Setembro de 2018. RTDS News, Summer 2005.

THOMAS, D.; MOORBY, P. The Verilog Description Language, 5th ed. New York: Springer, 2002.

WANG, Z.; ZENG, F.; LI, P.; WANG, C.; FU, X.; WU, J. Kernel Solver Design of FPGA-Based Real-Time Simulator for Active Distribution Networks, In: *IEEE Access*, vol. 6, pp. 29146-29157, 2018.

ZANETTA JR, L. Transitórios Eletromagnéticos em Sistemas de Potência, Edusp - Editora da Universidade de São Paulo, São Paulo – SP, Brasil: 2003.

ZHANG, X; HE, G.; ZHOU, X. FPGA Design and Implementation for Real-Time Electromagnetic Transient Simulation System. In: *17th International Conference on High Performance Computing and Communications; 7th International Symposium on Cyberspace Safety and Security; 12th International Conference on Embedded Software and Systems,* New York-NY, USA, Aug 2015, p. 848-851.

ZHU, J.; TENG, G.; QIN, Y.; LU, D.; HU, H.; XING, Y. A Fully FPGA-Based Real-Time Simulator for the Cascaded STATCOM, In: *2016 IEEE Energy Conversion Congress and Exposition (ECCE)*, Milwaukee, WI, 2016, pp. 1-6.

ANEXO A – ALGORITMO DO GERADOR DE SINAIS SENOIDAIS TRIFÁSICOS: FASE A

```
module senoide(ent,saida,ordem,key);

input ent;
input [3:0]ordem;
output reg signed [23:0]saida;
input key;

reg signed [23:0]fator;
reg signed [46:0]n;
reg signed [46:0]delta;
reg signed [46:0]n_int;
reg signed [23:0]seno[180:0];

wire obs1;
wire [4:0] obs2;

integer cont = 0;
integer freq;
reg signed [45:0] max;
integer quad = 1;
integer sinc = 0;
integer m = 0;

always@(*)
begin
case(ordem)
0:freq = 1*60;
1:freq = 1*60;
```

```
2:freq = 2*60;
3:freq = 3*60;
4:freq = 4*60;
5:freq = 5*60;
6:freq = 6*60;
7:freq = 7*60;
8:freq = 8*60;
9:freq = 9*60;
10:freq = 10*60;
11:freq = 11*60;
12:freq = 12*60;
13:freq = 13*60;
14:freq = 14*60;
15:freq = 15*60;
endcase
seno[0] = 0;
seno[1] = 36601;
seno[2] = 73200;
seno[3] = 109794;
seno[4] = 146379;
seno[5] = 182952;
seno[6] = 219512;
seno[7] = 256056;
seno[8] = 292579;
seno[9] = 329081;
seno[10] = 365557;
seno[11] = 402006;
seno[12] = 438424;
seno[13] = 474808;
seno[14] = 511156;
seno[15] = 547466;
seno[16] = 583734;
seno[17] = 619957;
seno[18] = 656133;
```

```
seno[19] = 692259;
seno[20] = 728333;
seno[21] = 764351;
seno[22] = 800310;
seno[23] = 836209;
seno[24] = 872044;
seno[25] = 907813;
seno[26] = 943512;
seno[27] = 979140;
seno[28] = 1014693;
seno[29] = 1050169;
seno[30] = 1085565;
seno[31] = 1120878;
seno[32] = 1156106;
seno[33] = 1191246;
seno[34] = 1226295;
seno[35] = 1261251;
seno[36] = 1296110;
seno[37] = 1330871;
seno[38] = 1365531;
seno[39] = 1400087;
seno[40] = 1434536;
seno[41] = 1468875;
seno[42] = 1503103;
seno[43] = 1537217;
seno[44] = 1571213;
seno[45] = 1605090;
seno[46] = 1638844;
seno[47] = 1672474;
seno[48] = 1705976;
seno[49] = 1739349;
seno[50] = 1772589;
seno[51] = 1805693;
seno[52] = 1838661;
```

```
seno[53] = 1871488;
seno[54] = 1904173;
seno[55] = 1936713;
seno[56] = 1969105;
seno[57] = 2001348;
seno[58] = 2033438;
seno[59] = 2065373;
seno[60] = 2097151;
seno[61] = 2128769;
seno[62] = 2160225;
seno[63] = 2191517;
seno[64] = 2222641;
seno[65] = 2253597;
seno[66] = 2284381;
seno[67] = 2314990;
seno[68] = 2345424;
seno[69] = 2375679;
seno[70] = 2405753;
seno[71] = 2435644;
seno[72] = 2465349;
seno[73] = 2494866;
seno[74] = 2524194;
seno[75] = 2553329;
seno[76] = 2582270;
seno[77] = 2611015;
seno[78] = 2639560;
seno[79] = 2667904;
seno[80] = 2696045;
seno[81] = 2723981;
seno[82] = 2751710;
seno[83] = 2779229;
seno[84] = 2806536;
seno[85] = 2833630;
seno[86] = 2860507;
```

```
seno[87] = 2887167;
seno[88] = 2913607;
seno[89] = 2939825;
seno[90] = 2965820;
seno[91] = 2991588;
seno[92] = 3017129;
seno[93] = 3042439;
seno[94] = 3067519;
seno[95] = 3092364;
seno[96] = 3116974;
seno[97] = 3141347;
seno[98] = 3165480;
seno[99] = 3189373;
seno[100] = 3213022;
seno[101] = 3236427;
seno[102] = 3259585;
seno[103] = 3282495;
seno[104] = 3305155;
seno[105] = 3327564;
seno[106] = 3349719;
seno[107] = 3371619;
seno[108] = 3393262;
seno[109] = 3414647;
seno[110] = 3435771;
seno[111] = 3456634;
seno[112] = 3477234;
seno[113] = 3497569;
seno[114] = 3517638;
seno[115] = 3537439;
seno[116] = 3556970;
seno[117] = 3576231;
seno[118] = 3595219;
seno[119] = 3613933;
seno[120] = 3632372;
```

```
seno[121] = 3650535;
seno[122] = 3668420;
seno[123] = 3686025;
seno[124] = 3703349;
seno[125] = 3720392;
seno[126] = 3737151;
seno[127] = 3753625;
seno[128] = 3769814;
seno[129] = 3785716;
seno[130] = 3801329;
seno[131] = 3816653;
seno[132] = 3831686;
seno[133] = 3846427;
seno[134] = 3860876;
seno[135] = 3875030;
seno[136] = 3888890;
seno[137] = 3902453;
seno[138] = 3915719;
seno[139] = 3928686;
seno[140] = 3941355;
seno[141] = 3953724;
seno[142] = 3965791;
seno[143] = 3977556;
seno[144] = 3989019;
seno[145] = 4000177;
seno[146] = 4011031;
seno[147] = 4021580;
seno[148] = 4031822;
seno[149] = 4041758;
seno[150] = 4051385;
seno[151] = 4060704;
seno[152] = 4069714;
seno[153] = 4078414;
seno[154] = 4086803;
```

```
seno[155] = 4094881;
seno[156] = 4102647;
seno[157] = 4110101;
seno[158] = 4117241;
seno[159] = 4124069;
seno[160] = 4130582;
seno[161] = 4136780;
seno[162] = 4142664;
seno[163] = 4148232;
seno[164] = 4153484;
seno[165] = 4158420;
seno[166] = 4163039;
seno[167] = 4167341;
seno[168] = 4171326;
seno[169] = 4174993;
seno[170] = 4178342;
seno[171] = 4181373;
seno[172] = 4184085;
seno[173] = 4186479;
seno[174] = 4188554;
seno[175] = 4190310;
seno[176] = 4191747;
seno[177] = 4192865;
seno[178] = 4193664;
seno[179] = 4194143;
seno[180] = 4194303;
end


always@(posedge ent)
  begin
    fator = (4194304 - 1);
    max = 1000000/(4*60) + 1;
    n = 180*cont*fator/max;
```

```
n_int = 180*cont/max;
delta = n - (n/fator)*fator;
m = n_int + 1;
if(quad == 1)
 begin
  if(n_int == 180)
     saida = seno[n_int];
  else
    saida = (seno[n_int] + ((seno[n_int+1]-seno[n_int])*delta)/fator);
  cont = cont + 1;
  if(cont == max)
    begin
      quad = 2;
    end
end
 else if(quad == 2)
   begin
     if(n_int == 180)
       saida = seno[n_int];
     else
       saida = (seno[n_int] + ((seno[n_int+1]-seno[n_int])*delta)/fator);
    cont = cont - 1;
    if(cont == 0)
      begin
        quad = 3;
      end
  end
    else if(quad == 3)
      begin
           if(n_int == 180)
             saida = -seno[n_int];
          else
            saida = -(seno[n_int] + ((seno[n_int+1]-seno[n_int])*delta)/fator);
         cont = cont + 1;
```

```
                    if(cont == max)
                      begin
                          quad = 4;
                      end
                end
            else if(quad == 4)
                begin
                  if(n_int == 180)
                     saida = -seno[n_int];
                  else
                    saida = -(seno[n_int] + ((seno[n_int+1]-seno[n_int])*delta)/fator);
                  cont = cont - 1;
                  if(cont == 1 && sinc == 0)
                    begin
                      quad = 1;
                      sinc = 1;
                      cont = 0;
                    end
                 if(cont == 1 && sinc == 1)
                   begin
                     quad = 1;
                     sinc = 2;
                     cont = 0;
                   end
                if(cont == 2 && sinc == 2)
                  begin
                    quad = 1;
                    sinc = 0;
                    cont = 0;
                  end
              end
        end
endmodule
```

