Politécnica
UFRJ

SIMULAÇÃO EM TEMPO REAL DE UM SISTEMA DE GERAÇÃO EÓLICA UTILIZANDO FPGA E IMPLEMENTAÇÃO CONTROLLER HARDWARE-IN-THE-LOOP EM DSP

Gabriel Provenzano Cardoso

Projeto de Graduação apresentado ao Curso de Engenharia Elétrica da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheiro.

Orientadores: Robson Francisco da Silva Dias
Felipe Novaes Francis Dicler

Rio de Janeiro
Dezembro de 2024

SIMULAÇÃO EM TEMPO REAL DE UM SISTEMA DE GERAÇÃO EÓLICA UTILIZANDO FPGA E IMPLEMENTAÇÃO CONTROLLER HARDWARE-IN-THE-LOOP EM DSP

Gabriel Provenzano Cardoso

PROJETO DE GRADUAÇÃO SUBMETIDO AO CORPO DOCENTE DO CURSO DE ENGENHARIA ELÉTRICA DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO ELETRICISTA.

Examinado por:

Documento assinado digitalmente
gov.br ROBSON FRANCISCO DA SILVA DIAS
Data: 13/12/2024 19:09:02-0300
Verifique em https://validar.iti.gov.br

Prof. Robson Francisco da Silva Dias, D.Sc.

Documento assinado digitalmente
gov.br FELIPE NOVAES FRANCIS DICLER
Data: 13/12/2024 19:13:54-0300
Verifique em https://validar.iti.gov.br

Eng. Felipe Novaes Francis Dicler, M.Sc.

Documento assinado digitalmente
gov.br ELKIN FERNEY RODRIGUEZ VELANDIA
Data: 13/12/2024 19:57:28-0300
Verifique em https://validar.iti.gov.br

Prof. Elkin Ferney Rodriguez Velandia, D.Sc

Documento assinado digitalmente
gov.br GABRIEL DE SOUZA ANTERO
Data: 13/12/2024 20:20:25-0300
Verifique em https://validar.iti.gov.br

Eng. Gabriel de Souza Antero, B.Sc

RIO DE JANEIRO, RJ – BRASIL
DEZEMBRO DE 2024

Provenzano Cardoso, Gabriel

SIMULAÇÃO EM TEMPO REAL DE UM SISTEMA DE GERAÇÃO EÓLICA UTILIZANDO FPGA E IMPLEMENTAÇÃO CONTROLLER HARDWARE-IN-THE-LOOP EM DSP/Gabriel Provenzano Cardoso. – Rio de Janeiro: UFRJ/ Escola Politécnica, 2024.

XI, 54 p.: il.; 29, 7cm.

Orientadores: Robson Francisco da Silva Dias
Felipe Novaes Francis Dicler

Projeto de Graduação – UFRJ/ Escola Politécnica/ Curso de Engenharia Elétrica, 2024.

Referências Bibliográficas: p. 49 – 50.

1. Simulação em Tempo-Real. 2. Transitórios Eletromagnéticos. 3. Modelo de Máquinas. 4. Análise Nodal Modificada Aumentada. 5. DFIG. 6. FPGA. I. Francisco da Silva Dias, Robson *et al.* II. Universidade Federal do Rio de Janeiro, Escola Politécnica, Curso de Engenharia Elétrica. III. Título.

O glaube,
du warst nicht umsonst geboren!
Hast nicht umsonst gelebt,
gelitten!
Gustav Mahler

Agradecimentos

Resumo do Projeto de Graduação apresentado à Escola Politécnica/ UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenheiro Eletricista.

SIMULAÇÃO EM TEMPO REAL DE UM SISTEMA DE GERAÇÃO EÓLICA UTILIZANDO FPGA E IMPLEMENTAÇÃO CONTROLLER HARDWARE-IN-THE-LOOP EM DSP

Gabriel Provenzano Cardoso

Dezembro/2024

Orientadores: Robson Francisco da Silva Dias
Felipe Novaes Francis Dicler

Curso: Engenharia Elétrica

Este trabalho tem como objetivo a implementação de um modelo de aerogerador no arranjo DFIG (*Doubly-Fed Induction Generator*) para simulação em tempo real. Isso é realizado através de um simulador baseado em FPGA (*Field-Programmable Gate Array*), cuja estrutura altamente paralelizável permite a implementação de modelos eletromagnéticos e de elementos com frequência de chaveamento elevada, resolvendo todas as equações que modelam o sistema dentro de um intervalo de microssegundos. O controle do conversor *Back-to-Back* é baseado em controle PI e implementado em um DSP (*Digital Signal Processor*). A interface entre o modelo da planta na FPGA e o controle do conversor na DSP caracteriza a topologia *Controller Hardware-in-the-loop* (C-HIL), onde o controle da planta pode ser avaliado em tempo real de uma forma barata e fidedigna com sua implementação física. O modelo da máquina de indução é baseado na Teoria Generalizada de Máquinas, que utiliza a Transformada de Park para transformar os eixos de fase em eixos direto e de quadratura. A solução das equações de circuitos é feito com a Análise Nodal Modificada Aumentada, que comporta modelos de chaves para encontrar as tensões nodais do sistema. Os resultados da simulação em tempo real são comparados com simulações realizadas no software de simulação offline PSCAD.

Abstract of Undergraduate Project presented to POLI/UFRJ as a partial fulfillment of the requirements for the degree of Engineer.

REAL-TIME SIMULATION OF A WIND GENERATION SYSTEM USING FPGA AND HARDWARE-IN-THE-LOOP CONTROLLER IMPLEMENTATION ON DSP

Gabriel Provenzano Cardoso

December/2024

Advisors: Robson Francisco da Silva Dias
Felipe Novaes Francis Dicler

Course: Electrical Engineering

This work aims to implement a wind generator model in the DFIG (*Doubly-Fed Induction Generator*) topology for real-time simulation. This is done through a simulator based on FPGA (*Field-Programmable Gate Array*), whose highly parallelizable structure allows the implementation of electromagnetic models and elements with high switching frequency, solving all the equations that model the system within of a microsecond interval. The *Back-to-Back* converter control is based on PI control and implemented in a DSP (*Digital Signal Processor*). The interface between the plant model in the FPGA and the converter control in the DSP characterizes the *Controller Hardware-in-the-loop* (C-HIL) topology, where the plant control can be evaluated in real time in a cheap and reliable with its physical implementation. The induction machine model is based on the Generalized Theory of Electrical Machines, which uses the Park Transform to transform the phase axes into direct and quadrature axes. The solution of the circuit equations is done with the Augmented Modified Nodal Analysis, which includes switch models to find the nodal voltages of the system. Real-time simulation results are compared with simulations performed in PSCAD offline simulation software.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

As redes elétricas, abrangendo os setores de geração, transmissão e distribuição de energia elétrica, são sistemas de grande escala descritos por equações não-lineares, resultando em comportamento dinâmico complexo [1]. A modelagem detalhada e análise desses sistemas demandam ferramentas computacionais avançadas para garantir uma representação precisa. Programas de simulação de transitórios eletromagnéticos surgem como exemplos fundamentais para uma avaliação detalhada da performance temporal desses sistemas de potência sob diversas condições operativas.

Na indústria, destacam-se versões consolidadas de programas de simulação de transitórios eletromagnéticos, tais como ATP-EMTP [2], EMTP-RV [3] e EMTDC/PSCAD [4]. Estas ferramentas desempenham um papel crucial ao proporcionar uma base sólida para a análise contínua da eficiência e da estabilidade das redes elétricas, contribuindo assim para a operação segura e eficaz desses sistemas em larga escala.

Essas ferramentas realizam simulações *offline*, o que significa que o tempo de simulação não está necessariamente sincronizado com o tempo físico, podendo ser maior ou menor que este. Para conduzir testes em malha fechada de equipamentos e controles, avaliando sua performance em condições reais, é essencial que o tempo de simulação esteja sincronizado com o tempo físico, para que o controle real seja de fato implementado, caracterizando assim a simulação em tempo real [5]. Este avanço foi impulsionado principalmente pelo surgimento de processadores digitais de ponto flutuante com frequência de operação de até 40MHz [6]. O pioneiro neste campo foi o RSCAD, do simulador RTDS, originado do EMTDC/PSCAD no Manitoba HVDC Research Centre. Este simulador em tempo real permite emular sistemas físicos específicos, testando funcionalidades e avaliando respostas sob condições determinadas. Outros softwares para simulação em tempo real amplamente utilizados no mercado atual incluem o RT-LAB e o Hypersim, ambos associados ao simulador OPAL-RT. Essas ferramentas desempenham um papel crucial ao proporcionar uma representação mais próxima da realidade, contribuindo para a validação

eficaz de equipamentos complexos com alto detalhamento e controles em ambientes operacionais reais.

Neste contexto, embora exista uma ampla variedade de simuladores em tempo real disponíveis na indústria, o alto custo dessas plataformas muitas vezes representa um desafio para o desenvolvimento. Assim, uma alternativa mais acessível e que oferece os recursos necessários para a implementação de modelos em tempo real é a FPGA (*Field Programmable Gate Array*). Essas placas são compostas por blocos lógicos básicos interconectados por uma matriz de fios e chaves programáveis, o que proporciona programabilidade a nível de hardware. Essa característica garante uma estrutura altamente paralelizável e flexível [7], tornando a FPGA ideal para simulações em tempo real. Embora versões mais econômicas de FPGA possuam capacidades inferiores às dos simuladores comerciais, isso geralmente não representa um problema em aplicações de menor escala.

Uma aplicação significativa da simulação em tempo real é o conceito de *Controller Hardware-in-the-loop* (C-HIL), no qual o simulador em tempo real, como uma FPGA, é integrado a um controlador físico embarcado em um processador de sinal digital (*Digital Signal Processor*, DSP), por exemplo. Essa abordagem possibilita testar o controle desejado de maneira mais econômica e em condições que poderiam ser inviáveis no sistema físico. Essa adaptação eficaz do *Harware-in-the-loop* (HIL) permite explorar cenários desafiadores de forma acessível e eficiente, destacando a versatilidade e potencial das FPGAs em aplicações práticas.

As plantas simuladas abrangem uma ampla gama, desde sistemas de potência multiáreas até fenômenos com frequências da ordem de dezenas de kilohertz, como os transientes eletromagnéticos [5]. A introdução de novas tecnologias nos sistemas de potência traz consigo desafios operacionais significativos que exigem uma avaliação minuciosa para mitigar ao máximo seus impactos sistêmicos. A crescente integração, por exemplo, de geração por meio de fontes renováveis intermitentes, como as eólicas e as fotovoltaicas, demanda a aplicação da eletrônica de potência para controlar suas grandezas elétricas e ajustá-las para a interface com o restante do sistema elétrico. Portanto, é crucial uma simulação precisa e abrangente para compreender as complexidades decorrentes dessas mudanças tecnológicas.

A modelagem desses sistemas heterogêneos e dinâmicos permite uma análise detalhada dos efeitos sobre o desempenho global, contribuindo para a identificação de soluções eficientes e a minimização de riscos operacionais. Assim, a simulação desempenha um papel essencial na antecipação e gestão proativa dos desafios associados à evolução dos sistemas de potência, promovendo a resiliência e a eficiência em face das transformações tecnológicas.

Neste cenário, a energia eólica no Brasil destaca-se como uma das tecnologias de geração em maior expansão. Conforme indicam dados da ABEEólica [8], a capaci-

dade instalada de parques eólicos *onshore* já ultrapassa os 21 GW, consolidando o país como o terceiro maior em capacidade instalada no mundo. Globalmente, o progresso da tecnologia *offshore* impulsiona ainda mais o avanço dessa forma de geração, sendo que, em 2022, essa modalidade representou expressivos 20% dos investimentos totais em geradores eólicos [9].

As tecnologias de geração eólica podem ser de dois tipos: velocidade fixa e velocidade variável. Os primeiros eram muito usados nos anos 1980 e 1990 e operavam a uma velocidade constante, equivalente a frequência da rede elétrica. No entanto, atualmente, a tecnologia mais usada é o aerogerador de velocidade variável, que foi possível com o avanço da eletrônica de potência. Assim, o gerador é interfaceado com um conversor, que irá controlar, dentre outras grandezas, a velocidade de rotação do rotor, permitindo seu acoplamento com a rede elétrica. O conversor pode ser usado de duas maneiras: isolando completamente o gerador da rede ou conectando apenas o rotor da máquina ao conversor, com o estator ligado diretamente à rede. Na primeira configuração, são utilizados geradores síncronos de ímã permanente; já na segunda, chamada Gerador de Indução Duplamente Alimentado (DFIG, do inglês *Doubly-Fed Induction Generator*), é empregado um gerador de indução de rotor bobinado. Como o rotor da máquina de indução não gira na velocidade síncrona, um conversor AC/DC/AC (*Back-to-Back*) com um elo CC é necessário para sincronizar a frequência do rotor com a frequência da rede elétrica. O presente trabalho terá foco na modelagem da topologia do aerogerador DFIG.

Dessa forma, para que um projeto de controle do chaveamento das chaves IGBT do conversor *Back-to-Back* seja avaliado, é possível utilizar a simulação em tempo real implementando o controle físico com um modelo da DFIG em tempo real utilizando o hardware FPGA para tanto.

1.1 Objetivos

Este trabalho possui como principal objetivo o desenvolvimento de um modelo de gerador eólico na configuração DFIG e sua implementação para simulação em tempo real para análise de transitórios eletromagnéticos. Esse modelo deve ser capaz de resolver todas as equações que o descreve em um curto intervalo de tempo, de forma a possibilitar a sincronização com tempo físico, que caracteriza a simulação em tempo real.

A implementação do modelo ocorrerá por meio da configuração C-HIL. A FPGA é responsável por conter e resolver todas as equações que descrevem o aerogerador, enquanto o controle do conversor é implementado em uma DSP, configurando assim a implementação do C-HIL.

Os resultados obtidos a partir da simulação em tempo real são comparados com

o software de simulação *offline* PSCAD, reconhecido como referência na indústria e em estudos de transitórios eletromagnéticos.

1.2 Organização

O presente trabalho está dividido em 5 capítulos além da presente introdução. No capítulo 2 é apresentado a fundamentação teórica necessária para o desenvolvimento do trabalho. Ele engloba os conceitos que caracterizam a simulação em tempo real, uma apresentação da plataforma de simulação FPGA, além da formulação da Teoria Generalizada de Máquinas e da Análise Nodal Modificada Aumentada, necessários para compreender a modelagem da DFIG. O capítulo 3 descreve a implementação do modelo da DFIG apresentado no capítulo anterior na FPGA e o controle do conversor no DSP. Os resultados encontrados e a comparação com a simulação *offline* são apresentados no capítulo 4. Finalmente, o capítulo 5 se desenvolve acerca das conclusões finais do trabalho e a proposição de trabalhos futuros que poderão ser realizados a partir deste.

Capítulo 2

Fundamentação Teórica

2.1 Simulação em tempo real

A simulação em tempo real é uma técnica de simulação digital onde o tempo é discretizado com um passo de tempo constante. Nessa abordagem, os resultados da simulação para uma dada iteração devem estar disponíveis antes do término do passo de simulação, mas apenas são exportados ao seu fim [5].

Seja T_i^e o tempo de execução do modelo implementado em uma iteração i e h o passo de simulação. Podemos definir quatro estados para um dado instante da simulação:

$$\text{state} = \begin{cases} \text{start}, & \text{se o comando de começo da simulação ainda não foi dado} \\ \text{solve}, & \text{se } t < T_i^e \text{ em uma iteração } i \text{ para } t < h \\ \text{idle}, & \text{se } t > T_i^e \text{ em uma iteração } i \text{ para } t < h \\ \text{error}, & \text{se } h < T_i^e \text{ em uma iteração } i \end{cases}$$

O estado inicial, denominado *start*, representa o momento em que a simulação aguarda o sinal de comando para ser iniciada. Em seguida, o estado *solve* é acionado simultaneamente ao início de cada iteração, durante o qual todas as equações que modelam o sistema em questão são calculadas. É crucial que essas equações sejam resolvidas em um intervalo de tempo inferior ao passo de integração ($T_i^e \leq h \, \forall i$), caso contrário, a simulação entra em um estado de erro, representada pelo estado *error*.

Se as saídas do sistema estão disponíveis antes do término do passo de integração, a simulação entra no estado de *idle*, aguardando o final da iteração atual para iniciar a próxima. No entanto, o sinal do simulador só é exportado ao final do estado de *idle*. A Figura 2.1 ilustra o processo de exportação de resultados em uma simulação

em tempo real.

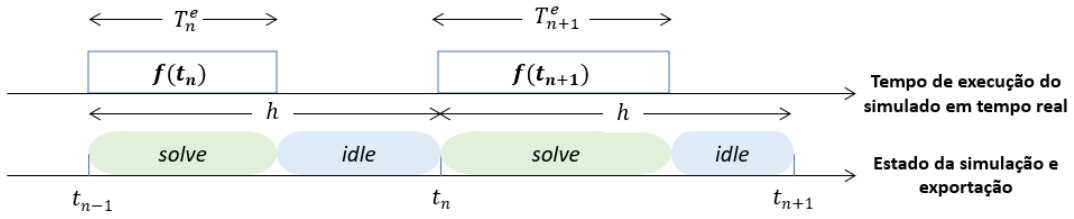

Figura 2.1: Diagrama com estados da simulação em tempo real.

O propósito das simulações em tempo real é testar equipamentos elétricos sob condições que imitam o mais próximo possível as condições naturais, como se estivessem integrados aos sistemas físicos reais aos quais estão associados. Nesse contexto, é imperativo que o simulador em tempo real reproduza de maneira precisa o comportamento dinâmico do sistema a ser controlado. Essa exigência implica a necessidade de considerável poder de processamento para resolver um grande número de equações em um curto passo de simulação. Para simulações de transitórios eletromagnéticos, o passo de integração deve estar na ordem de dezenas de microssegundos, permitindo a representação de transitórios na ordem de kilohertz [5].

