Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Programa de Pós-Graduação de Engenharia Elétrica

DÊNIS RICARDO HÜLLER

SIMULAÇÃO EM TEMPO REAL DA MÁQUINA SÍNCRONA A IMÃ PERMANENTE IMPLEMENTADA EM DISPOSITIVOS LÓGICOS PROGRAMÁVEIS (FPGA - *FIELD PROGRAMMABLE GATE ARRAY*)

Campina Grande, Paraíba, Brasil

DÊNIS RICARDO HÜLLER

SIMULAÇÃO EM TEMPO REAL DA MÁQUINA SÍNCRONA A IMÃ PERMANENTE IMPLEMENTADA EM DISPOSITIVOS LÓGICOS PROGRAMÁVEIS (FPGA - *FIELD PROGRAMMABLE GATE ARRAY*)

Dissertação de Mestrado submetida à Coordenação dos cursos de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande em cumprimento às exigências para a obtenção do Título de Mestre em Engenharia Elétrica.

Área de Concentração: Processamento da Energia

Orientadores:

Mauricio Beltrão de Rossiter Corrêa, D.Sc.

Eisenhawer de Moura Fernandes, D.Sc.

Campina Grande, Paraíba, Brasil

FICHA CATALOGÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG

H913s Hüller, Dênis Ricardo.

Simulação em tempo real da máquina síncrona a imã permanente implementada em dispositivos lógicos programáveis (FPGA – Field Programmable Gate Array) / Dênis Ricardo Hüller. – Campina Grande, 2014.

137 f.: il. color.

Dissertação(Mestrado em Engenharia Elétrica) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2014.

"Orientação: Prof. Dr. Maurício Beltrão de Rossiter Corrêa, Prof. Dr. Eisenhawer de Moura Fernandes".

Referências.

1. Simulador em Tempo Real. 2. Hardware In The Loop. 3. FPGA 4. Máquina Síncrona. I. Corrêa, Maurício Beltrão Rossiter. II. Fernandes, Eisenhawer de Moura. III Título.

CDU 621.391.8(043)

"SIMULAÇÃO EM TEMPO REAL DA MÁQUINA SÍNCRONA A IMÃ PERMANENTE IMPLEMENTADO EM FPGA"

DENIS RICARDO HÜLLER

DISSERTAÇÃO APROVADA EM 29/05/2013

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

Eisenhawer de Moura Fernandes

EISENHAWER DE MOURA FERNANDES, D.Sc., UFCG
Orientador(a)

ANTONIO MARCUS NOGUEIRA LIMA, Dr., UFCG
Examinador(a)

MARCOS RICARDO DE ALCÂNTARA MORAIS, D.Sc, UFCG
Examinador(a)

CAMPINA GRANDE - PB

Dedicatória

Dedico este trabalho a todos os meus familiares, em especial os meus pais Dari Hüller e Martha Elisabeth Hüller, minha esposa Keylha Santana Hüller e a minha filha Iasmin Santana Hüller, pois juntos formam a base fundamental que me impulsiona a alcançar meus objetivo e lutar pela realização dos meus sonhos.

Agradecimentos

Agradeço primeiramente a Deus, por ter permitido momentos de realizações como esse.

A Universidade Federal de Campina Grande, ao Programa de Pós Graduação em Engenharia Elétrica do Centro de Engenharia Elétrica e Informática.

Aos meus orientadores, professor Eisenhawer de Moura Fernandes e professor Maurício Beltrão de Rossiter Corrêa, pela contribuição e o conhecimento transmitido, elemento que foi essencial para a elaboração deste trabalho.

Aos meus professores, que de modo geral, colaboraram para obtenção do título de Mestre, e em especial ao professor Alexandre Cunha Oliveira que sempre me auxiliou nas dificuldades ocorridas durante a pós-graduação.

Aos meus colegas de curso, que contribuíram, apoiando nos momentos de reflexão e discussão dos temas referentes ao estudo.

A todos os meus familiares, pelo incentivo e colaboração. Em especial a minha mãe Martha Elisabeth Hüller, meu pai Dari Hüller, minha esposa e filha, Keylha Santana Hüller e Iasmin Santana Hüller, pelo estímulo, buscando sempre prover um ambiente favorável para os meus estudos.

Enfim, agradeço, a todos, os que direta ou indiretamente, colaboram e caminham comigo nesta trajetória que me permite o título de Mestre em Engenharia Elétrica.

Resumo

A funcionalidade de um controlador de uma máquina precisa ser testada e verificada antes da sua instalação e comissionamento. Simuladores de tempo real são opções seguras para se realizar testes em malha fechada e verificar sua funcionalidade e desempenho. Como os cenários são examinados em tempo real, esta prática reduz substancialmente o tempo total de execução do estudo.

O emprego de dispositivos FPGA em simuladores de tempo real e *hardware-in-the-loop* no projeto e testes de controladores e máquinas vem crescendo atualmente. Estes dispositivos possuem um custo de implementação menor em relação à forma tradicional de simulação, através de *softwares* com processadores operando em paralelo. A versatilidade na reconfiguração dos circuitos dos FPGA e a crescente expansão na capacidade de *hardware* permitem configurar modelos matemáticos cada vez mais complexos.