ANEXO B – ALGORITMO DO MÓDULO SELETOR DE TIPO DE SINAL E DO CLOCK ARTIFICIAL

MÓDULO SELETOR

```
module Tensao_Ent (
input sel,
output reg signed [31:0] Vin,
input clk,
input [3:0]n,
input start);

wire signed [23:0] sin_sen;
integer A = 230;

senoide s1(.ent(clk),.saida(sin_sen),.ordem(n),.key(start));

always@(*)
begin
if(sel)
   begin
         Vin = A*(4194304-1)/462;
   end
else
   begin
         Vin = ((A*(sin_sen))/462);
    end
end
endmodule
```

CLOCK ARTIFICIAL

```
module clock_100k(
input clock,
output reg clock_out,
output reg [10:0]cont,
input sel);

reg[10:0] contador = 0;

always@(posedge clock)
        begin
                if(sel)
                begin
                if(!clock_out)
                        cont = contador;
                else
                        cont = 1;
                if(contador < 25)
                        contador <= contador + 1'b1;
                else
                        begin
                                contador <= 0;
                                clock_out <= ~clock_out;
                        end
                end
                else
                begin
                clock_out = 0;
                end
        end
endmodule
```

ANEXO C –TRANSMISSÃO SERIAL

CLOCK ARTIFICIAL

```
always@(posedge CLOCK_50)
begin
        if(SW[17])
        begin
        cont1 = cont1+1;
        if(cont1>=2171)
        begin
                cont1 = 0;
                clck = clck ^1;
        end
        end
        else
                clck = 0;
end
```