As plataformas mais comuns para simulação em tempo real são as CPUs, utilizando processadores de multi-núcleo e supercomputadores. No entanto, a limitada capacidade de paralelização nas CPUs resulta em simuladores onerosos. Uma abordagem mais recente na simulação em tempo real envolve o uso de FPGAs [5], que apresentam alto grau de paralelismo e proporcionam considerável flexibilidade para o desenvolvimento de algoritmos especializados. Vale ressaltar que as FPGAs demandam maior tempo de desenvolvimento, uma vez que é necessário utilizar linguagens de descrição de hardware (HDL, do inglês *Hardware Description Language*) e realizar a síntese e o mapeamento do circuito projetado para a FPGA.

A simulação em tempo real de um sistema elétrico a ser controlado passa pelos seguintes passos:

1. Fase de modelagem, onde há a descrição das equações a serem modeladas;
2. Fase de concepção, onde são escolhidas especificações como discretização e o design do modelo no simulador;
3. Fase de implementação em tempo real.

Uma das abordagens primordiais para implementar simulações em tempo real é por meio da técnica conhecida como *Hardware-in-the-loop* (HIL). Nesse método,

os sinais reais provenientes de um controlador são vinculados a um simulador, que replica em tempo real o comportamento da planta física. Para o controlador, não há distinção entre estar conectado ao simulador ou à planta física; em ambos os casos, seu comportamento é idêntico. Essa equivalência possibilita a avaliação de uma ampla variedade de cenários, uma proeza impossível de ser realizada se o controlador estivesse conectado apenas à planta física. A interação entre o simulador e o controlador no contexto do HIL é representada na Figura 2.2.

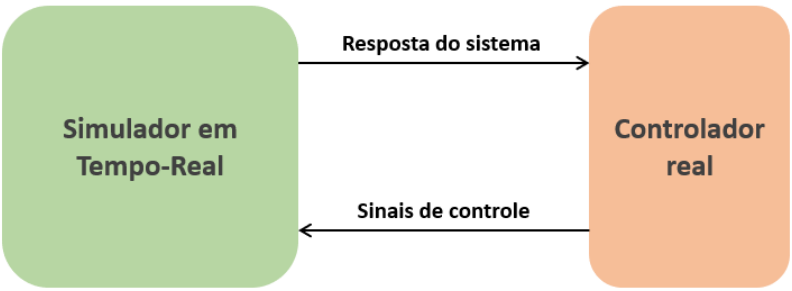

Figura 2.2: Esquema da configuração HIL.

2.2 Arquitetura da FPGA

A implementação digital de sistemas pode ser realizada por meio de microprocessadores, hardwares reconfiguráveis ou circuitos integrados de aplicação específica (ASIC, do inglês, *Application-specific integrated circuit*). Cada uma dessas abordagens possui características distintas que as tornam mais adequadas para determinadas aplicações. Entre essas tecnologias, os hardwares reconfiguráveis, notadamente as FPGAs, ocupam uma posição estratégica ao equilibrar a flexibilidade dos microprocessadores com o desempenho elevado dos ASICs.

As FPGAs consistem tipicamente em blocos lógicos programáveis interconectados por uma matriz de fios e chaves programáveis, permitindo uma grande flexibilidade na definição das funções que o hardware pode desempenhar [7]. Essa flexibilidade é alcançada através da reprogramação das interconexões e das funções lógicas dos blocos, o que pode ser feito diversas vezes ao longo do ciclo de vida do dispositivo.

A programabilidade das FPGAs pode ser classificada em três tipos principais:

1. **Baseadas em antifusíveis:** Essas FPGAs são programáveis uma única vez, onde as conexões são feitas de forma permanente.

2. **Baseadas em memória estática (SRAM):** Permitem múltiplas reprogramações e são rápidas, mas requerem uma fonte de alimentação constante para

manter a configuração.

3. **Baseadas em memória flash:** Também reprogramáveis, com a vantagem adicional de manterem a configuração mesmo sem alimentação elétrica [10].

Uma característica distinta e altamente vantajosa das FPGAs é sua estrutura maciçamente paralelizável. Isso decorre da capacidade de particionar uma grande quantidade de unidades paralelas, permitindo o processamento simultâneo de múltiplas operações. Enquanto CPUs e DSPs são dispositivos sequenciais, seguindo uma orientação temporal, as FPGAs podem ser concebidas como orientadas espacialmente. Em outras palavras, cada processo é executado em uma unidade de hardware em paralelo, o que pode resultar em um aumento significativo de desempenho para aplicações que podem ser paralelizadas.

A programação paralela oferece uma técnica valiosa conhecida como *pipeline*. Nela, uma função é dividida em diversos estágios, garantindo que o fluxo de dados persista em todos os ciclos de clock. Esse método é viabilizado pela introdução de registradores que armazenam os valores obtidos em cada estágio, proporcionando eficiência e otimização do processamento.

Além disso, as FPGAs podem ser utilizadas para implementar algoritmos complexos que demandariam um grande poder de processamento em CPUs convencionais. Aplicações típicas incluem processamento digital de sinais, criptografia, inteligência artificial e controle de sistemas em tempo real [10]. A capacidade de customizar o hardware para otimizar essas operações torna as FPGAs uma escolha atraente para muitas aplicações avançadas.

2.2.1 Fluxo de design

A implementação de um sistema digital em uma FPGA é um processo que exige o uso de Linguagens de Descrição de Hardware (HDL - *Hardware Description Language*) para descrever o comportamento do sistema. As duas linguagens mais comuns são o VHDL e o Verilog, que permitem o desenvolvimento tanto a nível de arquitetura quanto a nível lógico. Essas linguagens possibilitam uma descrição detalhada dos componentes e das interconexões do sistema digital.

Para componentes de hardware como blocos de memória e operações aritméticas, é comum utilizar *IP Cores* (*Intellectual Property Cores*) fornecidos pelos próprios fabricantes das FPGAs. Esses IP Cores são amplamente testados e otimizados para utilização na FPGA, podendo ser importados para o design pelo usuário. No entanto, o usuário não tem acesso ao código que descreve o IP importado, apenas à sua interface e funcionalidades.

Após a descrição do sistema em HDL, realiza-se uma simulação funcional para verificar o funcionamento da arquitetura antes do processo de síntese. Isso é feito

através de uma *testbench*, onde são inseridos os estímulos que o design deve receber, como o sinal de clock e a inicialização de variáveis. A *testbench* mostra temporalmente os sinais desejados, permitindo a verificação do comportamento do sistema. Esta etapa é crucial para validar o modelo antes de seguir para etapas subsequentes que consomem muito tempo, como a síntese e a implementação.

Na etapa de síntese, o código fonte descrito em HDL é traduzido para um formato de *netlist*, que representa o sistema digital em termos de componentes lógicos básicos, como portas lógicas e multiplexadores. A síntese converte a descrição de alto nível em um conjunto de primitivas lógicas que podem ser implementadas na FPGA.

Com o fim da síntese, inicia-se o processo de implementação, que pode ser dividido em três sub-processos: *Map*, *Place* e *Route*:

1. *Map:* O design é subdividido em blocos lógicos básicos, como BRAM (memória de acesso aleatório embutida), DSP (processadores de sinal digital) e IOE (elementos de entrada/saída).

2. *Place:* Os blocos lógicos são designados a localizações físicas específicas na FPGA.

3. *Route:* As interconexões entre os blocos lógicos são determinadas, definindo os caminhos que os sinais percorrerão na FPGA.

Durante a implementação, é possível verificar se as restrições de *timing* são atendidas, garantindo que o sistema funcionará corretamente nas frequências de operação desejadas.

A etapa final é a programação do dispositivo. Nessa fase, a arquitetura resultante da implementação é convertida para um formato aceito pela FPGA. Esse formato, chamado de arquivo de *bitstream*, é carregado na FPGA, configurando-a para executar a função desejada.

A Figura 2.3 apresenta um resumo de todo o fluxo de design na FPGA, ilustrando as principais etapas desde a descrição do sistema até a programação do dispositivo.

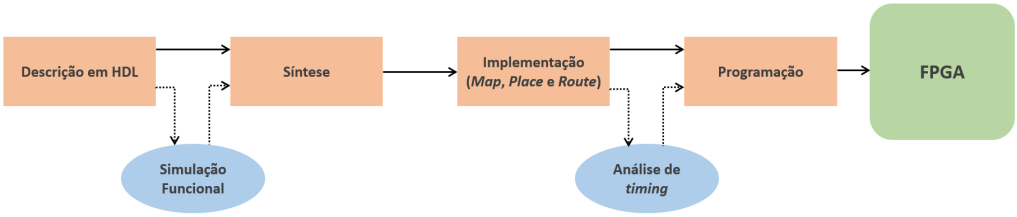

Figura 2.3: Diagrama com as etapas do fluxo de design na FPGA.

2.2.2 *High-Level Synthesis*

O *High-Level Synthesis* (HLS) é uma poderosa ferramenta que transforma código escrito em linguagens de alto nível, como C ou C++, em uma implementação RTL (*Register Transfer Level*), traduzida em código HDL. Dessa forma, é possível exportar essa implementação RTL de maneira protegida para a arquitetura em HDL, acelerando significativamente o fluxo de design na etapa de descrição em HDL e permitindo a implementação de funções em alto nível.

A principal vantagem do HLS é a aceleração no tempo de desenvolvimento. No entanto, essa ferramenta oferece outros benefícios valiosos para o desenvolvimento de sistemas digitais em hardware. Embora o código seja escrito em uma linguagem de alto nível, as ferramentas de HLS permitem explorar o paralelismo utilizando métodos como *pipeline*, *loop unrolling* e o uso de diversos componentes de hardware para paralelizar iterações. Isso proporciona grande flexibilidade na busca por um ponto ótimo entre a velocidade de execução e os recursos de hardware utilizados. Com um código eficientemente estruturado, é possível gerar designs que se aproximam do desempenho de códigos HDL escritos manualmente.

Além de acelerar o desenvolvimento, o HLS melhora a produtividade dos designers, permitindo que engenheiros com experiência em software contribuam para o desenvolvimento de hardware sem a necessidade de um conhecimento profundo de HDL. Isso amplia a base de desenvolvedores que pode trabalhar em projetos de hardware, facilitando a integração de algoritmos complexos e o desenvolvimento de protótipos rápidos.

No entanto, o HLS apresenta algumas limitações. O código deve seguir um estilo específico de escrita para que o paralelismo seja efetivamente explorado. Sem esse conhecimento, é fácil obter uma implementação ineficiente. Além disso, recursos comuns no desenvolvimento de software, como recursão ou algoritmos baseados em ponteiros, não resultam em uma boa síntese no HLS [11]. Portanto, é crucial que os desenvolvedores compreendam as melhores práticas e limitações do HLS para maximizar seu potencial.

Fluxo de design no HLS

Diferentes fabricantes possuem diferentes fluxos de design para suas ferramentas de HLS. Neste trabalho, foi utilizado o *Vitis HLS*, da Xilinx.

O fluxo de design no HLS começa com a descrição do design em linguagem C/C++. Esse processo deve seguir uma determinada forma que permita sua futura transcrição para descrição em hardware. É essencial possuir conceitos de hardware bem definidos para tomar decisões como os tipos de dados, que podem gerar uma diferença significativa na sua representação em hardware.

Após descrever a funcionalidade do hardware, é possível realizar uma simulação em C/C++ para validar o algoritmo implementado. Nesta etapa, a função *main* é considerada como um *testbench* pelo qual o código é executado. Embora essa etapa seja opcional, ela é fundamental para garantir que o código C/C++ se comporte da maneira esperada. Uma vez garantido que o código possui a funcionalidade desejada, é possível adicionar diretivas, que são otimizações específicas para o hardware ao código C/C++. Isso é necessário, pois o compilador C não consegue interpretar determinados conceitos aplicados a nível de hardware.

Em seguida, o código está pronto para a síntese, onde será obtido um *IP Core* descrito em HDL. Esse IP será futuramente exportado e adicionado ao código em HDL como uma caixa preta, sem conhecimento da estrutura do seu código interno. Na síntese, é possível obter relatórios detalhados, como utilização de *timing* e utilização de recursos, que fornecem uma visão abrangente sobre a eficiência e viabilidade do design.

Após a síntese, outro passo opcional é a co-simulação, onde o código HDL gerado será utilizado juntamente com o *testbench* em C/C++ para realizar uma co-simulação entre hardware e software. Nesta fase, o comportamento da simulação é muito próximo ao do implementado finalmente na FPGA. Embora essa etapa seja temporalmente custosa, ela é crucial para garantir a performance desejada na implementação final em hardware.

A etapa final consiste na geração do IP, que será utilizado na etapa de síntese da FPGA como uma caixa preta que desempenhará a funcionalidade desejada a partir de entradas digitais. A Figura 2.4 ilustra o processo de fluxo de design do HLS.

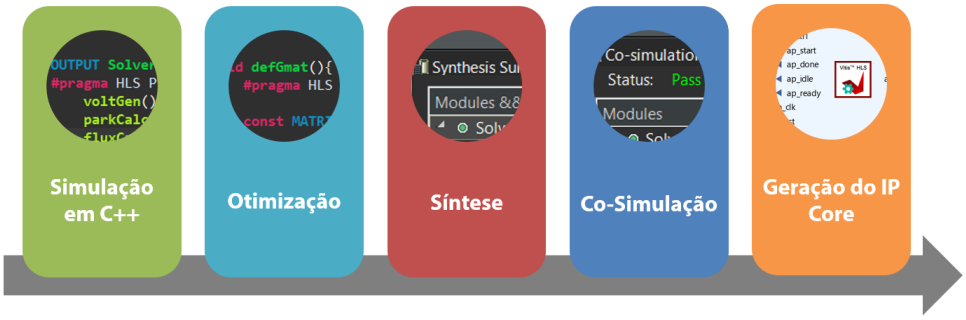

Figura 2.4: Fluxo de design no HLS.

Embora o IP Core gerado a partir do processo de *High-Level Synthesis* possa ter um consumo de recursos maior e uma eficiência menor do que a mesma funcionalidade implementada puramente em HDL, ele proporciona uma significativa aceleração no tempo de desenvolvimento. Por essa razão, é comum que parte da arquitetura seja feita através do processo de HLS e parte da arquitetura seja feita em HDL, onde IPs com determinadas funcionalidades podem ser implementados

de forma mais otimizada. Desta forma, o HLS permite uma combinação de desenvolvimento rápido e otimização detalhada, criando um equilíbrio entre tempo de desenvolvimento e eficiência do hardware.

2.3 O Gerador de Indução Duplamente Alimentado

A utilidade da geração eólica é converter a energia cinética do vento em energia elétrica. Para tanto, o sistema de geração eólica é normalmente composto pelos seguintes elementos: turbina eólica, caixa de engrenagens, gerador elétrico, conversor de potência e transformador. Embora o conversor seja opcional, ele permite configurações que dão ao sistema de geração mais flexibilidade operativa.

Os primeiros aerogeradores a surgirem foram os de velocidade fixa. Nele, o gerador opera sempre na velocidade síncrona, independente da velocidade do vento. É possível ainda colocar um conversor de eletrônica de potência antes da conexão com a rede para regular a tensão e um banco de capacitores para diminuir o consumo de potência reativa [12]. Essa configuração costuma utilizar um gerador de indução gaiola de esquilo e possui uma construção mais simples e barata comparado aos geradores de velocidade variável. No entanto, mudanças na velocidade do vento se traduzem em flutuações mecânicas e, portanto, elétricas. Além disso, não suportam nenhum tipo de controle de velocidade para rastreamento de potência máxima e também devem ser capazes de suportar um alto estresse mecânico.

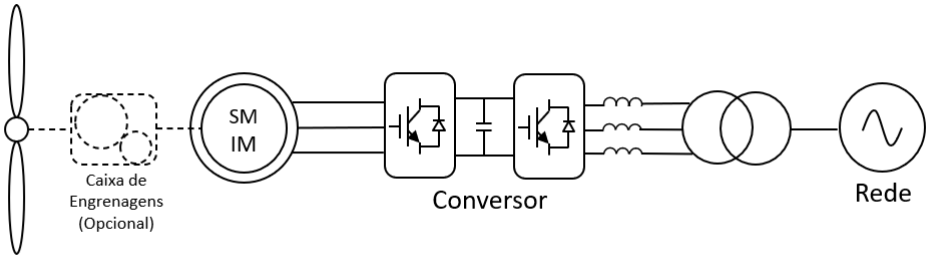

Figura 2.5: Esquema da topologia *full-scale*.

Dessa forma, o tipo de geração eólica mais utilizado atualmente é o com velocidade variável, que permite o ajuste da velocidade do rotor para diferentes velocidades do vento. Uma vez que nesse caso o rotor não gira na velocidade síncrona, é mandatório o uso de um conversor de eletrônica de potência para conectá-lo à rede elétrica. Desses, dois tipos são mais comuns: com conversores *full-scale* e com conversor *partial-scale*. A primeira pode ser atingida através de um conversor *Back-to-Back* conectado a um gerador de indução gaiola de esquilo ou a um gerador síncrono. Essas topologias, no entanto, necessitam de um conversor maior, uma vez

que ele processará toda a energia do gerador nele acoplado. A Figura 2.5 mostra este esquema.

Já a topologia com conversor *partial-scale* permite o uso de um conversor com potência nominal cerca de 30% menor que a do gerador associado [12]. Ela é baseada no gerador de indução duplamente alimentado (DFIG - *Double-Fed Induction Generator*), que usa um gerador de indução de rotor bobinado com o estator conectado diretamente à rede elétrica e o rotor conectado ao conversor *Back-to-Back*, como observado na Figura 2.6. A utilização de um conversor menor e a possibilidade de operação nos quatro quadrantes de potência são algumas das principais vantagens da topologia DFIG.