MÓDULO SERIAL

```
module serial (clock ,sinal,sinal_out,sel,trigg,sel2);

input clock;
input [31:0] sinal;
input sel;
input sel2;
output reg [15:0] sinal_out;
output reg trigg;

integer cont1 = 0;
```

```
integer cont2 = 0;
integer cont3 = 0;
integer cont4 = 0;

reg [31:0] vetor [200:0];
reg [31:0] vetor_transit[200:0];
reg [31:0] aux;
reg start;

always@(posedge clock)
   begin
                    trigg = 0;
                    if(!sel)
                      begin
                        vetor[cont1] = sinal;
                        vetor_transit[cont4] = sinal;
                        cont1 = cont1 + 1;
                        cont4 = cont4 + 1;
                        if(cont1 > 200)
                            cont1 = 0;
                        if(cont4 > 200)
                            cont4 = cont4 - 1;
                      end
                    else
                      begin
                             if(sel2)
                                    begin
                                      aux = vetor_transit[cont4];
                                      if(!cont3)
                                        begin
                                              start = 0;
                                              trigg = 0;
                                              cont3 = cont3 + 1;
                                        end
```

```
else
  begin
        start = 1;
        cont3 = 0;
  end
if(cont2 == 0 && start)
  begin
        trigg = 1;
        sinal_out = {8'b00000000,aux[31:24]};
        cont2 = cont2 + 1;
   end
else if(cont2 == 1 && start)
  begin
        trigg = 1;
        sinal_out = {8'b00000000,aux[23:16]};
        cont2 = cont2 + 1;
   end
else if(cont2 == 2 && start)
  begin
        trigg = 1;
        sinal_out = {8'b00000000,aux[15:8]};
        cont2 = cont2 + 1;
   end
else if(cont2 == 3 && start)
  begin
        trigg = 1;
        sinal_out = {8'b00000000,aux[7:0]};
        cont2 = 0;
        cont4 = cont4 + 1;
   end
if(cont4 > 200)
      cont4 = 0;
if(cont5 > 200)
      cont5 = 0;
```

```
end
else
  begin
     start2 = 0;
     aux = vetor[cont1];
     if(!cont3)
        begin
              start = 0;
              trigg = 0;
              cont3 = cont3 + 1;
         end
     else
        begin
              start = 1;
              cont3 = 0;
         end
     if(cont2 == 0 && start)
        begin
              trigg = 1;
              sinal_out = {8'b00000000,aux[31:24]};
              cont2 = cont2 + 1;
         end
    else if(cont2 == 1 && start)
         begin
              trigg = 1;
              sinal_out = {8'b00000000,aux[23:16]};
              cont2 = cont2 + 1;
         end
    else if(cont2 == 2 && start)
         begin
              trigg = 1;
              sinal_out = {8'b00000000,aux[15:8]};
              cont2 = cont2 + 1;
         end
```

```
                    else if(cont2 == 3 && start)
                        begin
                            trigg = 1;
                            sinal_out = {8'b00000000,aux[7:0]};
                            cont2 = 0;
                            cont1 = cont1 + 1;
                        end
                    if(cont1 > 200)
                            cont1 = 0;
                end
            end
        end
endmodule
```

MÓDULO PORTASERIAL

```
module PortaSerial (
                    clk,
                    RxD,
                    SerialOut,
                    flag,
                    SerialOut1,
                    TxD_Req,
                    SerialIn,
                    TxD,
                    TxBusy);

input clk;
input RxD;

input TxD_Req;
input [15:0] SerialIn;
```

```
output [15:0] SerialOut;
output [10:0] SerialOut1;
output flag;
output TxD;
output TxBusy;

reg [7:0] SerialData;
reg [10:0] DataBits;
reg [13:0] acc_serial;
reg [13:0] acc_serial_in;

reg [3:0] cont_bit;
reg [2:0] cont_half;
reg flag_start;
reg flag_read;

reg flag_TxBusy;
reg flag_start_Tx;
reg flag_read_Tx;
reg [11:0] DataBitsTx;
reg [3:0] cont_bit_Tx;

wire BaudTick;
wire Baud8Tick;

assign BaudTick = acc_serial_in[13];
assign Baud8Tick = acc_serial[13];

assign flag = flag_read;
assign TxBusy = flag_TxBusy;

assign SerialOut = {8'h00,SerialData};
assign SerialOut1 = DataBits;
```

```
assign TxD = DataBitsTx[cont_bit_Tx];

always@(negedge RxD or posedge flag_read)
begin
        if(flag_read) flag_start <= 0;
        else flag_start <= 1;
end


always@(posedge TxD_Req or posedge flag_read_Tx)
begin
        DataBitsTx = {3'b111,SerialIn[7:0],1'b0};
        if(flag_read_Tx) flag_start_Tx = 0;
        else flag_start_Tx = 1;
end


always@(posedge clk)
begin
/* ------------------- RxD --------------------*/
        if( (flag_start) && (!flag_read) )
        begin
                flag_read  <= 1;
                acc_serial <= 0;
                cont_bit   <= 0;
                cont_half  <= 0;
        end

        if( (Baud8Tick) && (flag_read) )
        begin
                cont_half <= cont_half + 1;
                if(cont_half==4)
                begin
                        DataBits[cont_bit] <= RxD;
```

```
                cont_bit <= cont_bit + 1;
        end
        if(cont_bit>10)
        begin
                cont_bit <= 0;
                flag_read <= 0;
                SerialData = DataBits[8:1];
        end
    end


/* ------------------- TxD ---------------------*/
    if( (flag_start_Tx) && (!flag_read_Tx) )
    begin
        flag_read_Tx  = 1;
        flag_TxBusy   = 1;
        cont_bit_Tx   = 0;
        acc_serial_in = 0;
    end

    if( (BaudTick) && (flag_read_Tx) )
    begin
        cont_bit_Tx = cont_bit_Tx + 1;
        if(cont_bit_Tx>10)
        begin
                flag_read_Tx = 0;
                flag_TxBusy  = 0;
                cont_bit_Tx  = 10;
        end
    end

    if( (!flag_TxBusy) && (cont_bit_Tx!=10) ) cont_bit_Tx =10;

//      acc_serial <= acc_serial[12:0]+280;
```

```
        //      acc_serial_in = acc_serial_in[12:0]+35;

                acc_serial <= acc_serial[12:0]+151;
                acc_serial_in = acc_serial_in[12:0]+19;

        end

endmodule
```

ANEXO D – MODELOS DE LINHA E DE TRANSFORMADOR

LINHAS

```
module Linhas (
input clk,
input signed [31:0] Tens_in,
input sel,
input [63:0] Zclinha,
input signed [31:0] Iklinha,
input ultim,
output reg signed [31:0] Vm,
output reg signed [31:0] ikm,
output reg signed [31:0] imk,
output reg [63:0] Zcout,
output reg signed [31:0] Ikout,
input trans);

wire f1;

reg signed [46:0] Ik [999:0];
reg signed [46:0] Im [999:0];

integer Zc = 500;
reg [53:0] carga;
integer ind = 0;

integer Vbase = 462;
integer Ibase = 1;
```

```
        integer Zbase = 462;
        integer chave = 2;

        reg signed [127:0]Zpfixo;
        reg signed [46:0]Vkpfixo;
        reg signed [46:0]Iklinhapfixo;
        reg signed [46:0]Vmpfixo;
        reg signed [46:0]ikpfixo;
        reg signed [46:0]impfixo;
        reg signed [46:0]Ikoutaux;
        reg signed [50:0]intermed1;
        reg signed [50:0]intermed2;
        reg signed [46:0]fat1;
        reg signed [46:0]fat2;

        wire obs1;
        wire [4:0] obs2;

always@(*)
      begin
        if(sel)
            begin
                carga = 2000;
                chave = 1;
            end
        else
          begin
             carga = 50'b11100011010111111010100100110001101000000000000000;
             chave = 0;
           end
       end


        clock_100k c1 (.clock(clk),.clock_out(f1),.ativ(obs1),.sel(trans),.cont(obs2));
```

```
        always@(posedge clk)
            begin
                fat1 = (2**22);
                fat2 = (2**17);
                if(chave)
                   Zpfixo = ((fat2-
1)*Zc*(carga*Zclinha/(carga+Zclinha)))/(Zbase*(Zc+(carga*Zclinha/(carga+Zclinha)))
);
                else
                Zpfixo = (Zc/(2-ultim))*(fat2-1)/Zbase;
                Vkpfixo = Tens_in;
                Vmpfixo = (Zpfixo*(Im[0]+Iklinha))/fat2;
                intermed1 = ((Zbase*((fat2)-1)/Zc)*Vkpfixo/(fat2));
                intermed2 = ((Zbase*((fat2)-1)/Zc)*Vmpfixo/(fat2));
                ikpfixo = intermed1 - Ik[0];
                impfixo = intermed2 - Im[0];
                Ik[1000] = intermed2 + impfixo;
                Im[1000] = intermed1 + ikpfixo;

                for(ind = 1; ind <=1000; ind = ind + 1)
                    begin
                        Ik[ind-1]=Ik[ind];
                        Im[ind-1]=Im[ind];
                    end

                Vm = Vmpfixo[31:0];
                ikm = ikpfixo[31:0];
                imk = impfixo[31:0];
                Ikoutaux = Ik[1];
                Ikout = Ikoutaux[31:0];
                Zcout = Zc;
      end
endmodule
```