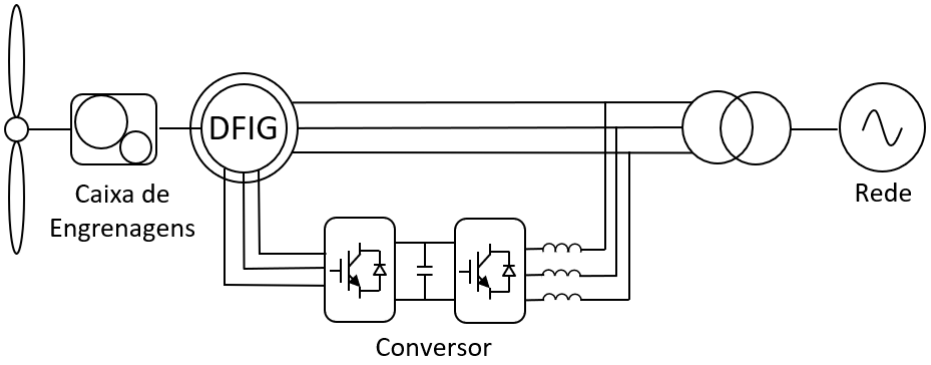

Figura 2.6: Esquema da topologia da DFIG.

A DFIG oferece diversas vantagens adicionais, como a capacidade de controle independente de potência ativa e reativa, o que melhora a estabilidade e a eficiência do sistema. Além disso, a DFIG permite um maior aproveitamento da energia eólica ao operar em uma faixa de velocidades variáveis, maximizando a captura de energia através de técnicas de controle avançadas, como o rastreamento do ponto de potência máxima (MPPT - *Maximum Power Point Tracking*). Essa tecnologia também proporciona uma resposta mais rápida a variações na velocidade do vento, o que é essencial para manter a estabilidade da rede elétrica.

Em comparação com os geradores de velocidade fixa, a DFIG também oferece uma redução significativa nos esforços mecânicos sobre a turbina eólica, aumentando sua vida útil e reduzindo os custos de manutenção. As configurações modernas de DFIG incorporam técnicas avançadas de controle e proteção, garantindo uma operação segura e eficiente mesmo em condições de vento turbulento e em situações de falhas na rede elétrica [13].

2.3.1 Teoria Generalizada de Máquinas

A Teoria Generalizada de Máquinas surgiu como uma abordagem para representar máquinas elétricas de forma simplificada e eficiente. Inicialmente, desenvolveu-se para modelar uma máquina idealizada de dois polos [14]. Neste contexto, cada enrolamento é equivalente a uma bobina, permitindo a representação sistemática de máquinas elétricas complexas.

A representação comum utiliza o modelo da máquina primitiva de Kron [15], ilustrado na Figura 2.7. Este modelo emprega quatro eixos fundamentais: D, Q, F e G. Os eixos D e Q são rotativos, enquanto F e G são estacionários, mantendo uma defasagem constante de 90° entre si. A máquina primitiva serve como um modelo fundamental que simplifica e estende a análise de sistemas multifásicos e interconectados.

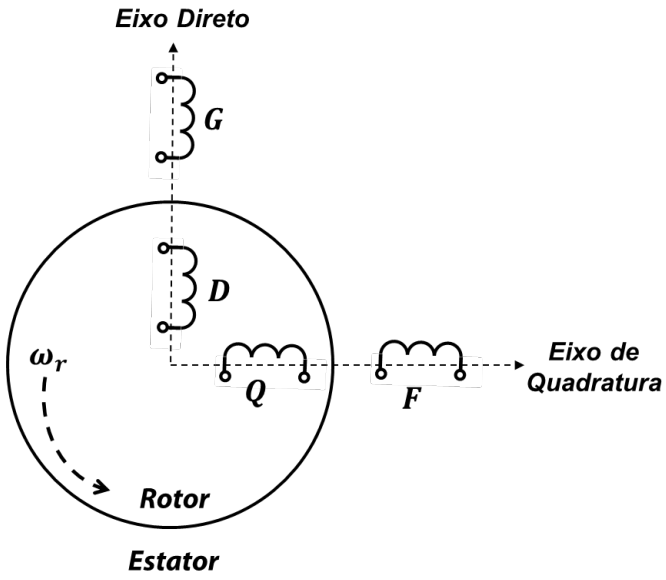

Figura 2.7: Diagrama da máquina primitiva de Kron.

A escolha da representação dos enrolamentos é crucial para definir a notação. Para uma modelagem trifásica da máquina de indução de rotor bobinado, por exemplo, são necessários seis enrolamentos: três no estator e três no rotor. No entanto, é possível utilizar a Transformada de Park [4] para mudar do sistema de referência trifásico (abc) para o sistema de eixos de Kron (direto e de quadratura).

Para tanto, são necessárias duas transformações, conforme (2.1), devido à necessidade de dois eixos dq, um para o estator e outra para o rotor.

$$\mathbf{T_s} = \frac{2}{3}\begin{bmatrix} cos(\theta) & cos(\theta - \frac{2\pi}{3}) & cos(\theta + \frac{2\pi}{3}) \\ -sen(\theta) & -sen(\theta - \frac{2\pi}{3}) & -sen(\theta + \frac{2\pi}{3}) \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \tag{2.1a}$$

$$\mathbf{T_r} = \frac{2}{3}\begin{bmatrix} cos(\theta - \theta_r) & cos(\theta - \theta_r - \frac{2\pi}{3}) & cos(\theta - \theta_r + \frac{2\pi}{3}) \\ -sen(\theta - \theta_r) & -sen(\theta - \theta_r - \frac{2\pi}{3}) & -sen(\theta - \theta_r + \frac{2\pi}{3}) \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \tag{2.1b}$$

onde θ e θ_r são os deslocamentos dos eixos do estator e do rotor, respectivamente.

Observa-se que as transformadas apresentadas geram três sistemas de eixos, pois o eixo de sequência zero é adicionado aos eixos direto e de quadratura. No entanto, caso os enrolamentos da máquina estejam conectados em estrela e não aterrados, a componente de sequência zero será sempre nula [4], não necessitando ser considerada.

Para retornar ao sistema de referência trifásico (abc), aplica-se a transformada inversa de Park, que é a inversa das matrizes de (2.1). As equações abaixo mostram as equações trifásicas de tensão nos enrolamentos do estador. O análogo ocorre no rotor.

$$v_s^a = R_a i_s^a + \frac{d\lambda_s^a}{dt}. \tag{2.2a}$$

$$v_s^b = R_b i_s^b + \frac{d\lambda_s^b}{dt}. \tag{2.2b}$$

$$v_s^c = R_c i_s^c + \frac{d\lambda_s^c}{dt}. \tag{2.2c}$$

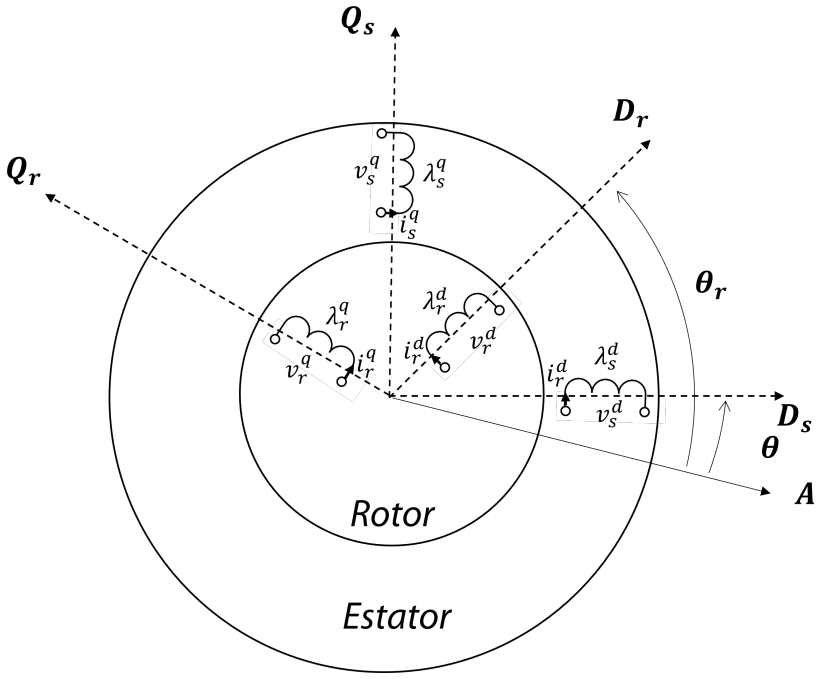

Figura 2.8: Modelo dq da máquina de indução de rotor bobinado.

A Figura 2.8 mostra os dois enrolamentos dq provenientes da representação da máquina de Kron. As equações de tensão nos enrolamentos, tanto estáticos quanto

rotativos, são fundamentais para descrever o comportamento da máquina. As equações [16] para os enrolamentos estáticos e rotativos são dadas por:

$$v_s^d = R_s i_s^d + \omega \lambda_s^q + \frac{d\lambda_s^d}{dt}. \tag{2.3a}$$

$$v_s^q = R_s i_s^q - \omega \lambda_s^d + \frac{d\lambda_s^q}{dt}. \tag{2.3b}$$

$$v_r^d = R_r i_r^d + (\omega - \omega_r) \lambda_r^q + \frac{d\lambda_r^d}{dt}. \tag{2.4a}$$

$$v_r^q = R_r i_r^q - (\omega - \omega_r) \lambda_r^d + \frac{d\lambda_r^q}{dt}. \tag{2.4b}$$

onde λ^ϕ é o fluxo concatenado no enrolamento equivalente do eixo ϕ, i^ϕ é a corrente no enrolamento equivalente do eixo ϕ, R_s é a resistência dos enrolamentos do estator e R_r é a resistência dos enrolamentos do rotor.

As velocidades angulares ω e ω_r são respectivamente as velocidades angulares do eixo dq do estator e do eixo dq do rotor e são dados pela derivada da diferença angular entre o respectivo eixo direto e a referência.

$$\omega = \frac{d\theta}{dt}. \tag{2.5a}$$

$$\omega_r = \frac{d\theta_r}{dt}. \tag{2.6a}$$

Nota-se que as equações de tensão de eixo direto e quadratura tem acoplamentos entre si, além dos termos rotatórios ω e ω_r, enquanto a representação trifásica não o trás. No entanto, essa representação ainda apresenta ganhos se consideramos que a correlação entre os fluxos concatenados e correntes no rotor e no estator são dependentes de operação senoidais e cossenoidais na modelagem trifásica do gerador de indução.

Os fluxos concatenados são relacionados às correntes e às indutâncias que os definem [16]. Esta relação é expressa nas equações:

$$\lambda_s^d = L_s i_s^d + L_m i_r^d, \tag{2.7a}$$

$$\lambda_s^q = L_s i_s^q + L_m i_r^q, \tag{2.7b}$$

$$\lambda_r^d = L_r i_r^d + L_m i_s^d, \tag{2.8a}$$

$$\lambda_r^q = L_s i_r^q + L_m i_s^q, \tag{2.8b}$$

onde L_m é a indutância mútua entre rotor e estator, L_s é a indutância de dispersão do estator e L_r é a indutância de dispersão do rotor.

Essas equações podem ainda ser representadas de forma matricial, conforme (2.9), facilitando a formulação do modelo de Espaço de Estados na Subseção 2.3.3.

$$\begin{bmatrix} \lambda_s^d \\ \lambda_s^q \\ \lambda_r^d \\ \lambda_r^q \end{bmatrix} = \begin{bmatrix} L_s + L_m & 0 & L_m & 0 \\ 0 & L_s + L_m & 0 & L_m \\ L_m & 0 & L_m + L_r & 0 \\ 0 & L_m & 0 & L_m + L_r \end{bmatrix} \begin{bmatrix} i_s^d \\ i_s^q \\ i_r^d \\ i_r^q \end{bmatrix} \tag{2.9}$$

A grande vantagem dos eixos direto e de quadratura está na simplicidade das equações, uma vez que as indutâncias L_s, L_r e L_m são invariantes no tempo. Ademais, para um sistema trifásico equilibrado, as grandezas direta e de quadratura são contantes.

O Torque Elétrico produzido pela máquina é uma medida direta da interação entre os fluxos concatenados e as correntes nos eixos direto e de quadratura do rotor [16]. Esta relação é expressa pela equação:

$$T_e = \frac{3}{4} p(\lambda_r^d i_r^q - \lambda_r^q i_r^d), \tag{2.10}$$

onde p é o número de polos da máquina.

2.3.2 Modelo eletromecânico

Para a modelagem do aerogerador é fundamental a descrição das equações mecânicas e sua interface com as grandezas elétricas. Para tanto, deve-se percorrer a modelagem da turbina eólica e as relações de torque da máquina. A potência cinética da turbina (P_t) pode ser descrita [17] por (2.11).

$$P_t = \frac{1}{2} C_p(\lambda, \beta) \rho \pi r^2 v_w^3, \tag{2.11}$$

onde ρ é a densidade do ar, r é o raio do rotor da turbina e v_w é a velocidade do vento. O coeficiente $C_p(\lambda, \beta)$ representa a eficiência aerodinâmica da turbina como uma função da razão do *tip speed ratio* (λ) e do ângulo das pás (β). O máximo teórico deste coeficiente é de 0.59 [18].

As turbinas eólicas podem ser representadas como um sistema de duas massas conectadas por meio de um eixo elástico [16]. Dessa forma, o torque mecânico da turbina (T_m) referenciado ao lado de alta rotação é dado portanto por:

$$T_m = \frac{P_t}{\omega_r}, \tag{2.12}$$

onde ω_r é a velocidade angular do rotor da máquina.

A interface entre os domínios eletromagnéticos e mecânicos ocorrem através do torque mecânico da turbina e do torque elétrico (T_e) gerado pela máquina de indução da forma descrita por (2.13).

Um desbalanço entre os torques elétrico e mecânico gera uma variação na velocidade da máquina de indução, descrita por (2.13)

$$J\frac{d\omega_r}{dt} = T_m - T_e - D\omega_r, \tag{2.13}$$

onde J é a inércia de todas as massas girantes e D é o coeficiente de atrito.

A velocidade mecânica se relaciona com a frequência elétrica (ω_e) sobre a influência do número de polos (p) da máquina através da relação.

$$\omega_e = \frac{p}{2}\omega_r. \tag{2.14}$$

Como visto na subseção 2.3.1, o modelo de máquinas utilizado leva em conta uma máquina equivalente de dois polos apenas. A relação é simplificada portanto a $\omega_e = \omega_r$.

Após realizados os cálculos do torque mecânico, (2.12), e do torque elétrico através do modelo dinâmico da máquina a ser apresentado na subseção 2.3.3, a velocidade do rotor pode ser calculada através de integração numérica. A equação (2.15) mostra o método de integração trapezoidal, utilizada neste trabalho. Para mais detalhes sobre este método de integração, se referir ao apêndice B.

$$x_k = x_{k-1} + \frac{h}{2}(\dot{x}_{k-1} + \dot{x}_k) \tag{2.15}$$

onde h é o passo de integração.

Se o coeficiente de amortecimento D é desprezado e se aplicada (2.15) em (2.12), têm-se (2.16), que provê a velocidade do rotor em um dado passo k da simulação.

$$\omega_{r_k} = \omega_{r_{k-1}} + \frac{hp}{4J}(T_{e_k} + T_{e_{k-1}} - T_{m_k} - T_{m_{k-1}}) \tag{2.16}$$

2.3.3 Representação em Espaço de Estados

Apresentadas as equações que modelam a máquina de indução duplamente alimentada, deve-se entender como esse conjunto de relações é resolvidas numericamente. Neste trabalho foi utilizado o modelo em Espaços de Estados. Nele, o sistema deve ser representado na forma apresentada em:

$$\dot{x} = \mathbf{A}x + \mathbf{B}u, \tag{2.17}$$

onde x é o vetor de estado, u é o vetor de entradas, $\mathbf{A}$ é a matriz de estado e $\mathbf{B}$ é a matriz de entrada.

São escolhidas como variáveis de estado os fluxos concatenados, (2.18), enquanto as entradas serão as tensões terminais nos enrolamentos fictícios, (2.19). Portanto, considerando a formulação da subseção 2.3.1, encontra-se (2.20) e (2.21), que modelam o sistema desejado (ver apêndice A). Nota-se que, embora a matriz de entrada seja constante, a matriz de estado é variável no tempo, sendo função da velocidade de rotação da máquina ω_r. Deve-se portanto, atualiza-lá a cada iteração da solução após resolvidas as equações mecânicas.

$$x = \begin{bmatrix} \lambda_s^d \\ \lambda_s^q \\ \lambda_r^d \\ \lambda_r^q \end{bmatrix} \tag{2.18}$$

$$u = \begin{bmatrix} v_s^d \\ v_s^q \\ v_r^d \\ v_r^q \end{bmatrix} \tag{2.19}$$

$$\mathbf{A} = \begin{bmatrix} -R_s\mathbf{L^{-1}}_{11} & -\omega & -R_s\mathbf{L^{-1}}_{13} & 0 \\ \omega & -R_s\mathbf{L^{-1}}_{22} & 0 & -R_s\mathbf{L^{-1}}_{24} \\ -R_r\mathbf{L^{-1}}_{31} & 0 & -R_r\mathbf{L^{-1}}_{33} & \omega_r - \omega \\ 0 & -R_r\mathbf{L^{-1}}_{42} & \omega - \omega_r & -R_r\mathbf{L^{-1}}_{44} \end{bmatrix} \tag{2.20}$$

onde $\mathbf{L^{-1}}$ é a inversa da matriz de indutâncias (2.9), que relaciona as correntes e fluxos concatenados nos enrolamentos equivalentes direto e de quadratura.

$$\mathbf{B} = -\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = -\mathbf{I} \tag{2.21}$$

Por fim, para resolver (2.17), deve-se escolher um método de integração numérica. Se utilizado o método de integração trapezoidal apresentado em (2.15), o vetor de estado em uma iteração k será função da derivada da função de estado.