LINHAS_TRI

```
module Linhas_Tri (
input clk,
input signed [31:0] Tens_inA,
input signed [31:0] Tens_inB,
input signed [31:0] Tens_inC,
input sel,
input signed [31:0] Iklinha_a,
input signed [31:0] Iklinha_b,
input signed [31:0] Iklinha_c,
input ultim,
output reg signed [31:0] Vm_a,
output reg signed [31:0] Vm_b,
output reg signed [31:0] Vm_c,
output reg signed [31:0] ikm_a,
output reg signed [31:0] ikm_b,
output reg signed [31:0] ikm_c,
output reg signed [31:0] imk_a,
output reg signed [31:0] imk_b,
output reg signed [31:0] imk_c,
output reg signed [31:0] Ikout_a,
output reg signed [31:0] Ikout_b,
output reg signed [31:0] Ikout_c,
input trans);

wire f1;
wire[4:0] conta;

reg[11:0] ind1 = 0;
reg[12:0] ind2 = 0;
reg[11:0] Z0 = 648;
reg[11:0] Z1 = 290;
```

```
reg [10:0]Zbase = 462;

reg signed [20:0]Z[2:0][2:0];
reg signed [46:0]Vk[2:0];
reg signed [46:0]Vkmod[2:0];
reg signed [46:0]Vm[2:0];
reg signed [46:0]Vmmod[2:0];
reg signed [46:0]ikm[2:0];
reg signed [46:0]imk[2:0];
reg signed [46:0]Jkm[2:0];
reg signed [46:0]Jmk[2:0];
reg signed [46:0]intermed1[2:0];
reg signed [30:0]fat1;
reg signed [30:0]fat2;

reg signed [46:0] Jk0 [2000:0];
reg signed [46:0] Jk1 [1000:0];
reg signed [46:0] Jk2 [1000:0];
reg signed [46:0] Jm0 [2000:0];
reg signed [46:0] Jm1 [1000:0];
reg signed [46:0] Jm2 [1000:0];
reg signed [46:0] Ik [2:0];
reg signed [46:0] Im [2:0];

always@(*)
  begin
                if(sel)
                  begin
                        if(ultim)
                                begin
                                        Z[0][0] = 94187;
                                        Z[1][1] = Z[0][0];
                                        Z[2][2] = Z[0][0];
                                        Z[0][1] = 22332;
```

```
                    Z[0][2] = Z[0][1];
                    Z[1][0] = Z[0][1];
                    Z[1][2] = Z[0][1];
                    Z[2][0] = Z[0][1];
                    Z[2][1] = Z[0][1];
                end
            else
                begin
                    Z[0][0] = 51939;
                    Z[1][1] = Z[0][0];
                    Z[2][2] = Z[0][0];
                    Z[0][1] = 13582;
                    Z[0][2] = Z[0][1];
                    Z[1][0] = Z[0][1];
                    Z[1][2] = Z[0][1];
                    Z[2][0] = Z[0][1];
                    Z[2][1] = Z[0][1];
                end
            end
        else
            begin
                if(ultim)
                    begin
                        Z[0][0] = 116129;
                        Z[1][1] = Z[0][0];
                        Z[2][2] = Z[0][0];
                        Z[0][1] = 33855;
                        Z[0][2] = Z[0][1];
                        Z[1][0] = Z[0][1];
                        Z[1][2] = Z[0][1];
                        Z[2][0] = Z[0][1];
                        Z[2][1] = Z[0][1];
                    end
                else
```

```
                    begin
                        Z[0][0] = 58064;
                        Z[1][1] = Z[0][0];
                        Z[2][2] = Z[0][0];
                        Z[0][1] = 16927;
                        Z[0][2] = Z[0][1];
                        Z[1][0] = Z[0][1];
                        Z[1][2] = Z[0][1];
                        Z[2][0] = Z[0][1];
                        Z[2][1] = Z[0][1];
                    end
                end
            end


        always@(posedge clk)
            begin
                fat1 = (2**22);
                fat2 = (2**17);

                Vk[0] = Tens_inA;
                Vk[1] = Tens_inB;
                Vk[2] = Tens_inC;

                Vkmod[0] = ((Vk[0]+ Vk[1] + Vk[2])/3);
                Vkmod[1] = ((Vk[0] - Vk[1])/3);
                Vkmod[2] = ((Vk[0] - Vk[2])/3);

                Vm[0] = (Z[0][0]*(Im[0]+Iklinha_a) +
Z[0][1]*(Im[1]+Iklinha_b) + Z[0][2]*(Im[2]+Iklinha_c))/fat2;
                Vm[1] = (Z[1][0]*(Im[0]+Iklinha_a) +
Z[1][1]*(Im[1]+Iklinha_b) + Z[1][2]*(Im[2]+Iklinha_c))/fat2;
                Vm[2] = (Z[2][0]*(Im[0]+Iklinha_a) +
Z[2][1]*(Im[1]+Iklinha_b) + Z[2][2]*(Im[2]+Iklinha_c))/fat2;
```

```
Vmmod[0] = (Vm[0] + Vm[1]+ Vm[2])/3;
Vmmod[1] = (Vm[0] - Vm[1])/3;
 Vmmod[2] = (Vm[0] - Vm[2])/3;

intermed1[0] = (Zbase*((fat2)-1)/Z0);
intermed1[1] = (Zbase*((fat2)-1)/Z1);
intermed1[2] = (Zbase*((fat2)-1)/Z1);

Jkm[0] = intermed1[0]*Vkmod[0]/(fat2) - Jk0[0];
Jkm[1] = intermed1[1]*Vkmod[1]/(fat2) - Jk1[0];
Jkm[2] = intermed1[2]*Vkmod[2]/(fat2) - Jk2[0];

Jmk[0] = intermed1[0]*Vmmod[0]/(fat2) - Jm0[0];
Jmk[1] = intermed1[1]*Vmmod[1]/(fat2) - Jm1[0];
Jmk[2] = intermed1[2]*Vmmod[2]/(fat2) - Jm2[0];

Jk0[2000] = intermed1[0]*Vmmod[0]/(fat2) + Jmk[0];
Jk1[1000] = intermed1[1]*Vmmod[1]/(fat2) + Jmk[1];
Jk2[1000] = intermed1[2]*Vmmod[2]/(fat2) + Jmk[2];

Jm0[2000] = intermed1[0]*Vkmod[0]/(fat2) + Jkm[0];
Jm1[1000] = intermed1[1]*Vkmod[1]/(fat2) + Jkm[1];
Jm2[1000] = intermed1[2]*Vkmod[2]/(fat2) + Jkm[2];

for(ind1 = 1; ind1 <=1000; ind1 = ind1 + 1)
    begin
            Jk1[ind1-1]=Jk1[ind1];
            Jk2[ind1-1]=Jk2[ind1];
            Jm1[ind1-1]=Jm1[ind1];
            Jm2[ind1-1]=Jm2[ind1];
     end

for(ind2 = 1; ind2 <=2000; ind2 = ind2 + 1)
```

```
        begin
                Jk0[ind2-1]=Jk0[ind2];
                Jm0[ind2-1]=Jm0[ind2];
        end

    ikm[0] = Jkm[0] + Jkm[1] + Jkm[2];
    ikm[1] = Jkm[0] - 2*Jkm[1] + Jkm[2];
    ikm[2] = Jkm[0] + Jkm[1] - 2*Jkm[2];

    imk[0] = Jmk[0] + Jmk[1] + Jmk[2];
    imk[1] = Jmk[0] - 2*Jmk[1] + Jmk[2];
    imk[2] = Jmk[0] + Jmk[1] - 2*Jmk[2];

    Ik[0] = Jk0[1] + Jk1[1] + Jk2[1];
    Ik[1] = Jk0[1] - 2*Jk1[1] + Jk2[1];
    Ik[2] = Jk0[1] + Jk1[1] - 2*Jk2[1];

    Im[0] = Jm0[0] + Jm1[0] + Jm2[0];
    Im[1] = Jm0[0] - 2*Jm1[0] + Jm2[0];
    Im[2] = Jm0[0] + Jm1[0] - 2*Jm2[0];

    Vm_a = Vm[0];
    Vm_b = Vm[1];
    Vm_c = Vm[2];
    ikm_a = ikm[0];
    ikm_b = ikm[1];
    ikm_c = ikm[2];
    imk_a = imk[0];
    imk_b = imk[1];
    imk_c = imk[2];
    Ikout_a = Ik[0];
    Ikout_b = Ik[1];
    Ikout_c = Ik[2];
end
```

```
endmodule
```

TRANSFORMADOR

```
module Transformador(
input clk,
input signed [31:0] Vk_a,
input signed [31:0] Vk_b,
input signed [31:0] Vk_c,
input signed [31:0] iklinha_a,
input signed [31:0] iklinha_b,
input signed [31:0] iklinha_c,
output reg signed [31:0] Vlowa,
output reg signed [31:0] Vlowb,
output reg signed [31:0] Vlowc,
output reg signed [31:0] Ia,
output reg signed [31:0] Ib,
output reg signed [31:0] Ic,
output reg signed [31:0] Ika,
output reg signed [31:0] Ikb,
output reg signed [31:0] Ikc);

reg signed [30:0]fat2;

reg [2:0]Rh;
reg Rl;
reg [10:0] Yl;

reg signed[46:0] Vha;
reg signed[46:0] Vhb;
reg signed[46:0] Vhc;

reg signed [46:0] Vla;
```

```
reg signed [46:0] Vlb;
reg signed [46:0] Vlc;

reg signed[46:0] Ila;
reg signed[46:0] Ilb;
reg signed[46:0] Ilc;

reg signed[46:0] iL_a;
reg signed[46:0] iL_b;
reg signed[46:0] iL_c;

reg signed[46:0] out_a;
reg signed[46:0] out_b;
reg signed[46:0] out_c;

reg signed [6:0]n;

always@(posedge clk)
    begin
                        fat2 = 16384;
                        Rh = 4;
                        Rl = 1;
                        Yl = 37;
                        n = 10;

                        Vha = Vk_a;
                        Vhb = Vk_b;
                        Vhc = Vk_c;

                        Vla = Vha/10;
                        Vlb = Vhb/10;
                        Vlc = Vhc/10;

                        iL_a = (Yl*Vla/fat2 + Ila);
```

```
                    iL_b = (Yl*Vlb/fat2 + Ilb);
                    iL_c = (Yl*Vlc/fat2 + Ilc);

                    Ila= Yl*Vla/fat2 + iL_a;
                    Ilb= Yl*Vlb/fat2 + iL_a;
                    Ilc= Yl*Vlc/fat2 + iL_a;

                    out_a = -(Yl*Vla/fat2 + Ila);
                    out_b = -(Yl*Vlb/fat2 + Ilb);
                    out_c = -(Yl*Vlc/fat2 + Ilc);

                    Vlowa = Vla;
                    Vlowb = Vlb;
                    Vlowc = Vlc;
                    Ia = iL_a;
                    Ib = iL_b;
                    Ic = iL_c;
                    Ika = out_a;
                    Ikb = out_b;
                    Ikc = out_c;
                end
endmodule
```