Substituindo (2.15) em (2.17), é possível obter (2.22), que descreve a relação do vetor de estados para uma dada iteração com a matriz de estado, a matriz de entrada, o vetor de entrada, o passo de integração e variáveis históricas (ver apêndice B).

$$x_k = \left[\mathbf{I} - \frac{h}{2}\mathbf{A}\right]^{-1} \times \left[\left(\frac{h}{2}\mathbf{A} + \mathbf{I}\right)x_{k-1} + \frac{h}{2}\mathbf{B}\left(u_k + u_{k-1}\right)\right] \tag{2.22}$$

O processo de solução, portanto, consiste nos seguintes passos, ilustrados na Figura 2.9.

1. Leitura das tensões trifásicas nos terminais do estator (v_s^a,v_s^b,v_s^c) e do rotor (v_r^a,v_r^b,v_r^c) da máquina;

2. Aplicar a Transformada de Park para o estator ($[\mathbf{T_s}]$) e para rotor ($[\mathbf{T_r}]$) nas tensões terminais;

3. Com as tensões nos eixos direto e de quadratura (v_s^d, v_s^q, v_r^d, v_r^q) e a velocidade do rotor (ω_r) do passo anterior, atualizar o modelo em Espaço de Estados;

4. Resolver o modelo em Espaços de Estados utilizando o método de integração trapezoidal de forma a obter o vetor com os fluxos concatenados (λ_s^d, λ_s^q, λ_r^d, λ_r^q);

5. Obter o vetor de correntes de Park (i_s^d, i_s^q, i_r^d, i_r^q) através de (2.9);

6. Utilizar a Tranformada Inversa de Park para retornar as correntes do estator para os referenciais trifásicos no estator ($[\mathbf{T_s}]^{-1}$) e no rotor ($[\mathbf{T_r}]^{-1}$).

7. solução do modelo mecânico da turbina e da máquina apresentado na subseção 2.3.2 para obtenção da velocidade rotórica.

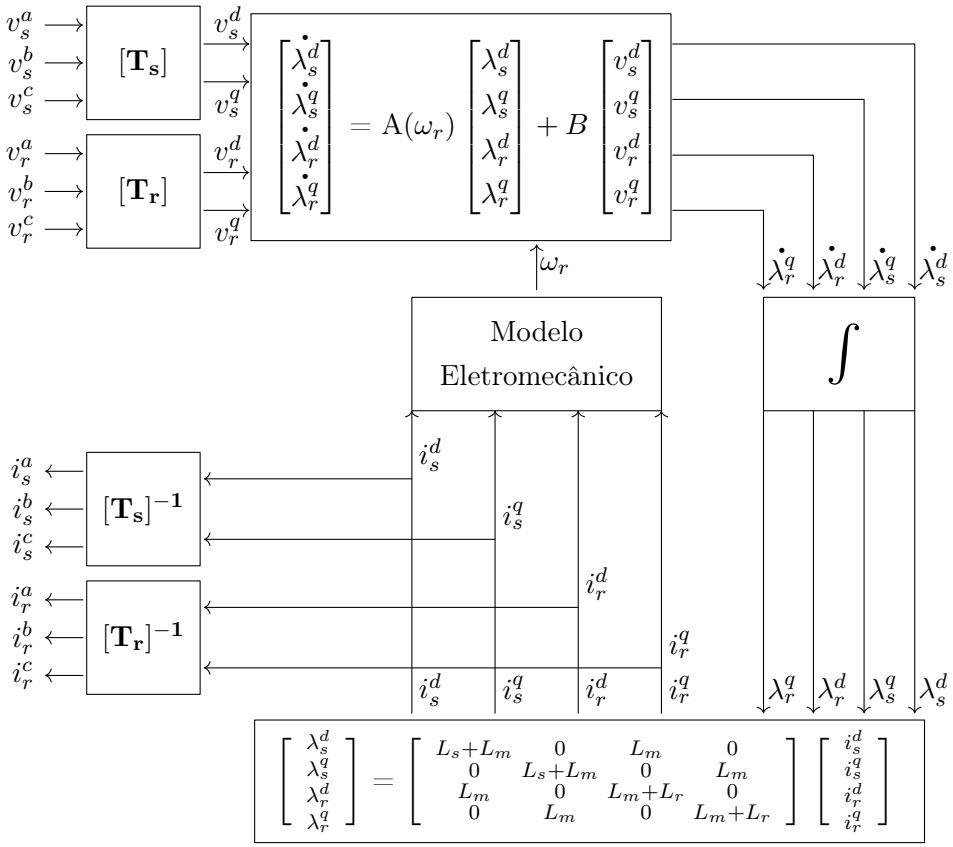

Figura 2.9: Diagrama de solução do modelo da máquina de indução

2.3.4 Controle do Conversor *Back-to-Back*

O conversor *Back-to-Back* da DFIG é tipicamente composto por duas partes distintas: o GSC (*Grid-Side Converter*), responsável por manter uma tensão constante no barramento CC do conversor, e o RSC (*Rotor-Side Converter*), que pode ser configurado para controlar potência, torque ou velocidade de rotação do aerogerador. O controle da DFIG é comumente realizado utilizando coordenadas *dq* (direta e quadratura), que permitem decompor as variáveis trifásicas em componentes constantes. Esse método facilita a aplicação de técnicas de controle baseadas em conceitos clássicos.

Na configuração *Back-to-Back*, o GSC desempenha um papel crucial na estabilização do barramento CC, assegurando uma tensão constante para o funcionamento adequado do sistema de controle. Por outro lado, o RSC é responsável por ajustar dinamicamente a operação do gerador de indução duplamente alimentado, permitindo uma resposta rápida às variações nas condições de vento e demanda de rede.

Essa abordagem coordenada em *dq* não só simplifica o projeto e a implementação dos algoritmos de controle, mas também melhora a eficiência e a estabilidade do aerogerador DFIG durante a geração de energia elétrica.

Controle do GSC

Enquanto o lado CA do conversor está diretamente conectado à rede elétrica, o GSC regula a tensão no barramento CC compartilhado com o RSC. Devido à variação possível na frequência da rede, é crucial sincronizar adequadamente o sistema. Para isso, utiliza-se o PLL (*Phase-Locked Loop*) para sincronizar um sinal de tensão, calculand seu ângulo de fase.

O PLL consiste em três componentes principais: o detector de fase, que gera um sinal de erro proporcional à diferença de fase entre o sinal de entrada e saída; o filtro passa-baixa, que extrai o sinal de controle do sinal de erro; e o oscilador, responsável por gerar um sinal de saída proporcional à saída do filtro. A Figura 2.10 mostra o diagrama do PLL para um sinal trifásico.

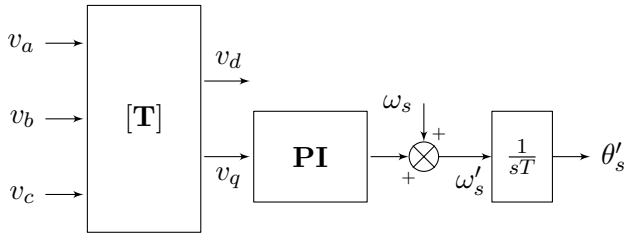

Figura 2.10: Diagrama do PLL.

O detector de fase realiza a Transformada de Park para converter as tensões trifásicas da rede (v_a, v_b, v_c) em componentes de eixo direto (v_d) e de quadratura (v_q). Segundo [18], as tensões da rede nesse sistema de coordenadas são dadas por (2.23).

$$\begin{aligned} v_d &= v_g \, cos(\theta_g - \theta'_g), \\ v_q &= v_g \, sen(\theta_g - \theta'_g), \end{aligned} \tag{2.23}$$

onde v_g é a amplitude da tensão da rede.

Para sinais de erro pequenos $\theta_g - \theta'_g \approx 0$ e portanto $sen(\theta_g - \theta'_g) \approx \theta_g - \theta'_g$. Substituindo esse termo em (2.23), obtém-se:

$$v_q = v_g \, (\theta_g - \theta'_g), \tag{2.24}$$

Como o erro entre a fase da rede e a fase de saída é proporcional à componente de quadratura, a Transformada de Park é eficaz para estimar o ângulo de fase da rede. O filtro passa-baixa, implementado como um controlador PI, ajusta essa componente para que o PLL siga corretamente a fase da rede. O oscilador é composto por um integrador que, ao integrar o sinal de frequência, fornece o sinal de fase da rede.

O ângulo de fase da rede (θ_g) é crucial para o controle do GSC, sendo utilizado

como entrada para o bloco final da Transformada de Park que gera os sinais de chaveamento do conversor do lado da rede. Além disso, o controle do GSC requer também a tensão do barramento CC do conversor (d_{dc}). O controle implementado é baseado em um controlador PI, onde a malha externa controla a tensão do barramento CC e a malha interna controla a corrente do GSC.

Controle do RSC

Neste trabalho, o RSC é abordado para controlar a potência reativa injetada ou absorvida na rede, assim como a velocidade de rotação. Entre várias técnicas de controle, como controle em frequência e controle por ângulo de fase, uma das mais eficazes é o controle vetorial, onde o vetor de fluxo no estator da máquina de indução desempenha um papel central. A equação que define o vetor de fluxo no estator é expressa por (2.25) [19].

$$\phi_s = \int_0^t v_s - R_s i_s \, dt. \tag{2.25}$$

Utilizando a Tranformada de Clark, é possível decompô-la em (2.26), em termos das coordenadas α e β.

$$\begin{aligned} \phi_\alpha &= \int_0^t v_\alpha - R_s i_\alpha \, dt. \\ \phi_\beta &= \int_0^t v_\beta - R_s i_\beta \, dt. \end{aligned} \tag{2.26}$$

Portanto, é possível obter a posição do fluxo através da transformação do vetor de fluxo em componentes α e β seguido da integração das grandezas de tensão e corrente. É comum ainda a utilização de um filtro passa alta para eliminar os problemas de *drift* observado em frequência muito baixa [19]. Para obter a posição do fluxo basta utilizar (2.27). O diagrama completo do estimador de fluxo é observado na Figura 2.11.

Assim como no controle do GSC, o controle do RSC é baseado em reguladores PI, onde a malha externa controla a potência reativa e a velocidade da máquina, enquanto a malha interna regula as correntes do conversor no lado do rotor. Portanto, para o controle do RSC, além das entradas necessárias ao estimador de fluxo, são requeridos a potência reativa instantânea absorvida ou injetada na rede (q) e a velocidade de rotação da máquina (ω_r).

$$\theta_s = tan^{-1}\left(\frac{\phi_\beta}{\phi_\alpha}\right) \tag{2.27}$$

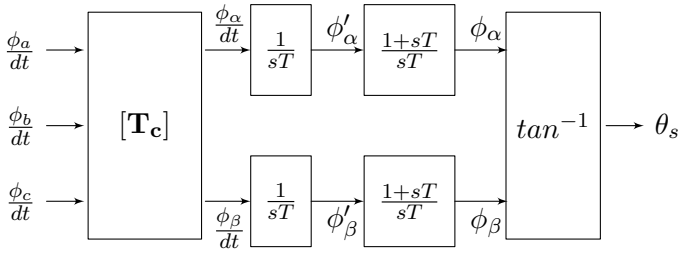

Figura 2.11: Diagrama do Estimador de Fluxo.

2.4 Análise Nodal Modificada Aumentada

Como visto no inínio desta seção, a máquina de indução de rotor bobinado estará conectada à rede elétrica e à um conversor *Back-to-Back*. Sendo assim, é necessário modelar estes outros elementos do sistema para que, com as correntes calculadas com o modelo de máquinas, sejam calculadas as tensões nodais, que serão novamente utilizadas como entradas do modelo de máquinas.

Para a representação do conversor e solução de rede foi utilizado a Análise Nodal Modificada Aumentada (MANA - *Modified Augmented Nodal Analysis*), método de solução baseado nas Leis de Kirchhoff. Para circuitos com apenas resistências lineares e fontes de corrente, a formulação, mostrada em (2.28), é igual àquela obtida pela Lei dos Nós [20].

$$\mathbf{Y}_n\mathbf{V}_n = \mathbf{I}_n. \tag{2.28}$$

onde $\mathbf{Y}_n$ representa a matriz de admitâncias, $\mathbf{V}_n$ representa as tensões nodais e $\mathbf{I}_n$, o vetor de fontes de corrente.

Para representar as fontes de tensão, são adicionados à formulação novos elementos tal que obtêm-se (2.29). A matriz $\mathbf{Y}_R$ se difere de $\mathbf{Y}_n$ pois leva em conta apenas a parte de $\mathbf{Y}$ relativa às tensões nodais desconhecidas.

$$\begin{bmatrix} \mathbf{Y}_R & \mathbf{V}_c \\ \mathbf{V}_r & \mathbf{V}_d \end{bmatrix} \begin{bmatrix} \mathbf{V}_n \\ \mathbf{I}_f \end{bmatrix} = \begin{bmatrix} \mathbf{I}_n \\ \mathbf{V}_f \end{bmatrix} \tag{2.29}$$

A matriz $\mathbf{V}_r$ contém as informações referentes às fontes de tensão do problema. Seja uma fonte de tensão com valor v_f entre os nós k e m, têm-se que $v_f = v_k - v_m$, e adiciona-se respectivamente 1 e -1 nas colunas k e m de $\mathbf{V}_r$. Além disso, a matriz $\mathbf{V}_c$ é obtida com a transposição de $\mathbf{V}_r$. O vetor $\mathbf{I}_f$ adicionado representa as correntes que passam pelas fontes de corrente e o vetor $\mathbf{V}_f$ terá a informação da tensão nas fontes de tensão. A matriz $\mathbf{V}_d$ é composta por zeros a menos que sejam representados fontes de tensão desligadas.

Para completar a formulação de todo os elementos que compõem a topologia

DFIG, é necessário adicionar à formulação os modelos de chaves controladas para representar o conversor como mostrado em (2.30).

$$\underbrace{\begin{bmatrix} \mathbf{Y}_R & \mathbf{V}_c & \mathbf{S}_c \\ \mathbf{V}_r & \mathbf{V}_d & 0 \\ \mathbf{S}_r & 0 & \mathbf{S}_d \end{bmatrix}}_{\mathbf{G}} \underbrace{\begin{bmatrix} \mathbf{V}_n \\ \mathbf{I}_f \\ \mathbf{I}_d \end{bmatrix}}_{\mathbf{V}} = \underbrace{\begin{bmatrix} \mathbf{I}_n \\ \mathbf{V}_f \\ \mathbf{S}_q \end{bmatrix}}_{\mathbf{I}_g} \tag{2.30}$$

(2.31) mostra o comportamento da chave ideal, que será utilizada para representar o conversor *Back-to-Back*. Isso é alcançado inserindo respectivamente 1 e -1 nas colunas k e m de $\mathbf{S}_r$ quando a chave está fechada, enquanto a matriz $\mathbf{S}_r$ será a sua transposta. Quando a chave está aberta, a nulidade da corrente é representada clocando o elemento diagonal da matriz $\mathbf{S}_d$ igual a 1.

$$\begin{cases} v_k - v_m = 0 & \text{quando a chave está fechada;} \\ i_s = 0 & \text{quando a chave está aberta.} \end{cases} \tag{2.31}$$

Nota-se, portanto, que a matriz $\mathbf{G}$ deverá ser atualizada a cada passo de iteração a depender do estado de cada chave presente no circuito elétrico a ser resolvido. Da mesma forma, o vetor $\mathbf{I}_g$ deve ser atualizado com o valores de fonte de corrente e fonte de tensão na dada iteração. A matriz $\mathbf{S}_q$ será sempre nula. Atualizadas as matrizes que representam o circuito elétrico, a solução da MANA é dada por (2.32). Da formulação apresentada, a matriz $\mathbf{G}$ é de dimensão $(N+N_v+N_s)$ x $(N+N_v+N_s)$, onde N é o número de nós do circuito, N_v é o número de fontes de tensão e N_s é o número de chaves. Além disso, para cada chave presente no circuito, dobra-se o número de matrizes $\mathbf{G}$ que podem representar o circuito, ou seja, deve-se armazenar um total de 2^{N_s} matrizes.

$$\mathbf{V} = \mathbf{G}^{-1}\mathbf{I}_g. \tag{2.32}$$

2.4.1 Discretização de indutores e capacitores

Devido a natureza diferencial das equações que descrevem o comportamento de indutores e capacitores, estes são de difícil acoplamento na matriz de admitâncias da formulação da MANA. Uma maneira de contornar o problema é utilizar a Substituição com Integração Numérica (NIS - *Numerical Intergation Substituition*) [21], que consiste em representar os ramos de indutores e capacitores atrvés de um Equivalente de Norton, que irá manter a corrente e tensão do ramo.

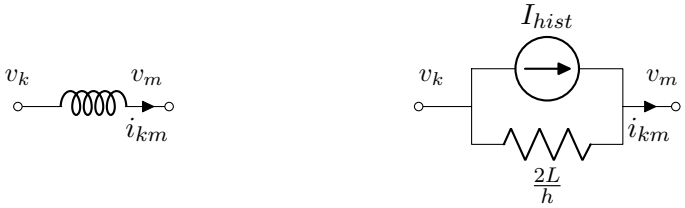

Figura 2.12: Discretização do indutor com NIS.