ANEXO E – MÓDULO PRINCIPAL: MOD_TESTE

```
module Mod_Teste(
input CLOCK_27,
input CLOCK_50,
input [3:0] KEY,
input [17:0] SW,
output [0:6] HEX0,
output [0:6] HEX1,
output [0:6] HEX2,
output [0:6] HEX3,
output [0:6] HEX4,
output [0:6] HEX5,
output [0:6] HEX6,
output [0:6] HEX7,

output [8:0] LEDG,
output [17:0] LEDR,
output UART_TXD,
input UART_RXD,

inout [7:0] LCD_DATA,
output LCD_ON, LCD_BLON, LCD_RW, LCD_EN, LCD_RS,
inout [35:0] GPIO

);


wire signed [31:0] fio1,fio29,fio30;
wire [31:0] fio2,fio3,fio4,fio5,fio12,fio13,fio14,fio17,fio18,fio19,fio21,fio23,
fio24,fio25,fio27;
wire [31:0] fio31,fio32,fio33,fio34,fio35,fio36,fio37,fio38,fio39,fio40,fio41,
```

```
fio42,fio43,fio44,fio45,fio46,fio47,fio48,fio49,fio50,fio51,fio52,fio53,fio54;
wire [31:0] fio55,fio56,fio57,fio58,fio59,fio60,fio61,fio62,fio63,fio64,fio65,
fio66,fio67,fio68,fio69,fio70,fio71,fio72,fio73,fio74,fio75,fio76,fio77,
fio78,fio79,fio80,fio81;
wire signed [31:0] fio82,fio83,fio84;
reg signed [31:0] fio85,fio86,fio87;
wire [15:0] fio7,fio8;
wire [63:0] fio6,fio20,fio26;
wire [3:0] fio_c0,fio_c1;
wire fio28,fioclk,fio_sinal,fio_dir,fio_ativ,fio_clock,fio_trig;
wire [63:0] fio10;
wire [31:0] fio11;
wire [7:0] fio_dado_1,fio_dado_2;
wire [10:0] fio_mod;

assign fio9 = clck;

assign LEDR[5] = SW[5];
assign LEDR[7] = SW[7];
assign LEDR[9] = SW[9];
assign LEDR[10] = SW[10];
assign LEDR[11] = SW[11];
assign LEDR[17] = SW[17];


assign fio_dado_2[0] = GPIO[25];
assign fio_dado_2[1] = GPIO[26];
assign fio_dado_2[2] = GPIO[27];
assign fio_dado_2[3] = GPIO[28];
assign fio_dado_2[4] = GPIO[29];
assign fio_dado_2[5] = GPIO[30];
assign fio_dado_2[6] = GPIO[31];
assign fio_dado_2[7] = GPIO[32];
assign fio_c1[0] = GPIO[20];
```

```
assign fio_c1[1] = GPIO[21];
assign fio_c1[2] = GPIO[22];
assign fio_c1[3] = GPIO[23];
assign fio_trig = GPIO[24];


Tensao_Ent Vent (.sel(SW[5]),.Vin(fio1),.clk(fioclk),.n(SW[4:1]),
.start(SW[17]));

Tensao_EntB Ventb (.sel(SW[10]),.Vin(fio29),.clk(fioclk),.n(SW[4:1]),
.start(SW[17]));

Tensao_EntC Ventc (.sel(SW[11]),.Vin(fio30),.clk(fioclk),.n(SW[4:1]),
.start(SW[17]));

clock_100k c1 (.clock(CLOCK_50),.clock_out(fioclk),.ativ(fio_ativ),
.sel(SW[17]),.cont(fio_mod));

//senoide sen1(.ent(fioclk),.saida(fio31),.ordem(SW[4:1]));

//senoideB sen2(.ent(fioclk),.saida(fio32),.ordem(SW[4:1]));

//senoideC sen3(.ent(fioclk),.saida(fio33),.ordem(SW[4:1]));

//transmissao t1 (.clock1(CLOCK_50),.clock2(fioclk),.dado_in(fio31),
.dado_in2(fio32),.dado_in3(fio33),.dado_out(fio_dado_1),.sinal(fio_sinal),
.start(SW[17]),.trig(fio_trig),.dado_trans(fio_dado_2),.data_out(fio55),
.data_out2(fio56),.data_out3(fio57),.mod(fio_mod),.index_in(fio_c1),
.index_out(fio_c0));

//Linhas L1 (.clk(CLOCK_50),.Tens_in(fio1),.sel(SW[6]),.Zclinha(fio10),
.Iklinha(fio11),.ultim(0),.Vm(fio2),.ikm(fio3),.imk(fio4),.Zcout(fio6),
.Ikout(fio5),.trans(SW[17]));
```

```
//Linhas L2 (.clk(CLOCK_50),.Tens_in(fio2),.sel(SW[9]),
.Zclinha(50'b11100011010111111010100100110001101000000000000000),
.Iklinha(0),.ultim(1),.Vm(fio12),.ikm(fio13),.imk(fio14),.Zcout(fio10),
.Ikout(fio11),.trans(SW[17]));

//Linhas L3 (.clk(CLOCK_50),.Tens_in(fio12),.sel(SW[6]),.Zclinha(fio26),
.Iklinha(fio27),.ultim(0),.Vm(fio17),.ikm(fio18),.imk(fio19),.Zcout(fio20),
.Ikout(fio21),.trans(SW[17]));

//Linhas L4 (.clk(CLOCK_50),.Tens_in(fio17),.sel(SW[9]),
.Zclinha(50'b11100011010111111010100100110001101000000000000000),
.Iklinha(0),.ultim(1),.Vm(fio23),.ikm(fio24),.imk(fio25),.Zcout(fio26),
.Ikout(fio27),.trans(SW[17]));

//Linhas_Tri LT1 (.clk(fioclk),.Tens_inA(fio1),.Tens_inB(fio29),
.Tens_inC(fio30),.Tens_outA(0),.Tens_outB(0),.Tens_outC(0),.sel(SW[12]),
.Iklinha_a(fio58),.Iklinha_b(fio59),.Iklinha_c(fio60),.Iklinha_a2(fio73),
.Iklinha_b2(fio74),.Iklinha_c2(fio75),.Imlinha_a(fio61),.Imlinha_b(fio62),
.Imlinha_c(fio63),.ultim(0),.Vm_a(fio31),.Vm_b(fio32),.Vm_c(fio33),
.ikm_a(fio34),.ikm_b(fio35),.ikm_c(fio36),.imk_a(fio37),.imk_b(fio38),
.imk_c(fio39),.Ikout_a(fio40),.Ikout_b(fio41),.Ikout_c(fio42),.Imout_a(fio46),
.Imout_b(fio47),.Imout_c(fio48),.Vk_a(fio43),.Vk_b(fio44),.Vk_c(fio45),
.prim(1),.paral(1));

//Linhas_Tri LT2 (.clk(fioclk),.Tens_inA(fio43),.Tens_inB(fio44),
.Tens_inC(fio45),.Tens_outA(fio31),.Tens_outB(fio32),.Tens_outC(fio33),
.sel(SW[12]),.Iklinha_a(fio40),.Iklinha_b(fio41),.Iklinha_c(fio42),
.Iklinha_a2(0),.Iklinha_b2(0),.Iklinha_c2(0),.Imlinha_a(fio46),
.Imlinha_b(fio47),.Imlinha_c(fio48),.ultim(0),.Vm_a(fio49),.Vm_b(fio50),
.Vm_c(fio51),.ikm_a(fio52),.ikm_b(fio53),.ikm_c(fio54),.imk_a(fio55),
.imk_b(fio56),.imk_c(fio57),.Ikout_a(fio58),.Ikout_b(fio59),.Ikout_c(fio60),
.Imout_a(fio61),.Imout_b(fio62),.Imout_c(fio63),.Vk_a(),.Vk_b(),.Vk_c(),
.prim(0),.paral(1));
```

```
//Linhas_Tri LT3 (.clk(fioclk),.Tens_inA(fio31),.Tens_inB(fio32),
.Tens_inC(fio33),.Tens_outA(0),.Tens_outB(0),.Tens_outC(0),.sel(SW[13]),
.Iklinha_a(0),.Iklinha_b(0),.Iklinha_c(0),.Iklinha_a2(fio85),.Iklinha_b2(fio86),
.Iklinha_c2(fio87),.Imlinha_a(0),.Imlinha_b(0),.Imlinha_c(0),.ultim(1),
.Vm_a(fio64),.Vm_b(fio65),.Vm_c(fio66),.ikm_a(fio67),.ikm_b(fio68),
.ikm_c(fio69),.imk_a(fio70),.imk_b(fio71),.imk_c(fio72),.Ikout_a(fio73),
.Ikout_b(fio74),.Ikout_c(fio75),.Imout_a(),.Imout_b(),.Imout_c(),.Vk_a(),
.Vk_b(),.Vk_c(),.prim(0),.paral(0));

Transformador t1 (.clk(fioclk),.Vk_a(fio1),.Vk_b(fio29),.Vk_c(fio30),
.iklinha_a(0),.iklinha_b(0),.iklinha_c(0),.Vlowa(fio76),.Vlowb(fio77),
.Vlowc(fio78),.Ia(fio79),.Ib(fio80),.Ic(fio81),.Ika(fio82),.Ikb(fio83),
.Ikc(fio84));

serial s1 (.clock(clck),.clock_2(CLOCK_50),.sinal(fio76),.sinal_out(fio7),
.sel(SW[7]),.trigg(LEDG[1]),.sel2(SW[8]),.sel_trans(fio28));

PortaSerial ps (.clk(CLOCK_50),.RxD(UART_RXD),.SerialOut(fio8),
.flag(LEDG[3]),.SerialOut1(),.TxD_Req(LEDG[1]),.SerialIn(fio7),
.TxD(UART_TXD),.TxBusy(LEDR[0]));

reg clck;
reg bit;
reg [25:0] cont1;

always@(posedge CLOCK_50)
begin
        if(SW[17])
        begin
        cont1 = cont1+1;
        if(cont1>=2171)
        begin
                cont1 = 0;
                clck = clck ^1;
```

```
            end
            fio85 = fio82/10;
            fio86 = fio83/10;
            fio87 = fio84/10;
            end
            else
                  clck = 0;
      end


      assign GPIO[0] = fio_dado_1[0];
      assign GPIO[1] = fio_dado_1[1];
      assign GPIO[2] = fio_dado_1[2];
      assign GPIO[3] = fio_dado_1[3];
      assign GPIO[4] = fio_dado_1[4];
      assign GPIO[5] = fio_dado_1[5];
      assign GPIO[6] = fio_dado_1[6];
      assign GPIO[7] = fio_dado_1[7];
      assign GPIO[8] = fio_c0[0];
      assign GPIO[9] = fio_c0[1];
      assign GPIO[10] = fio_c0[2];
      assign GPIO[11] = fio_c0[3];
      assign GPIO[12] = SW[17];
      assign GPIO[34] = fio_sinal;
      assign GPIO[35] = CLOCK_50;

      assign  LCD_ON          =     1'b1;
      assign  LCD_BLON  =     1'b1;

      assign LEDG[0] = clck;
      assign  LCD_DATA  =     8'hzz;


endmodule
```