Indutância

(2.33) descreve a relação entre tensão e corrente de um indutor com indutância L entre os nós k e m.

$$v_k - v_m = L\frac{di_{km}}{dt}. \tag{2.33}$$

Aplicando a integração do tempo inicial ($0s$) até um tempo t e separando a integral em um tempo $t-h$, onde h é o passo de integração:

$$\begin{aligned} i_{km_t} &= \frac{1}{L}\int_0^t (v_k - v_m)dt \\ &= \frac{1}{L}\int_0^{t-h} (v_k - v_m)dt + \frac{1}{L}\int_{t-h}^t (v_k - v_m)dt, \end{aligned} \tag{2.34}$$

O termo $\frac{1}{L}\int_0^{t-h}(v_k - v_m)dt$ de (2.34) é exatamente a corrente no indutor no tempo $t-h$. Assim, pode-se reescrever a equação como:

$$i_{km_t} = i_{km_{t-h}} + \frac{1}{L}\int_{t-h}^t (v_k - v_m)dt, \tag{2.35}$$

Aplicando o método de interação trapezoidal:

$$\begin{aligned} i_{km_t} &= i_{km_{t-h}} + \frac{h}{2L}(v_{km_t} + v_{km_{t-h}}), \\ i_{km_t} &= \underbrace{i_{km_{t-h}} + \frac{h}{2L}v_{km_{t-h}}}_{I_{hist}} + \frac{h}{2L}v_{km_t}, \end{aligned} \tag{2.36}$$

Os termos históricos de (2.36), são representado como uma fonte de corrente (I_{hist}) de valores já calculados na iteração anterior. Já o termo $\frac{h}{2L}v_{km_t}$ é representado como ma resistência de valor $R_{eq} = \frac{2L}{h}$ com uma tensão aplicada de v_{km}. Assim, (2.37) mostra a formulação final da discretização de um indutor, também representada na Figura 2.12.

$$i_{km_t} = I_{hist} + \frac{1}{R_{eq}}v_{km_t}. \tag{2.37}$$

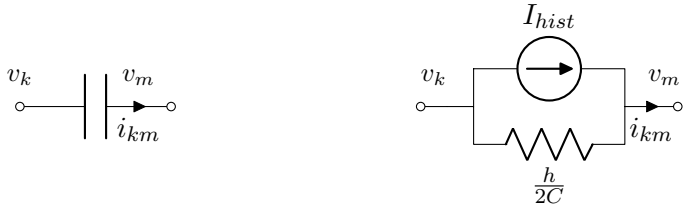

Figura 2.13: Discretização do capacitor com NIS.

Capacitância

(2.38) descreve a relação entre tensão e corrente de um capacitor com capacitância C entre os nós k e m.

$$i_{km} = C\frac{dv_{km}}{dt}. \tag{2.38}$$

Aplicando a integração do tempo inicial ($0s$) até um tempo t e separando a integral em um tempo $t-h$, onde h é o passo de integração:

$$\begin{aligned} v_{km_t} &= \frac{1}{C}\int_0^t i_{km}dt \\ &= \frac{1}{C}\int_0^{t-h} i_{km}dt + \frac{1}{L}\int_{t-h}^{t} i_{km}dt, \end{aligned} \tag{2.39}$$

O termo $\frac{1}{C}\int_0^{t-h} i_{km}dt$ de (2.39) é exatamente a tensão no capacitor no tempo $t-h$. Assim, pode-se reescrever a equação como:

$$v_{km_t} = v_{km_{t-h}} + \frac{1}{C}\int_{t-h}^{t} i_{km}dt, \tag{2.40}$$

Aplicando o método de interação trapezoidal:

$$\begin{aligned} v_{km_t} &= v_{km_{t-h}} + \frac{h}{2C}(i_{km_t} + i_{km_{t-h}}), \\ v_{km_t} &= \underbrace{v_{km_{t-h}} + \frac{h}{2C}i_{km_{t-h}}}_{-I_{hist}} + \frac{h}{2C}i_{km_t}, \end{aligned} \tag{2.41}$$

Os termos históricos de (2.41), são representado como uma fonte de corrente (I_{hist}) de valores já calculados na iteração anterior. Já o termo $\frac{2C}{h}v_{km_t}$ é representado como ma resistência de valor $R_{eq} = \frac{h}{2C}$ com uma tensão aplicada de v_{km}. Assim, (2.42) mostra a formulação final da discretização de um indutor, também representada na Figura 2.13.

$$i_{km_t} = I_{hist} + \frac{1}{R_{eq}}v_{km_t}. \tag{2.42}$$

Capítulo 3

Desenvolvimento e Implementação

3.1 Acoplamento do modelo de rede com o modelo de máquinas

O modelo de máquina de indução apresentado na subseção 2.3.3 utiliza as tensões nodais como entrada e retorna as correntes trifásicas no estator e no rotor da máquina de indução. Dessa forma, ela é modelada como injeções de corrente no modelo de rede.

Para reduzir instabilidades numéricas que podem ser causadas pelo acoplamento da solução de rede (MANA) com o modelo da máquina (Espaço de Estados), é adicionado uma resistência de $R' = 2L'/h$ em paralelo com os dois modelos [4], onde L' é a indutância subtransitória da máquina. Para compensar a ação da dessa resistência, adiciona-se também em paralelo uma fonte de corrente de compensação $i_c = \frac{v_{t-h}}{R'}$.

A Figura 3.1 mostra o circuito que representa os elementos da DFIG. Nele, há 12 fontes de corrente que representam as injeções de corrente do rotor e do estator e o acoplamento entre modelo de máquinas e modelo de rede, 6 resistores de acoplamento, 4 equivalentes de Norton que representam o capacitor e os indutores do conversor, 12 chaves ideais e uma fonte de tensão trifásica, que representa a rede elétrica.

Da formulação apresentada na seção 2.4, conclui-se que a matriz $\mathbf{G}$ deve ter ordem de 26x26, equanto os vetores $\mathbf{V}$ e $\mathbf{I}_g$ terão 26 elementos.

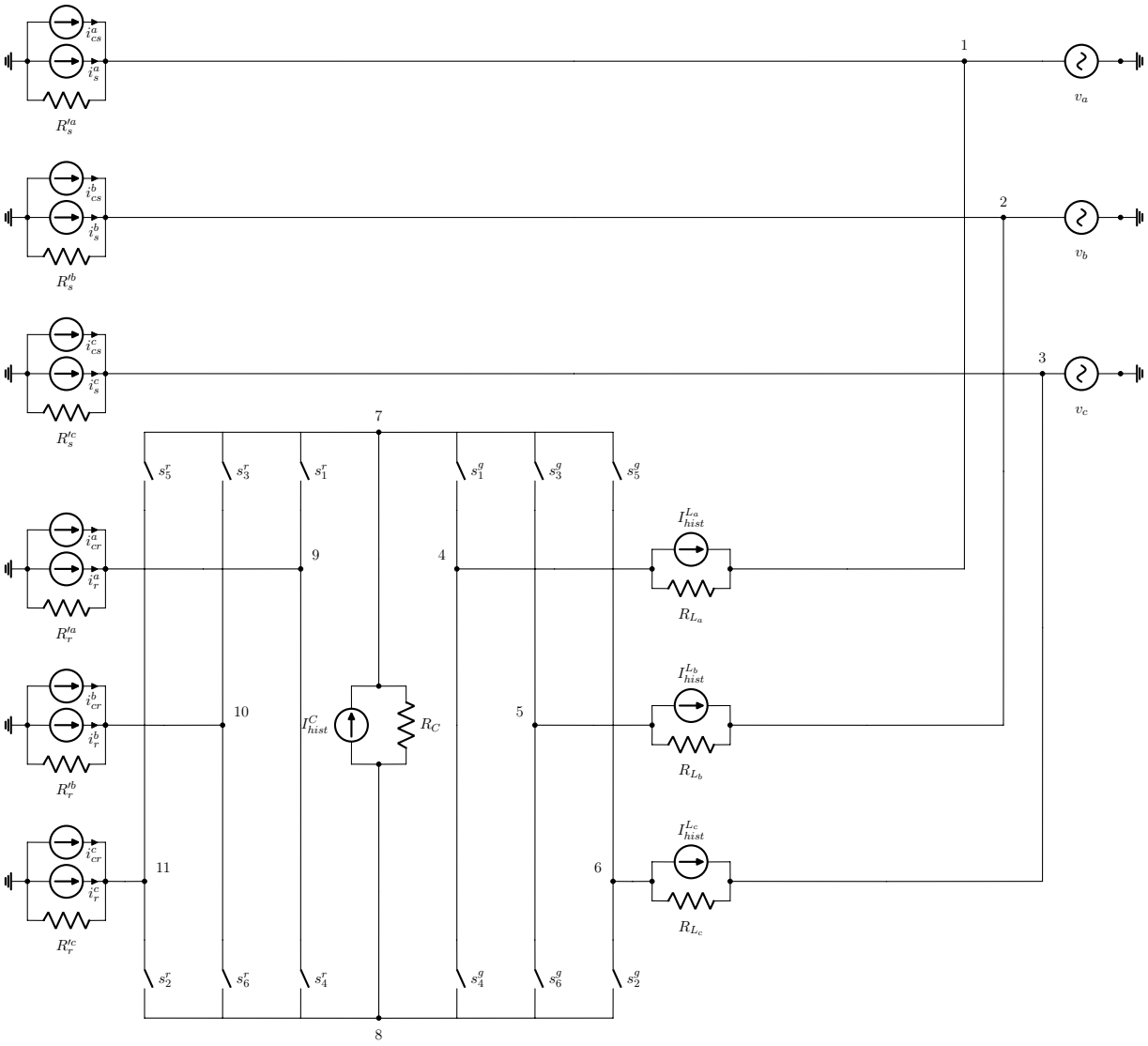

Figura 3.1: Circuito que representa o modelo da DFIG.

Após solucionadas as equações das máquinas, deve-se seguir os seguintes passos para encontrar a solução de rede:

1. Atualizar as fontes históricas do modelo de discretização de capacitores e indutores;

2. Atualizar o vetor $\mathbf{I}_g$ com os valores das fontes de corrente e fontes de tensão da iteração atual;

3. Identificar o estado das chaves do circuito e identificar a matriz $\mathbf{G}^{-1}$ que representa a topologia atual do sistema;

4. Resolver (2.32) e encontrar o vetor $\mathbf{V}$.

3.2 Arquitetura do simulador

A implementação de um simulador em tempo real de um aerogerador a velocidade variável na topologia DFIG é o principal objetivo deste trabalho. Para tanto, é necessário implementar toda a formulação das seções 2.3, 2.4 e 3.1 repeitando os requisitos que garantam a sincronia com o tempo físico, isto é, a duração de execução de um passo de simulação não deve ser maior que h, como detalhado na seção 2.1. A simulação, utilizando como plataforma de simulação a FPGA, é totalmente digital. Dessa maneira, é necessário uma DAC (*Digital-to-Analog Converter*) de forma que a exportação final dos sinais seja realizada de forma analógica. A Figura 3.2 mostra a estrutura da solução implementada na FPGA.

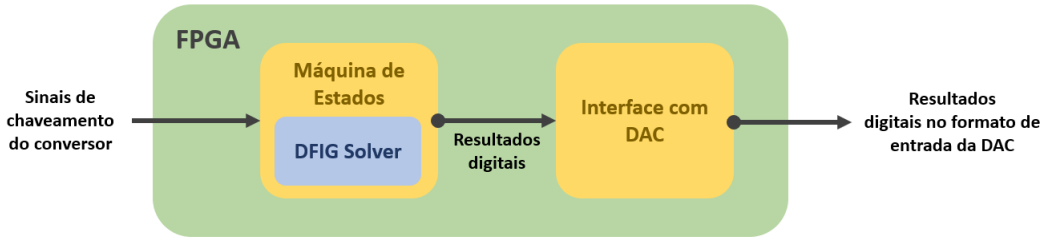

Figura 3.2: Diagrama da arquitetura de solução e exportação para a DAC.

A arquitetura possui 3 principais componentes: a maquina de estados, que irá impor os estados da simulação em tempo real; o *Solver*, onde o modelo da DFIG será implementado e a Interface com a DAC, responsável pela padronização dos sinais internos para que possam ser lidas pela DAC. A seguir esses módulos são detalhados.

3.2.1 Máquina de estados

A máquina de estados é responsável por coordenar os estados da simulação em tempo real apresentadas na seção 2.1. Isso significa que ele será capaz de identificar quando o *Solver* terminou sua execução e também o fim do passo de simulação para exportação dos resultados. Em caso de falha nos requisitos da simulações em tempo real, a máquina de estados deve entrar em um estado de erro. Esse componente, modelado em VHDL, é uma autômato que possui quatro estados: *start*, *solve*, *idle* e *error*, mostrados na Figura 3.3.

No estado de *start*, o simulador aguarda o comando de inicio da simulação (*cmd*), que nesta implementação é acionado com um *push buttom* na FPGA. Depois que este é acionado, o solver entra em um *loop* infinito.

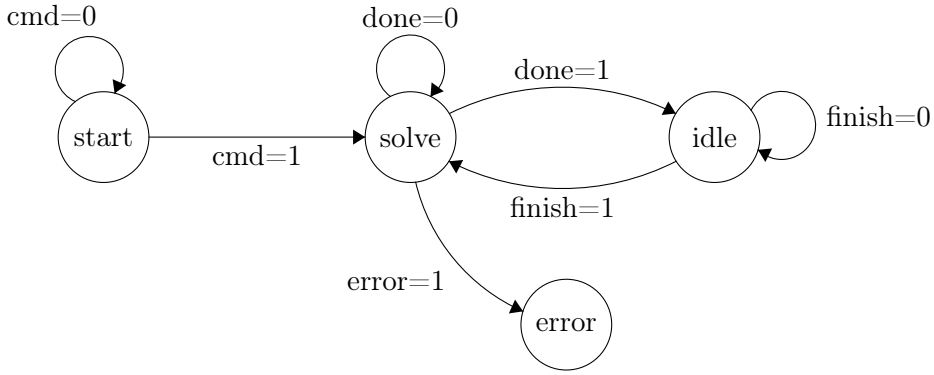

Figura 3.3: Máquina de estados implementada em VHDL.

Esse *loop*, para uma execução com sucesso será composto de dois estados: *solve* e *idle*. O primeiro é onde as equações implementadas na FPGA são solucionadas e todas as saídas do sistema são calculadas. Ou seja, nele, o núcleo de solução *DFIG Solver* está ativo e em plena execução. Quando as saídas já estão calculadas mas o passo de simulação ainda não terminou, o comando *done* é acionado e o automato passa para o estado de *idle*, onde ele aguarda o fim do passo de simulação. Neste estágio, o núcleo de solução entra em estado de espera para o inicio do próximo passo da simulação. Com o fim do estado de *idle*, os resultados são exportados para o componente de interface com a DAC e a máquina de estados retorna ao estado de *solve* com o envio do comando *finish*.

Entretanto, se em alguma iteração o tempo de execução, isto é, o tempo no estado *solve*, é maior que o passo de simulação, o simulador em tempo real falhou e deve-se, pois, entrar em um estado que indique a falha na execução. Este é o estado *error*, no qual permanece a simulação até que ela seja reiniciada ou encerrada. A identificação do fim do passo de simulação é implementado através de um contador, que conta o número de *clocks* (n_{clk}) recebidos e é reiniciado a cada N_{clk}, o número de *clocks* dentro de um passo de integração. (3.1) mostra a relação entre a frequência do *clock* e o número de clocks em um passo de simulação.

$$N_{clk} = f_{clk} h \tag{3.1}$$

3.2.2 *DFIG Solver*

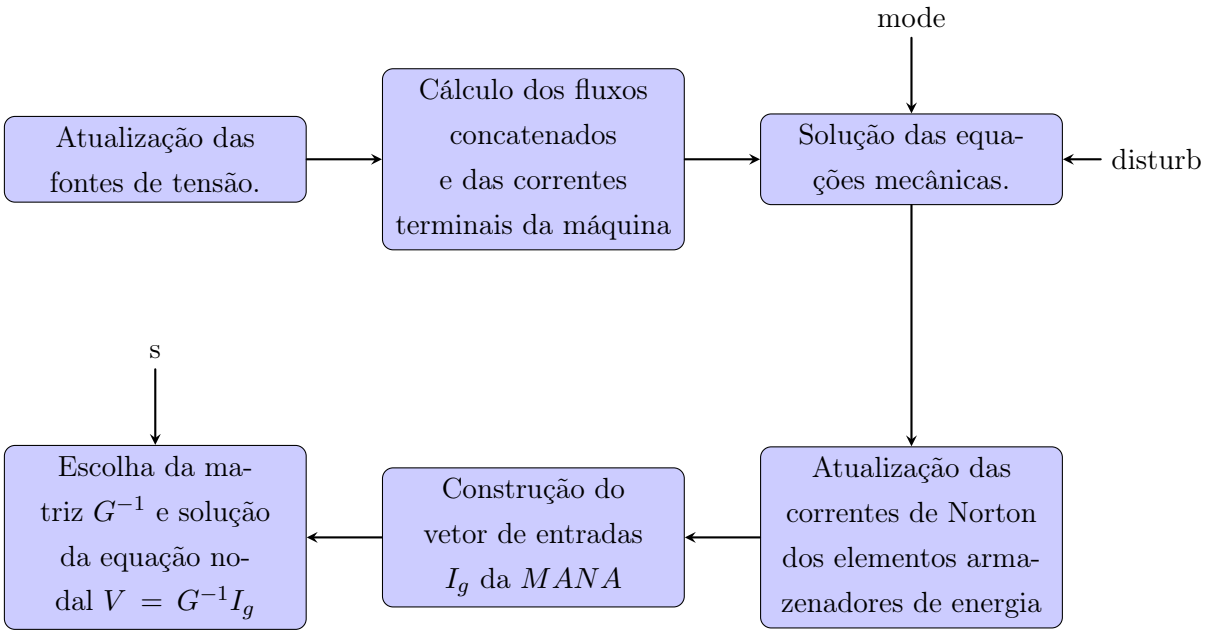

Figura 3.4: Diagrama com as principais funções do *solver* que modela a DFIG.

O principal componente controlado pela máquina de estados é o *DFIG Solver*, responsável por receber os sinais de chaveamento do conversor *Back-to-Back* e calcular todas as saídas necessárias para o seu controle. Para a modelagem dele foi utilizado o *Vitis HLS*, ferramenta HLS da Xilinx. Como apresentado na subseção 2.2.2, o uso dessa ferramenta permite escrever um código em C/C++ e convertê-lo em um *IP Core* no código em VHDL. Ela é composta por 7 funções que descrevem a formulação do Capítulo 2.