ANEXO F – TRANSMISSÃO DE DADOS ENTRE PLACAS FPGA

```
module transmissao (clock1,clock2,dado_in,dado_in2,dado_in3,dado_out,sinal,
start,trig,dado_trans,data_out,data_out2,data_out3,mod,index_in,index_out);

input clock1;
input clock2;
input [31:0] dado_in;
input [31:0] dado_in2;
input [31:0] dado_in3;
input start;
input trig;
input [7:0] dado_trans;
input [10:0] mod;
input [3:0] index_in;
output reg [7:0] dado_out;
output reg sinal;
output reg [31:0] data_out;
output reg [31:0] data_out2;
output reg [31:0] data_out3;
output reg [3:0] index_out;

integer cont1;
integer cont2;
integer cont3;
integer ind;

reg [7:0] dado_ent [49:0];
reg [31:0] dado0;
reg [31:0] dado1;
```

```
reg [31:0] dado2;
reg [31:0] aux;


always@(posedge clock1)
        begin
            if(start)
                begin
                    if(mod == 0)
                        begin
                            cont1 = 0;
                            cont2 = 0;
                        end
                    if(!cont1)
                        begin
                            if(cont2 == 0)
                                begin
                                    index_out = 4'b0100;
                                    dado0 = dado_in;
                                    dado_out = dado0[7:0];
                                    cont2 = cont2 + 1;
                                    sinal = 1;
                                end
                            else if(cont2 == 1)
                                begin
                                    index_out = 4'b0101;
                                    dado_out = dado0[15:8];
                                    cont2 = cont2 + 1;
                                    sinal = 1;
                                end
                            else if(cont2 == 2)
                                begin
                                    index_out = 4'b0110;
                                    dado_out = dado0[23:16];
```

```
                cont2 = cont2 + 1;
                sinal = 1;
            end
        else if(cont2 == 3)
            begin
                index_out = 4'b0111;
                dado_out = dado0[31:24];
                cont2 = cont2 + 1;
                sinal = 1;
            end
        else if(cont2 == 4)
            begin
                index_out = 4'b1000;
                dado1 = dado_in2;
                dado_out = dado1[7:0];
                cont2 = cont2 + 1;
                sinal = 1;
            end
        else if(cont2 == 5)
            begin
                index_out = 4'b1001;
                dado_out = dado1[15:8];
                cont2 = cont2 + 1;
                sinal = 1;
            end
        else if(cont2 == 6)
            begin
                index_out = 4'b1010;
                dado_out = dado1[23:16];
                cont2 = cont2 + 1;
                sinal = 1;
            end
    else if(cont2 == 7)
            begin
```

```
            index_out = 4'b1011;
            dado_out = dado1[31:24];
            cont2 = cont2 + 1;
            sinal = 1;
        end
    else if(cont2 == 8)
        begin
            index_out = 4'b1100;
            dado2 = dado_in3;
            dado_out = dado2[7:0];
            cont2 = cont2 + 1;
            sinal = 1;
        end
     else if(cont2 == 9)
        begin
            index_out = 4'b1101;
            dado_out = dado2[15:8];
            cont2 = cont2 + 1;
            sinal = 1;
        end
     else if(cont2 == 10)
        begin
            index_out = 4'b1110;
            dado_out = dado2[23:16];
            cont2 = cont2 + 1;
            sinal = 1;
        end
     else if(cont2 == 11)
        begin
            index_out = 4'b1111;
            dado_out = dado2[31:24];
            cont2 = cont2 + 1;
            sinal = 1;
        end
```

```
                    else
                        begin
                            sinal = 0;
                            cont1 = 1;
                            index_out = 0;
                        end
                end
                    if(trig)
                        begin
                            case(index_in)
                                4'b0100: dado_ent[0] = dado_trans;
                                4'b0101: dado_ent[1] = dado_trans;
                                4'b0110: dado_ent[2] = dado_trans;
                                4'b0111: dado_ent[3] = dado_trans;
                                4'b1000: dado_ent[4] = dado_trans;
                                4'b1001: dado_ent[5] = dado_trans;
                                4'b1010: dado_ent[6] = dado_trans;
                                4'b1011: dado_ent[7] = dado_trans;
                                4'b1100: dado_ent[8] = dado_trans;
                                4'b1101: dado_ent[9] = dado_trans;
                                4'b1110: dado_ent[10] = dado_trans;
                                4'b1111: dado_ent[11] = dado_trans;
                            endcase
                    data_out = {dado_ent[3],dado_ent[2],dado_ent[1],dado_ent[0]};
                   data_out2 = {dado_ent[7],dado_ent[6],dado_ent[5],dado_ent[4]};
                  data_out3 = {dado_ent[11],dado_ent[10],dado_ent[9],dado_ent[8]};
            end

    end
else
    begin
        cont2 = 0;
          sinal = 0;
     end
```

```
end
endmodule
```

ANEXO G – PARÂMETROS DOS COMPONENTES UTILIZADOS NO SISTEMA DO IEEE POWER SYSTEM RELAYING COMMITTEE

Tabela 12. Parâmetros gerais.

Frequência (Hz)	60
Carga indutiva (mH)	100

Fonte: IEEE Power System Relaying Committee (2005).

Tabela 13. Parâmetros do gerador.

Impedância de sequência positiva (Ω)	6.1 + j16.7
Impedância de sequência negativa (Ω)	2.7 + j8.37

Fonte: IEEE Power System Relaying Committee (2005).

Tabela 14. Parâmetros do transformador.

Tensão – Lado de alta (kV)	229.893
Tensão – Lado de baixa (kV)	22.8
Potência (MVA)	725
Resistência nos enrolamentos – Lado de alta (Ω)	0.1469
Resistência nos enrolamentos – Lado de baixa (Ω)	0.0044

Fonte: IEEE Power System Relaying Committee (2005).

Tabela 15. Parâmetros das linhas.

Condutor	Marigold 1113 kcmil AA
Diâmetro (in)	1.216
Resistência DC (Ω/milha)	0.009222

Fonte: IEEE Power System Relaying Committee (2005).

Tabela 16. Configuração das torres.

Condutor	Separação horizontal da referência (ft)	Altura da torre (ft)	Altura no meio da linha (ft)
1	0.0	100.0	73.0
2	0.0	83.5	56.5
3	0.0	67.0	40.0
4	29.0	67.0	40.0
5	29.0	83.5	56.5
6	29.0	100.0	73.0

Fonte: IEEE Power System Relaying Committee (2005).