A primeira função é responsável por atualizar o valor das fontes de tensão presentes no circuito. A fonte de tensão é implementada como uma *struct* com os atributos da Tabela 3.1.

Tabela 3.1: Atributos da estrutura de fonte de tensão

V_SOURCE

Atributo	Tipo
dtheta	double
amplitude	double
phase	double
DClevel	double

Assim, a tensão v_i^k da fonte i em um passo k da simulação pode ser calculado como descrito em (3.2). Os parâmetros `DClevel` e `amplitude` são, respectivamente,

o *offset* e a amplitude do sinal senoidal da fonte de tensão. No caso deste trabalho são utilizadas 3 instâncias de V_SOURCE, que representam a rede elétrica trifásica.

$$v_i^k = DClevel_i + amplitude_i + sin(phase_i^k) \tag{3.2}$$

O atributo `dtheta` é calculado como $dtheta_i = 2\pi f_i h$, onde f_i é a frequência da fonte de tensão i. Ele é utilizado para atualizar o argumento *phase* em toda iteração. Assim, em uma etapa k o parâmetro *phase* será:

$$phase_i^k = phase_i^{k-1} + dtheta_i,\ phase_i^k \in [0, 2\pi] \tag{3.3}$$

A próxima estrutura criada tem como objetivo representar a máquina de indução de rotor bobinado. A Tabela 3.2 mostra os parâmetros e seus tipos respectivos. Neste contexto, a segunda função definida irá identificar os terminais da máquina, que estão definidos nos atributos `statornodes[3]` e `rotornodes[3]` para o estator e para o rotor, respectivamente, ler suas tensões trifásicas e armazená-las nos atributos `statorVoltages[3]` e `rotorVoltages[3]` para em seguida calcular suas tensões em componentes direta e de quadratura utilizando as transformadas mostradas em (2.1), que são atualizadas toda iteração com o ângulo de posição do rotor da máquina, representado pelo atributo `theta`.

Antes de calcular as tensões nos eixos fictícios *dq*, deve-se armazenar o valor calculado na iteração anterior. Assim, o vetor de tensões de Park históricas é armazenado no atributo `historicParkVoltages[4]` enquanto o calculado na iteração atual é alocado no atributo `parkVoltages[4]`. Observa-se que os vetores que contém as tensões nos referenciais de Park possuem dimensão 4, sendo 2 para o estator e 2 para o rotor, pois como mencionado na subseção 2.3.1, não há necessidade aqui de considerar o eixo de sequência zero.

Tabela 3.2: Atributos da estrutura de máquina de indução

INDUCTION_MACHINE

Atributo	Tipo
theta	double
statornodes[3]	int
rotornodes[3]	int
statorVoltages[3]	double
rotorVoltages[3]	double
parkVoltages[4]	double
historicParkVoltages[4]	double
fluxVector[4]	double
historicFluxVector[4]	double
parkCurrents[4]	double
statorCurrents[3]	double
rotorCurrents[3]	double
speed	double
historicTe	double
Te	double
historicTm	double
Tm	double

Outra é responsável por calcular as correntes nos terminais da máquina de indução, possibilitando sua representação como fonte de corrente na solução da MANA. Para tanto, a primeira etapa é a montagem do vetor de fluxos concatenados, que são as variáveis de estado do sistema máquina de indução. Esse vetor é obtido através da integração numérica (2.22) e armazenado no atributo `fluxVector[4]`. Mais uma vez, é necessário armazenar o valor histórico do vetor de fluxos, pois é necessário para a integração trapezoidal. Isso é feito através do atributo `historicFluxVector[4]`. Em seguida, é possível calcular as correntes nos terminais da máquina em função utilizando (2.9). Essas correntes são armazenadas no atributo `parkCurrents[4]`, que será utilizando em seguida para calcular as correntes trifásicas `statorCurrents[3]` e `rotorCurrents[3]`, necessárias para a solução da MANA, utilizando a transformada inversa de Park.

Com o modelo elétrico resolvido, a próxima função lida com as equações eletromecânicas da máquina de indução. Isto é, calcula o torque elétrico `Te` e a velocidade de rotação da máquina `speed`, com (2.10) e (2.16). No entanto, antes desses cálculos é necessário saber o torque mecânico `Tm` realizado pela turbina, que será possível através de 2.11 e 2.12. Essa função terá duas entradas, `mode` e `disturb`, do tipo `int`. A primeira irá representar o modo de operação da máquina de indução. Isto é, modo

de velocidade constante ou modo de torque contante. No primeiro, a máquina de indução irá trabalhar com o rotor girando a uma velocidade constante; no segundo, o rotor é liberado e pode sofrer as variações do torque eletromecânico, que irá mudar sua velocidade. A segunda entrada, quando igual a 1, irá provocar uma mudança instantânea na velocidade de vento na turbina do aerogerador.

Com o fim desses cálculos, a solução do modelo de máquina de indução está completo e segue-se para a solução de rede. Para tanto, o primeiro passo é atualizar os equivalentes de Norton que representam os elementos armazenadores de energia. Isso é realizado na função seguinte atuando sobre a estrutura LC_ELEMENT, que representa indutores e capacitores. Seus atributos são descritos na Tabela 3.3. O atributo `type` é indica se o elemento em questão é um indutor ou um capacitor e os atributos `n_in` e `n_out` localizam os nós onde aquele elemento está inserido no circuito. Com o conhecimento da resistência de Norton `Req` calculada previamente é possível calcular o valor da corrente de Norton `Ih` que será adicionada no vetor $\mathbf{I}$ da solução nodal.

Tabela 3.3: Atributos da estrutura de elementos armazenadores de energia

LC_ELEMENT

Atributo	Tipo
type	unsigned char
n_out	unsigned char
n_in	unsigned char
Req	double
Ih	double

Chegando à etapa de solução de rede, é necessário uma função que insere no vetor $\mathbf{I}_g$ da análise nodal as entradas atualizadas. Isso deve ser feito para a fontes de tensão e fontes de corrente, como descrito na seção 2.4. Ou seja, as fontes de tensão, elementos armazenadores de energia e a máquina de indução são aqui representados. Deve-se lembrar, entretanto, da seção 3.1 que descreve a necessidade de uma fonte de corrente adicional em cada terminal da máquina de forma melhor a estabilidade do modelo. Esse acoplamento também é nesta função processado.

A última função do modelo é responsável por calcular as tensões nodais do circuito representado. Antes de performar (2.32) através de uma multiplicação matriz-vetor e encontrar a tensão em cada nó do circuito, deve escolher a matriz $\mathbf{G}^{-1}$ que representa o estado atual do circuito. Isso é necessário, pois a função terá uma entrada `s` do tipo `int` que representará o estado de cada chave do conversor *Back-to-Back*. Todas as 64 matrizes possíveis $\mathbf{G}^{-1}$ possíveis já estão armazenadas em memória não sendo necessário, portanto, calculá-las. A Figura 3.4 mostra em resumo as funções principais implementadas no componente *DFIG Solver*.

3.2.3 Interface com a DAC

Um componente crítico na implementação de um simulador em tempo real é a interface de exportação. Assim como a máquina de estados, essa interface é completamente modelada em VHDL. Sua principal função é garantir a comunicação adequada entre a FPGA e a DAC, sem influência direta no processo de solução do circuito. A comunicação é realizada através da padronização de sinais, variando para cada modelo de DAC, resultando na necessidade de ajustar o modelo de interface de exportação para cada situação específica.

A especificação mais relevante é o número de bits dos canais de entrada do conversor DAC, já que os resultados gerados pelo núcleo de solução são valores em ponto flutuante. Esses valores precisam ser convertidos para um formato de ponto fixo. A DAC utilizada neste trabalho foi a DAC34SH84 da Texas Instruments, que possui canais de entrada de 16 bits [22], portanto, cada um dos quatro sinais a serem exportados deve ser convertido para um padrão de 16 bits.

Assim, é crucial compreender as especificações de recepção da DAC. Na implementação em questão, a recepção ocorre através de dois canais de 16 bits independentes que seguem o padrão DDR (*Double-Data-Rate*). Nessa configuração, cada entrada processa um sinal durante a subida e descida do clock, permitindo a exportação simultânea de até quatro resultados da FPGA.

Portanto, além de converter valores de ponto flutuante para ponto fixo de 16 bits, a interface de exportação deve funcionar como um transmissor DDR. Assim, os resultados da simulação chegam devidamente padronizados à DAC, podendo ser convertidos para formato analógico para leitura através de um osciloscópio ou pelos conversores analógico-digitais (ADC) dos controladores.

3.3 Bancada Experimental e Plano de Simulações

A Bancada Experimental para a implementação é composta dos seguintes equipamentos:

1. FPGA Virtex 7 VC707 *Evaluation Kit* da Xilinx,
2. DAC DAC34SH84 EVM da Texas Instruments,
3. DSP LAUNCHXL-F28379D da Texas Instruments,
4. Adaptadores FMC (*FPGA Mezzanine Card*),
5. Protoboard,
6. Osciloscópio DL850EV da Yokogawa.

Na Figura 3.5 é possível ver a montagem da bancada que será utilizada para as simulações.

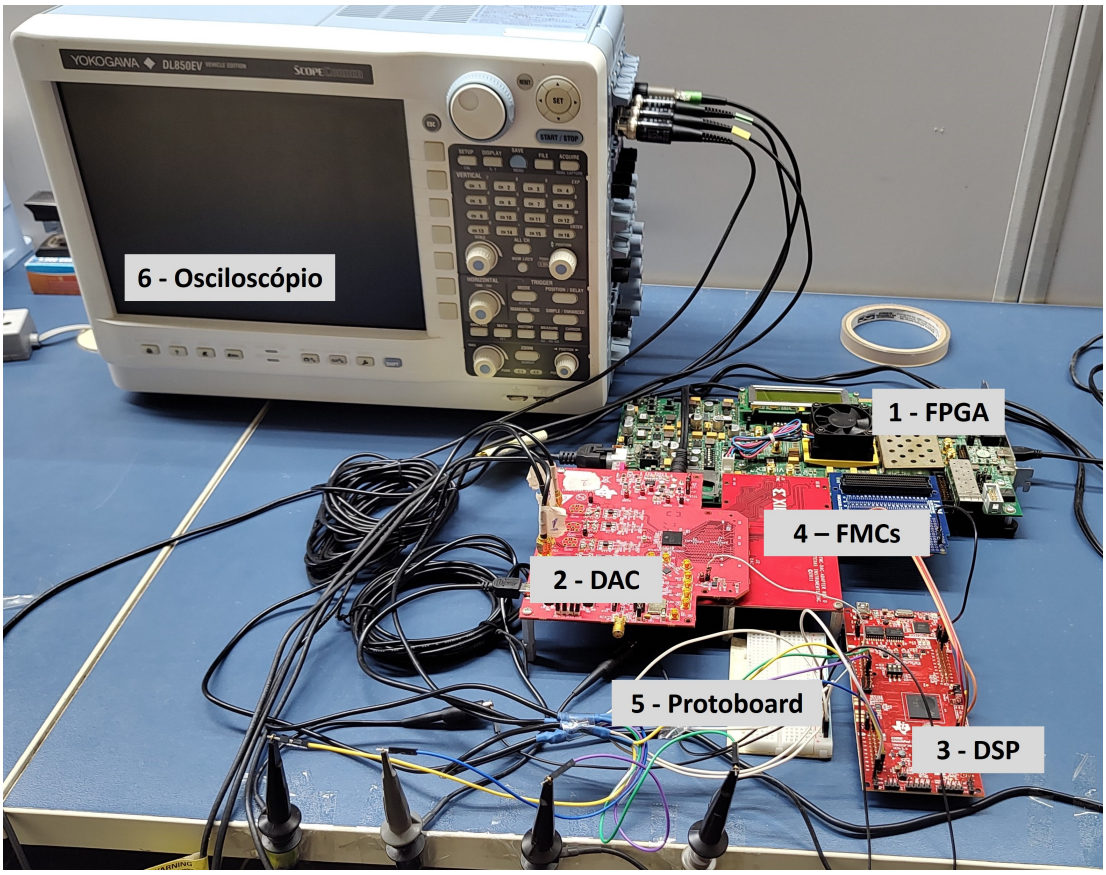

Figura 3.5: Bancada montada para a simulação em tempo real.

A Figura 3.6 apresenta o diagrama esquemático da bancada montada. A FPGA contém o modelo da DFIG e exporta seus resultados de maneira digital para a DAC. A DAC converte esses resultados em sinais analógicos, que são enviados para um osciloscópio, permitindo a visualização temporal dos dados, e para uma DSP, onde funcionam como entradas para controlar os chaveamentos do conversor. Por fim, a DSP retorna os sinais de chaveamento à FPGA, novamente em formato digital. Toda a interface com a FPGA é realizada por meio de FMCs, que são placas adaptadoras responsáveis por compatibilizar os módulos de entrada e saída da FPGA com um padrão de comunicação específico. Nesta bancada, são utilizadas FMCs tanto entre a FPGA e a DAC quanto entre a DSP e a FPGA.

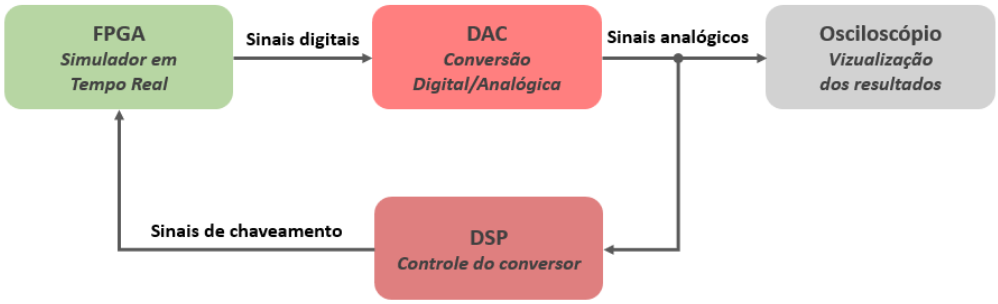

Figura 3.6: Diagrama da montagem da bancada.

A DAC utilizada possui 4 entradas digitais de 16 bits [22]. No entanto, como visto na subseção 2.3.4, para a simulação HIL do sistema DFIG são necessários 7 entradas para o controle do GSC (V_{DC}, v_a, v_b, v_c, i_s^a, i_s^b, i_s^c) e 8 entradas para o controle RSC (q, ω_r, v_a, v_b, v_c, i_s^a, i_s^b, i_s^c), totalizando 10 entradas diferentes. Por essa razão, escolheu-se realizar três simulações.

A primeira terá a máquina apenas em modo de velocidade constante, utilizado para o acionamento da DFIG. Neste modo, a máquina gira com o rotor travado em uma determinada velocidade. Por tal razão o controle do RSC fica desabilitado nessa simulação. Ainda, para reduzir o número de entradas da simulação, o PLL, apresentado na subseção 2.3.4, será implementado na FPGA, juntamente com o modelo da DFIG. Na FPGA também será já realizada a Trasformada de Park transformar as correntes i_s^a, i_s^b e i_s^c em i_s^d e i_s^q, no referencial de Park, que será utilizada no loop interno de controle de corrente do GSC. Portanto, dessa forma, é possível ter o controle do lado da rede do conversor com 4 entradas digitais na DAC, como desejado: v_{dc}, θ_{PLL}, i_s^d, i_s^q. Nesta simulação, foram incluídos, portanto, dois módulos adicionais na modelagem da DFIG no HLS apresentada da subseção 3.2.2.

Na segunda simulação será avaliado em DSP o controle do RSC. Para tanto, todo o controle do barramento DC do conversor será implementado internamente na FPGA. Aqui, o controle do RSC será implementado na DSP. Da mesma maneira como foi feito anteriormente, alguns elementos serão implementados na FPGA, são eles o estimador de fluxo , apresentado na subseção 2.3.4, e o cálculo de potência reativa. Assim, é possível fazer o controle do RSC com 3 sinais digitais da FPGA para a DSP, sendo eles: q, ω_r e θ_s. Nesta etapa, 3 funções adicionais são incluídas na modelagem apresentada na subseção 3.2.2: uma que modela o estimador de fluxo, outra que faz o caliculo instantâneo da potência reativa injetada na rede e uma terceira responsável pela geração dos sinais de chaveamento do GSC.

A última simulação terá uma configuração similar à segunda, onde o controle do RSC será implementado na DSP enquanto a FPGA terá embarcado o controle do GSC. No entando, enquanto na segunda simulação será validado a inicialização do controle do RSC, na segunda será avaliado a performance do controle frente uma

variação na velocidade do vento. A única diferença de configuração com relação à anterior é que serão exportados 4 sinais da FPGA: v_{dc}, q, ω_r e θ_s.

Tabela 3.4: Estrutura de cada simulação realizada.

Simulação	Controle na DSP	Controle na FPGA	FPGA → DSP	DSP → FPGA
1	Controle do GSC	PLL Correntes de Park do GSC	V_{DC} θ_{PLL} i_s^d i_s^q	Chaveamento do GSC
2	Controle do RSC	Controle do GSC Cálculo de q Estimador de Fluxo	q ω_r θ_s	Chaveamento do RSC
3	Controle do RSC	Controle do GSC Cálculo de q Estimador de Fluxo	V_{DC} q ω_r θ_s	Chaveamento do RSC

A Tabela 3.4 mostra o que será implementado em termos de controle na DSP e na FPGA. Nesta será também incluído o modelo da DFIG apresentado na seção 3.2. Além disso, são apresentados os sinais que serão trocados entre a FPGA e a DSP em cada simulação.

Capítulo 4

Resultados

Neste capítulo são apresentadas as simulações realizadas e os resultados obtidos, que são comparados com um modelo equivalente na plataforma de simulação *offline* PSCAD. Os dados da DFIG simulada são apresentados na Tabela 4.1.

Tabela 4.1: Parâmetros da DFIG utilizados nas simulações.

Máquina de Indução de Rotor Bobinado		
R_s	Resistência por fase do estator	0.00286 Ω
R_r	Resistência por fase do rotor	0.00321 Ω
L_s	Indutância por fase do estator	0.14 mH
L_r	Indutância por fase do rotor	0.15 mH
L_m	Indutância mútua	6.31 mH
J	Momento de inércia	0.55 kg.m^2
p	Número de polos	2
Conversor *Back-to-Back*		
L_c	Indutância por fase do filtro AC	20 mH
C_c	Capacitância do barramento CC	1 mF
Turbina Eólica		
r	Raio do rotor da turbina	40 m^2
v_w	Velocidade do vento	10 m/s
ρ	Densidade do ar	1.225
C_p	Eficiência aerodinâmica	0.26
Rede AC		
V_g	Amplitude da tensão da rede	0.69 kV
f_g	Frequência da tensão da rede	60 Hz

Além disso, todas as simulações realizadas consideram um passo de simulação de $h = 10\mu s$ e PWM triangular com frequência de chaveamento de 10 kHz.

4.1 Simulação 1 - Controle do GSC

Como mencionado na seção 3.3, a primeira simulação realizada terá apenas o controle do lado GSC do conversor. A Figura 4.1 esquematiza os elementos implementados na FPGA e a sua interface com a DSP, que terá o controle GSC embarcado.

Aqui, busca-se verificar se o modelo implementado em FPGA corresponde ao modelo desenvolvido em PSCAD quando submetido apenas ao controle de tensão do GSC. A máquina irá girar em velocidade constante e o controle do RSC estará inativo.

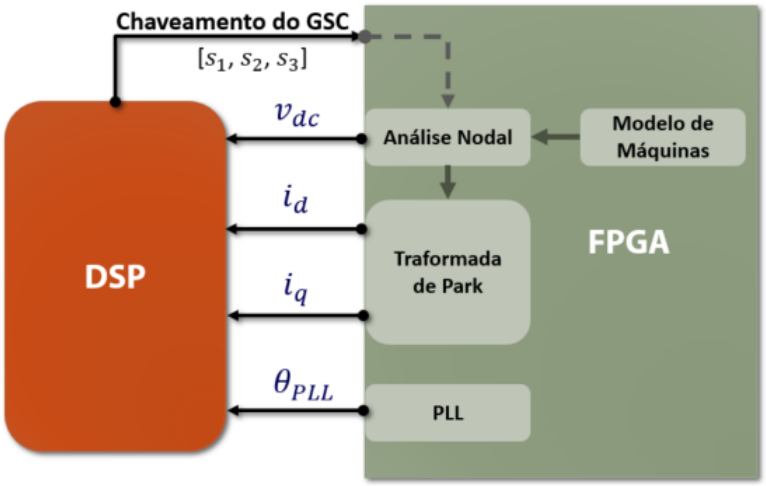

Figura 4.1: Esquema montado para a primeira simulação.

Uma vez que apenas as 3 pernas do lado do GSC estarão sendo controladas nesta etapa, são necessários armazenar $2^{N_s} = 2^3 = 8$ matrizes 26x26 na memória da FPGA.

As variáveis que são exportadas da FPGA são, a tensão do barramento CC do conversor Back-to-back (v_{dc}), o ângulo do PLL (θ_{PLL}), a corrente de eixo direto (i_d) e a corrente de quadratura (i_q). O ângulo do PLL é exportado em radianos enquanto todas as outras unidades são exportadas em pu. A Figura 4.2 mostra os resultados da simulação e sua comparação com o PSCAD.

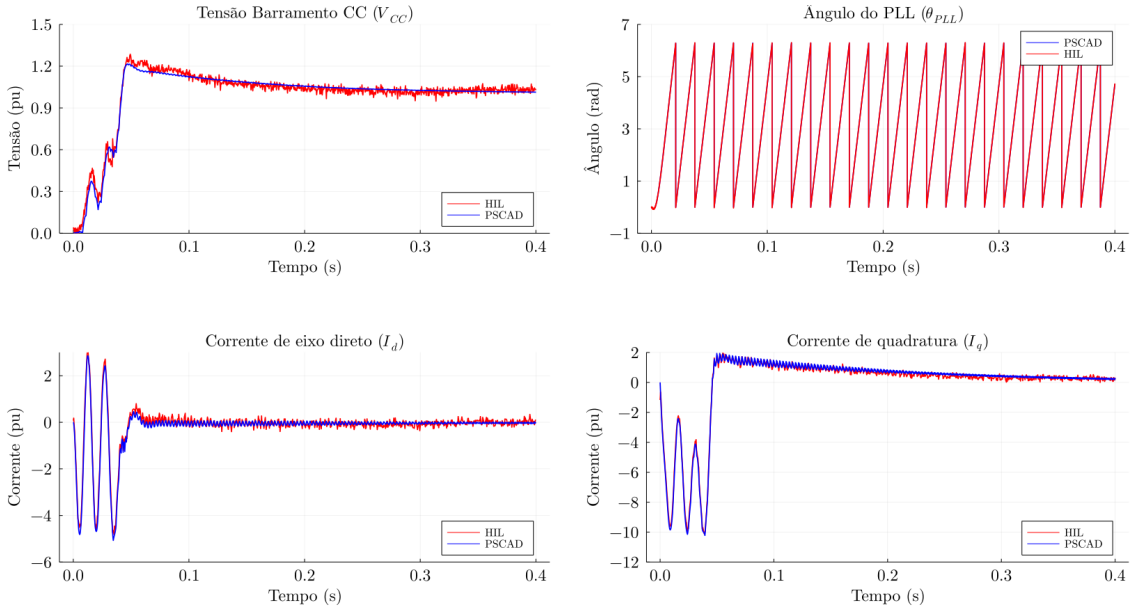

Figura 4.2: Saídas analógicas da simulação 1.

Tabela 4.2: Métricas para avaliação dos resultados da simulação 1

Saída	MSE	MAPE
V_{CC}	0.0005 pu	1.78 %
θ_{PLL}	0.0002 rad	1.12 %
I_d	0.0208 pu	1.47 %
I_q	0.0225 pu	3.51 %

Nota-se que o controle do GSC se mostrou efetivo na simulação em tempo real, controlando a tensão do barramento CC do conversor em 1 pu. Da Figura 4.2 é possível observar o alto nível de aderência entre a simulação em tempo real e a simulação offline. A Tabela 4.2 mostra os erros quadráticos médios (MSE) e os erros percentuais absolutos (MAPE) das saídas da simulação 1. Observa-se que os erros quadráticos são muito baixos e todos os erros percentuais estão abaixo de 4%, mostrando a coincidência dos resultados da simulação HIL com a simulação offline.

4.2 Simulação 2 - Controle do RSC

Na segunda simulação, o controle do GSC será implementado internamente na FPGA, como mostra o digrama da Figura 4.3. O controle do RSC, embracado na DSP, é ativo no momento no qual a máquina troca de modo de operação passando de velocidade contante para torque contante. Na simulação realizada, essa variação ocorre após 1 segundo, acionado por um *timer* incluído na arquitetura de

simulação em VHDL. Além disso, o controle de potência reativa está com ordem de 0.5 pu enquanto o controle de velocidade está com ordem de 1 pu. A Figura 4.3 mostra os sinais enviados e recebidos pelo simulador e o ele inclui.

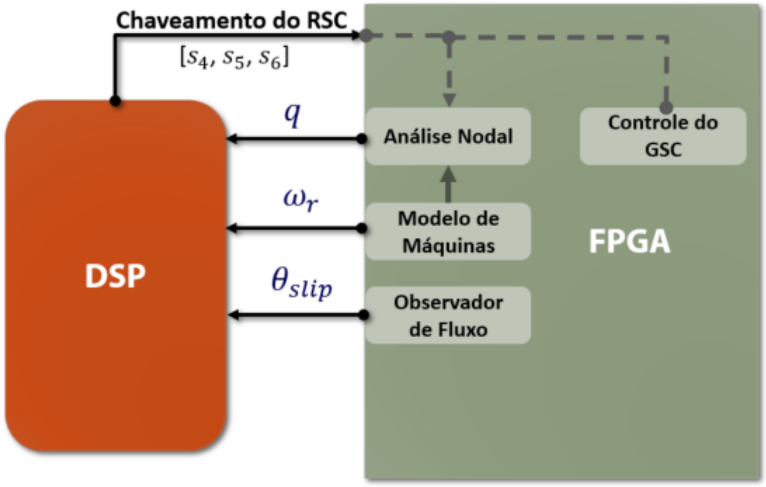

Figura 4.3: Esquema montado para a segunda simulação.

Para o controle do RSC é necessário armazenar $2^{N_s} = 2^6 = 64$ matrizes 26x26 na memória da FPGA para a execução da Análise Nodal Modificada Aumentada.

As variáveis exportadas pelo FPGA são a potência reativa injetada na rede (q), a velocidade de rotação da máquina de indução (ω_r) e o ângulo de escorregamento (θ_{slip}). Este é exportado em radianos enquanto as outras grandezas são exportadas em pu. A Figura 4.4 mostra os resultados da simulação e sua comparação com o PSCAD.

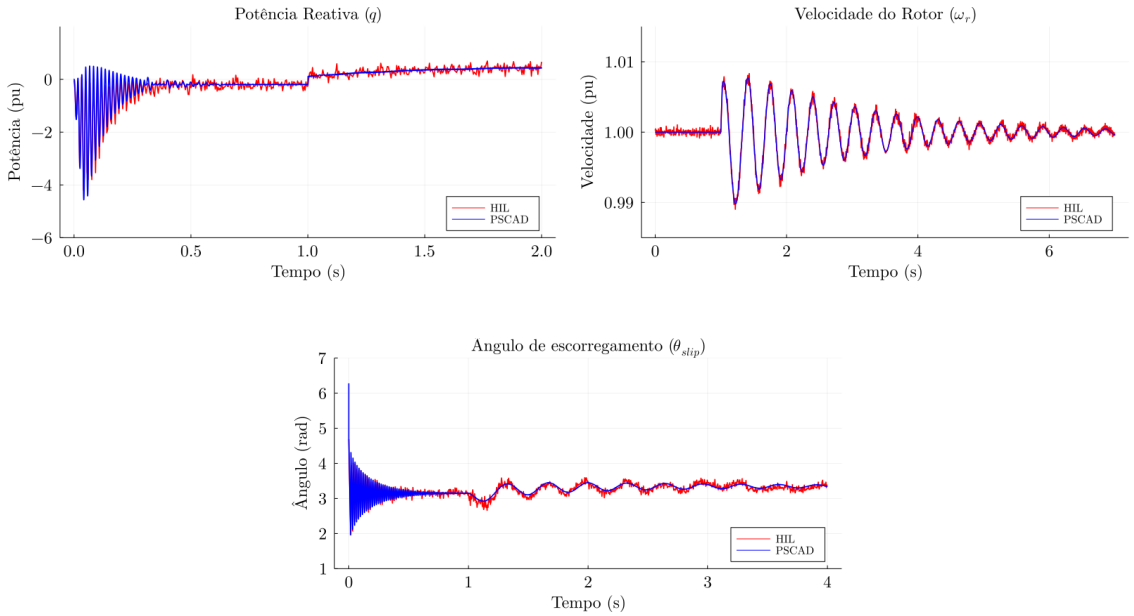

Figura 4.4: Saídas analógicas da simulação 2.

É possível notar que as curvas entre as simulações em tempo real e offline possuem grande aderência. Ademais, ambos os controles de potência e velocidade são capazes de manter as respectivas grandezas nos valores desejados. A Tabela 4.3 mostra os erros quadráticos médios e os erros percentuais absolutos referentes aos resultados da segunda simulação.

Tabela 4.3: Métricas para avaliação dos resultados da simulação 2

Saída	MSE	MAPE
q	0.0197 pu	2.31 %
ω_r	0.0001 pu	0.03%
θ_{slip}	0.0063 rad	1.79 %

É possível verificar que todas as saídas possuem erro percentual menor que 3 %, indicando concordância entre a simulação real e a simulação offline. Ademas, o controle do RSC se mostra efetivo, uma vez que consegue controlar a injeção de potência reativa em 0.5 pu e a velocidade do rotor da máquina de indução ao redor de 1.0 pu.

4.3 Simulação 3 - Variação no sinal de entrada

Na terceira simulação é realizado uma mudança em forma de degrau no sinal de velocidade do vento de $v_w = 10m/s$ para $v_w = 12m/s$. Assim como na simulação 2, o controle do GSC será implementado internamente na FPGA enquanto o RSC terá o seu controle implementado em DSP. Os sinais exportados do simulador são os mesmos da simulação 2 com a adição da tensão do barramento CC (v_{dc}) do conversor, para que a efetividade do controle do GSC também possa ser verificada.

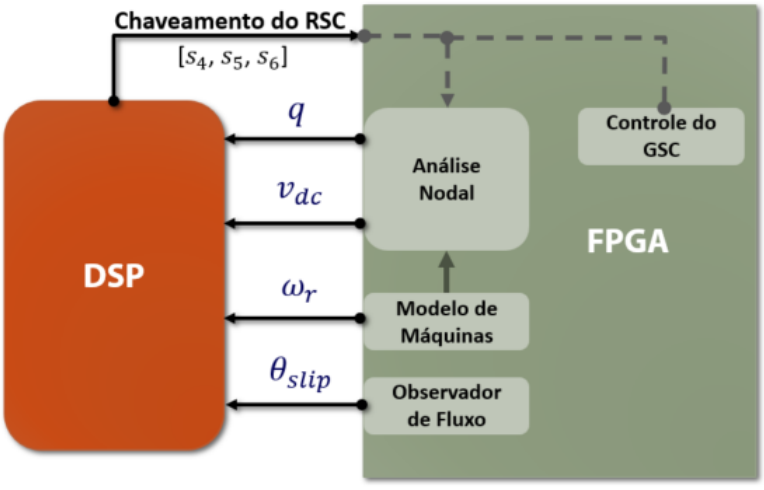

Figura 4.5: Esquema montado para a terceira simulação.

O momento de variação na velocidade do vento é comandado pelo usuário através de um *push button* na FPGA. Esse momento é apresentado como $t = 0s$ na Figura 4.6, que mostra os resultados da simulação em tempo real em comparação com o PSCAD.

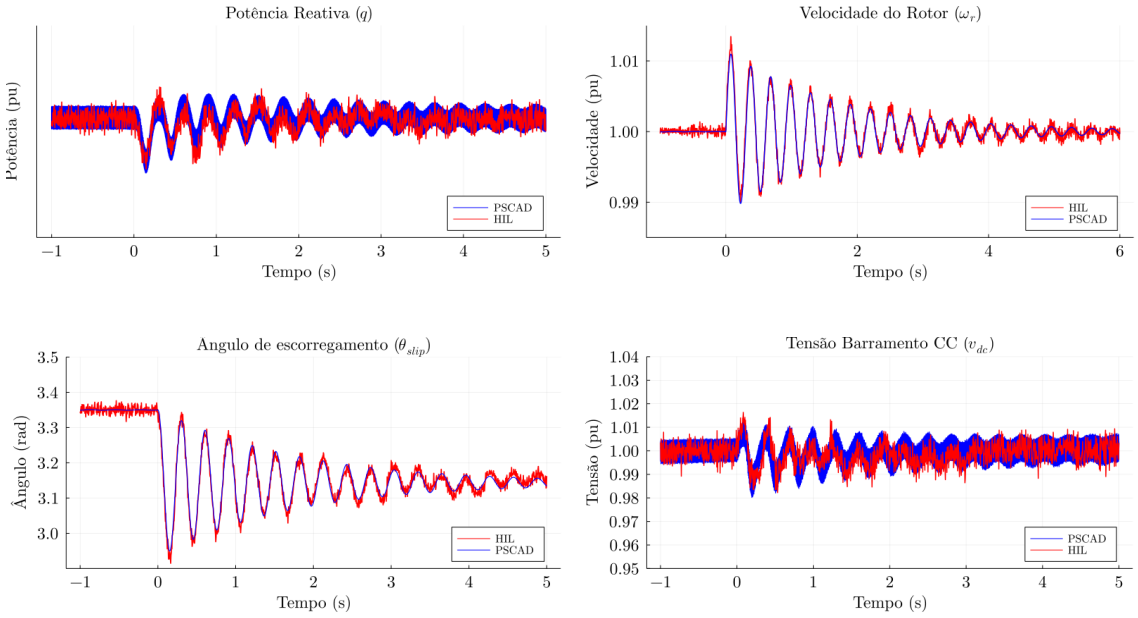

Figura 4.6: Saídas analógicas da simulação 3.

Tabela 4.4: Métricas para avaliação dos resultados da simulação 3

Saída	MSE	MAPE
q	0.0001 pu	0.05 %
ω_r	0.0002 pu	0.04 %
θ_{slip}	0.0182 rad	0.35 %
v_{dc}	0.155 pu	0.77 %

Observa-se que o comportamento dos sinais do HIL são muito similares aos sinais da simulação offline realizada no PSCAD. A aderência das duas curvas pode ser comprovada na Tabela 4.4, onde mostra erros quadráticos médios baixos e erros percentuais absolutos que não ultrapassam 1%. Ademais, os controles se mostraram efetivos em manter as variáveis controladas nos valores especificados.

Destaca-se também que todas as simulações são garantidamente em tempo real dado que em nenhum momento a máquina de estados entrou em estado de erro. Como mostrado na subseção 3.2.1, se em alguma etapa da simulação, o tempo computacional é maior que o passo de simulação a máquina de estados entra em modo de erro e os cálculos da simulação são interrompidos, não sendo, portanto, permitido nenhum *overrun* na simulação.

Um aspecto que deve ser destacado é que todas as simulações estão garantidamente em tempo real, uma vez que, em nenhum momento, a máquina de estados entrou em estado de erro, pois caso contrário, a máquina de estados entra em modo de erro e os cálculos são interrompidos, não permitindo, assim, qualquer ocorrência de *overrun* na simulação.

Capítulo 5

Conclusões

Este trabalhou apresentou uma modelagem para transitórios eletromagnéticos de aerogeradores DFIG e uma simulação em tempo real baseada na configuração C-HIL com simulador baseado em FPGA. A modelagem integra um modelo em espaço de estados da máquina e uma modelagem baseada em análise nodal modificada aumentada do conversor e da rede elétrica.

Devido à limitação no número de sinais exportáveis pelo simulador, foram realizadas duas simulações iniciais para validar as implementações dos controles do GSC e do RSC na DSP. Na segunda, o controle do GSC foi implementado internamente na FPGA. Os resultados dessas simulações mostrou uma alta capacidade do simulador em emular o comportamento do aerogerador implementado em PSCAD na simulação *offline*.

A terceira simulação teve como objetivo avaliar o comportamento do sistema modelado quando submetido a uma abrupta variação na velocidade do vento. Nessas condições, o simulador também apresentou um comportamento aderente ao observado na simulação *offline*.

Sendo assim, o simulador em tempo real implementado teve um desempenho satisfatório quando analisamos as aderências das curvas e as medidas estatísticas de erros quadráticos e erros absolutos, permanecendo este sempre abaixo de 4 %. A possibilidade de implementar um modelo complexo e detalhado para transitórios eletromagnéticos e com elementos de chaveamento em uma plataforma como a FPGA, trás uma alternativa mais barata se comparada com simuladores em tempo real comerciais.

Uma alternativa mais barata para fazer simulações C-HIL é fundamental para realizar testes e simulações para o comissionamento de equipamentos caros e de comportamento dinâmico complexo. Além de ser mais barata, a FPGA possui alta versatilidade para modelar diferentes sistemas elétricos de potência e simulações com os mais variados fins. No entanto, como desvantagem, as FPGAs demandam um alto tempo de modelagem.

Dessa forma, o simulador implementado pode ter grande utilidade no comissionamento de sistemas de controle para aerogeradores testados através de simulações *Hardware-in-the-loop* o que diminui enormemente os custos envolvidos neste processo.

Como trabalhos futuros, propõe-se o testes de controles mais avançados e situações operativas variadas e de maior estresse da máquina de indução, do conversor e da rede etérica. Uma melhor modelagem da rede elétrica é também possível utilizando equivalentes no domínio da frequência por exemplo (FDNE, *Frequency-Dependent Network Equivalent*). Outras adições no modelo de máquinas são também possíveis como a inclusão da sua saturação e de efeitos de alta frequência.

Um limitador neste trabalho foi a limitação a quatro saídas analógicas do simulador. Expandi-las tornaria mais fácil a implementação de mais de um controle ou de controles que necessitem de mais do que quatro sinais de realimentação. Com tal melhoria, tornaria-se possível a implementação de controles mais complexos sem a necessidade de fazer separações nos testes ou simulações adicionais, lembrando que o tempo de desenvolvimento é limitante em se tratando de implementações em FPGA quando comparado com simuladores comerciais.

Referências Bibliográficas

[1] KUNDUR, P. S., MALIK, O. P. *Power system stability and control*. McGraw-Hill Education, 2022.

[2] KIZILCAY, M., HOIDALEN, H. K., AMETANI, A. *Numerical Analysis of Power System Transients and Dynamics*. IET, 2015.

[3] MAHSEREDJIAN, J., DENNETIÈRE, S., DUBÉ, L., et al. "On a new approach for the simulation of transients in power systems", *Electric Power Systems Research*, v. 77, n. 11, pp. 1514–1520, 2007.

[4] WATSON, N., ARRILLAGA, J. *Power systems electromagnetic transients simulation*. 2003.

[5] BÉLANGER, J., VENNE, P., PAQUIN., J.-N. "The what, where and why of real-time simulation", pp. 25–29, 2010.

[6] AMETANI, A. "Electromagnetic transients program: History and future", *IEEJ transactions on electrical and electronic engineering*, v. 16, n. 9, pp. 1150–1158, 2021.

[7] FAROOQ, U., MARRAKCHI, Z., MEHREZ, H. *Tree-based heterogeneous FPGA architectures: application specific exploration and optimization*. Springer Science & Business Media, 2012.

[8] ABEEÓLICA. *Boletim Anual de Geração Eólica 2021*. In: Report, ABEEólica, 2022.

[9] IRENA, CPI. *Gloabl Landscpae of Renewable Energy Finace*. In: Report ISBN: 978-92-9260-523-0, IRENA, Abu Dhabi, 2023.

[10] KUON, I., TESSIER, R., ROSE., J. "FPGA architecture: Survey and challenges", *Foundations and Trends in Electronic Design Automation*, v. 2, n. 2, pp. 135–253, 2008.

[11] WINTERSTEIN, F., BAYLISS, S., A., G. "High-level synthesis of dynamic data structures: A case study using Vivado HLS." *2013 International conference on field-programmable technology (FPT)*, pp. 362–365, 2013.

[12] ACKERMANN, T. *Wind power in power systems.* John Wiley Sons, 2012.

[13] NGUYEN, H. M., NAIDU, D. S. "Advanced control strategies for wind energy systems: An overview." *2011 IEEE/PES Power Systems Conference and Exposition.*, pp. 1–8, 2011.

[14] S., M. "Generalized Theory of Electrical Machines—A review." *Int. J. Res. Sci. Innov*, v. 3, n. 8, pp. 67–71, 2016.

[15] ADKINS, B., HARLEY, R. G. *The general theory of alternating current machines: application to practical problems.* Springer, 2013.

[16] THEODORO, T. S. "Simulação híbrida no domínio do tempo de transitórios eletromecânicos e eletromagnéticos: integração de um aerogerador de indução duplamente excitado". 2016.

[17] POLINDER, H., VAN DER PIJL, F. F., DE VILDER, G. J., et al. "Comparison of direct-drive and geared generator concepts for wind turbines." *IEEE Transactions on energy conversion*, v. 21, n. 3, pp. 725–733, 2006.

[18] DEHONG XU, FREDE BLAABJERG, W. C. N. Z. *Advanced control of doubly fed induction generator for wind power systems.* John Wiley & Sons, 2018.

[19] SHAO, S., A. E. B. F. . M. R. "Stator-flux-oriented vector control for brushless doubly fed induction generator." *IEEE Transactions on Industrial Electronics*, v. 56, n. 10, pp. 4220–4228, 2009.

[20] HO, C.-W., RUEHLI, A., BRENNAN, P. "The modified nodal approach to network analysis." *IEEE Transactions on circuits and systems*, v. 22, n. 6, pp. 504–509, 1975.

[21] DOMMEL, H. W. "Digital computer solution of electromagnetic transients in single-and multiphase networks." *IEEE transactions on power apparatus and systems*, , n. 4, pp. 388–399, 1969.

[22] *DAC34SH84 Quad-Channel, 16-Bit, 1.5 GSPS Digital-to-Analog Converter (DAC).* Texas Instruments, 9 2015. Rev. 3.

Apêndice A

Obtenção das matrizes para a formulação em de Espaço de Estados

Um passo fundamental para a modelagem utilizando espaço de estados é a obtenção da matriz de estados ($\mathbf{A}$). Se tomamos como variáveis de estados (x) os fluxos concatenados, tem-se:

$$x = \begin{bmatrix} \lambda_s^d \\ \lambda_s^q \\ \lambda_r^d \\ \lambda_r^q \end{bmatrix} = \underbrace{\begin{bmatrix} L_s + L_m & 0 & L_m & 0 \\ 0 & L_s + L_m & 0 & L_m \\ L_m & 0 & L_m + L_r & 0 \\ 0 & L_m & 0 & L_m + L_r \end{bmatrix}}_{\mathbf{L}} \begin{bmatrix} i_s^d \\ i_s^q \\ i_r^d \\ i_r^q \end{bmatrix} \tag{A.1}$$

Ou em termo das correntes nos referenciais direta e de quadratura:

$$\begin{bmatrix} i_s^d \\ i_s^q \\ i_r^d \\ i_r^q \end{bmatrix} = \mathbf{L}^{-1} \begin{bmatrix} \lambda_s^d \\ \lambda_s^q \\ \lambda_r^d \\ \lambda_r^q \end{bmatrix} \tag{A.2}$$

É possível mostrar que $\mathbf{L}^{-1}$ é nula nos mesmos termos onde L também o é. Assim, escreve-se as correntes como:

$$i_s^d = \mathbf{L}_{22}^{-1}\lambda_s^d + \mathbf{L}_{24}^{-1}\lambda_r^d \tag{A.3a}$$

$$i_s^q = \mathbf{L}_{11}^{-1}\lambda_s^q + \mathbf{L}_{13}^{-1}\lambda_r^q \tag{A.3b}$$

$$i_r^d = \mathbf{L}_{41}^{-1}\lambda_s^d + \mathbf{L}_{44}^{-1}\lambda_r^d \tag{A.4a}$$

$$i_r^q = \mathbf{L}_{31}^{-1}\lambda_s^q + \mathbf{L}_{33}^{-1}\lambda_r^q \tag{A.4b}$$

onde $\mathbf{L}_{ij}^{-1}$ é o elemento da matriz $\mathbf{L}^{-1}$ na linha i e na coluna j.

Das equações de tensão da máquina de rotor bobinado, é possível isolar a derivada dos fluxos concatenados ($\dot{x}$), resultando em:

$$\dot{\lambda_s^d} = -R_s i_s^d - \omega\lambda_s^q + v_s^d \tag{A.5a}$$

$$\dot{\lambda_s^q} = -R_s i_s^q + \omega\lambda_s^d + v_s^q \tag{A.5b}$$

$$\dot{\lambda_r^d} = -R_r i_r^d - (\omega - \omega_r)\lambda_r^q + v_r^d \tag{A.6a}$$

$$\dot{\lambda_r^q} = -R_r i_r^q + (\omega - \omega_r)\lambda_r^d + v_r^q \tag{A.6b}$$

Substituindo A.3 e A.4 em A.5 e A.6:

$$\dot{\lambda_s^d} = -R_s(\mathbf{L}_{11}^{-1}\lambda_s^d + \mathbf{L}_{13}^{-1}\lambda_r^d) - \omega\lambda_s^q + v_s^d \tag{A.7a}$$

$$\dot{\lambda_s^q} = -R_s(\mathbf{L}_{22}^{-1}\lambda_s^q + \mathbf{L}_{24}^{-1}\lambda_r^q) + \omega\lambda_s^d + v_s^q \tag{A.7b}$$

$$\dot{\lambda_r^d} = -R_r(\mathbf{L}_{31}^{-1}\lambda_s^d + \mathbf{L}_{33}^{-1}\lambda_r^d) - (\omega - \omega_r)\lambda_r^q + v_r^d \tag{A.8a}$$

$$\dot{\lambda_r^q} = -R_r(\mathbf{L}_{41}^{-1}\lambda_s^q + \mathbf{L}_{44}^{-1}\lambda_r^q) + (\omega - \omega_r)\lambda_r^d + v_r^q \tag{A.8b}$$

E finalmente:

$$\begin{bmatrix} \dot{\lambda_s^d} \\ \dot{\lambda_s^q} \\ \dot{\lambda_r^d} \\ \dot{\lambda_r^q} \end{bmatrix} = \underbrace{\begin{bmatrix} -R_s\mathbf{L^{-1}}_{11} & -\omega & -R_s\mathbf{L^{-1}}_{13} & 0 \\ \omega & -R_s\mathbf{L^{-1}}_{22} & 0 & -R_s\mathbf{L^{-1}}_{24} \\ -R_r\mathbf{L^{-1}}_{31} & 0 & -R_r\mathbf{L^{-1}}_{33} & \omega_r - \omega \\ 0 & -R_r\mathbf{L^{-1}}_{42} & \omega - \omega_r & -R_r\mathbf{L^{-1}}_{44} \end{bmatrix}}_{\mathbf{A}} \begin{bmatrix} \lambda_s^d \\ \lambda_s^q \\ \lambda_r^d \\ \lambda_r^q \end{bmatrix} - \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{B}} \begin{bmatrix} v_s^d \\ v_s^q \\ v_r^d \\ v_r^q \end{bmatrix}$$

(A.9)

Apêndice B

Integração trapezoidal de modelo em espaço de estados

Para uma função contínua $f(x)$ definida em um intervalo $[a, b]$, a integral pode ser aproximada como:

$$\int_a^b f(x)dx \approx \frac{b-a}{2}\left(f(a)+f(b)\right) \tag{B.1}$$

Essa aproximação pode ser vista na Figura B.1 mostra graficamente esta equação, onde a integral aproximada é a área em vermelho. É evidente que para obter esta aproximação de forma mais precisa, pode-se dividir a curva em mais de um intervalo.

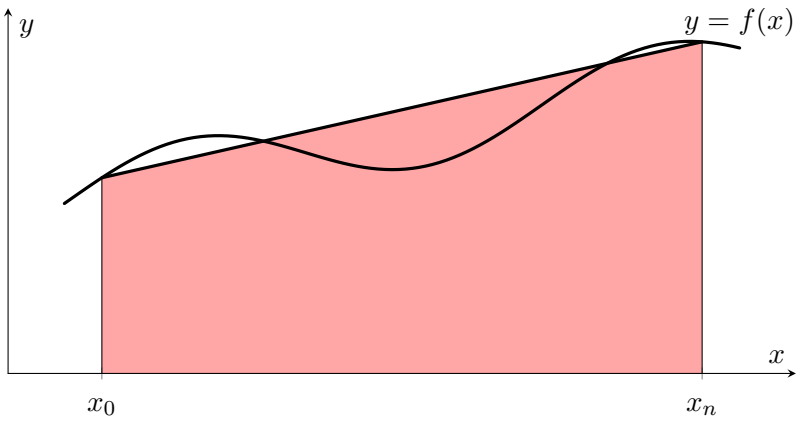

Figura B.1: Aproximação da integral por um trapézio

Neste caso, se o intervalo $[a, b]$ for dividido em n subintervalos de largura uniforme $h = \frac{b-a}{n}$, a aproximação da integral torna-se:

$$\int_a^b f(x)dx \approx \frac{h}{2}\left(f(x_0)+2\sum_{i=1}^{n-1} f(x_i)+f(x_n)\right) \tag{B.2}$$

onde $x_0 = a$, $x_n = b$, e $x_i = a + i \cdot h \ \forall i$. A Figura B.2 mostra na região colorida a

aproximação da integral como a soma das áreas dos trapézios gerados.

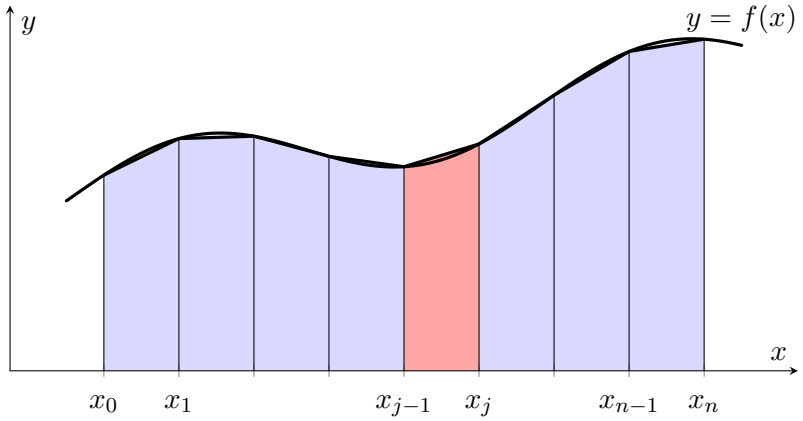

Figura B.2: Integração Trapezoidal

Para um ponto arbitrário $a < x_j$ com $0 < j$ onde A é a área delimitada pelo trapézio formado por x_j e por x_{j-1}, se obtém:

$$A = \int_{x_{j-1}}^{x_j} f(x) = \frac{h}{2}(f(x_j) - f(x_{j-1})) \tag{B.3}$$

Aplicando a derivada em relação a x de ambos os lados da equação:

$$f(x_j) - f(x_{j-1}) = \frac{h}{2}(\dot{f}(x_j) - \dot{f}(x_{j-1})) \tag{B.4}$$

Se $f(x) = \mathbf{x_j}$, onde $\mathbf{x_j}$ é o vetor de variáveis de estado no passo j, obtêm-se:

$$\mathbf{x_j} = \mathbf{x_{j-1}} + \frac{h}{2}(\dot{\mathbf{x}}_\mathbf{j} + \dot{\mathbf{x}}_\mathbf{j-1}) \tag{B.5}$$

Substituindo (2.17) em (B.5):

$$\mathrm{x}_j = \mathbf{x_{j-1}} + \tfrac{h}{2}(\mathbf{A}\mathbf{x_j} + \mathbf{B}\mathbf{u_j} + \mathbf{A}\mathbf{x_{j-1}} + \mathbf{B}\mathbf{u_{j-1}})$$

$$\mathrm{x}_j\Big(\mathbf{I} - \tfrac{h}{2}\mathbf{A}\Big) = \Big(\tfrac{h}{2}\mathbf{A} + \mathbf{I}\Big)\mathbf{x_{j-1}} + \tfrac{h}{2}\mathbf{B}(\mathbf{u_j} + \mathbf{u_{j-1}})$$

$$\mathrm{x}_j = \Big[\mathbf{I} - \tfrac{h}{2}\mathbf{A}\Big]^{-1} \times \Big[\Big(\tfrac{h}{2}\mathbf{A} + \mathbf{I}\Big)\mathbf{x_{j-1}} + \tfrac{h}{2}\mathbf{B}\Big(\mathbf{u_j} + \mathbf{u_{j-1}}\Big)\Big] \tag{B.6}$$