Este trabalho apresenta uma plataforma de simulação em tempo real com interface *hardware-in-the-loop* para máquinas elétricas, com o foco em máquinas síncronas a imã permanente. A plataforma é composta por um dispositivo FPGA que simula em tempo real um conversor de potência e uma máquina síncrona a imã permanente que, através da sua interface *hardware-in-the-loop,* permite o uso de controladores reais atuando na simulação. Os resultados obtidos pela plataforma são comparados com as máquinas reais e com simulações dos modelos originais executadas em *software*.

Palavras-chave: Simulador em tempo real, *Hardware-in-the-loop*, FPGA, máquina síncrona a imã permanente.

Abstract

The functionality of a controller of a machine needs to be tested and verified prior to installation and commissioning. Real-time simulators are practically the only safe options to perform closed-loop tests and verify its functionality and performance. In addition to substantially reduce the total execution time of the study .

The use of FPGA devices for real-time simulation and hardware-in-the-loop in the design and testing of controllers and machines come currently growing. These devices have a lower cost of deployment compared to the traditional form of simulation, using software with processors operating in parallel. The versatility of the reconfiguration of FPGA circuits and the increasing expansion in capacity to allow you to configure hardware increasingly complex mathematical models.

This paper presents a simulation platform for real-time interface with hardware-in-the-loop for electrical machines, with the focus on the permanent magnet synchronous machines. The platform comprises an FPGA device that simulates real time, a power converter and a synchronous machine with permanent magnet which, through its interface hardware-in-the-loop allows the use of real controllers acting in the simulation. The results obtained by the platform are compared with actual machines and with simulations of the original models implemented in software.

Keywords: Real-time simulator, *Hardware-in-the-loop,* FPGA, permanent magnet synchronous machines.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Símbolos

t_c	Tempo de Computação
Δ_t	Passo de tempo da simulação
HIL	*Hardware in-the-loop*
RCP	*Rapid Control Prototyping* (prototipagem rápida)
FPGA	*Field Programmable Gate Array*
A/D	Conversor analógico digital
PWM	*Pulse Width Modulation*
DSP	*Digital Signal Processor*
HDL	*Hardware Description*
PMSM	Máquina síncrona a imã permanente
SMPM	Máquina a imã permanente com imãs inserido na superfície do rotor
IPM	Máquina a imã permanente com imãs inseridos no interior do rotor
CT	*Cogging Torque*
*	Grandeza de referência
t	Tempo
k	Amostra
s	Variável no domínio da frequência
L_{si}	Indutância própria de fase, com i=1,2,3
L_{s0}	Valor médio da indutância própria
ω_r	Velocidade elétrica do rotor em rad/s
ω_m	Velocidade mecânica do rotor em rad/s
d	Eixo direto
q	Eixo em quadratura
L_{sm}	Amplitude da parte variante da indutância de fase
M_{s12}	Indutância mútua entre as fases 1 e 2
M_{s23}	Indutância mútua entre as fases 2 e 3
M_{s31}	Indutância mútua entre as fases 1 e 3
M_{s0}	Valor médio da indutância mútua
L_{ss}	Matriz de indutâncias
v_{s123}	Vetor das tensões estatóricas
v_{si}	Tensões estatóricas trifásicas, com i=1,2,3

R_s	Matriz de resistências de fase
r_s	Resistência de fase
i_{s123}	Vetor das correntes estatóricas
i_{si}	Correntes estatóricas trifásicas, com i=1,2,3
$\emptyset_{s123}$	Vetor dos fluxos estatóricos
$\emptyset_{si}$	Fluxos estatóricos trifásicos, com i=1,2,3
$\emptyset_{pm}$	Valor de pico do fluxo produzido pelo imã do rotor que enlaça os enrolamentos do estator
$\emptyset_{r123}$	Vetor do fluxo produzido pelo imã do rotor que enlaça os condutores do estator
T_t	Torque desenvolvido pela máquina
T_e	Torque eletromagnético produzido pela máquina
c_e	Conjugado eletromagnético
p	Pares de pólos
T	Transposta de uma matriz ou vetor
$v_{s\alpha\beta}$	Vetor das componentes de tensão em α e β
$i_{s\alpha\beta}$	Vetor das componentes de corrente em α e β
$L_{s\alpha\beta}$	Matriz de indutâncias entre os eixos α e β
$\emptyset_{s\alpha\beta}$	Vetor das componentes de fluxo produzidas pelo imã do rotor em α e β
$L_{s\alpha\alpha}$	Indutância própria do eixo α
$M_{s\alpha\beta}$	Indutância mútua entre α e β
$L_{s\beta\beta}$	Indutância própria do eixo β
Q	Transformação entre o sistema de coordenadas αβ e dq
V_{sd}	Componente da tensão estatórica de eixo d
V_{sq}	Componente da tensão estatórica de eixo q
I_{sd}	Componente da corrente estatórica de eixo d
I_{sq}	Componente da corrente estatórica de eixo q
L_{sd}	Indutância estatórica de eixo d
L_{sq}	Indutância estatórica de eixo q
J	Momento de inércia do rotor da máquina
J_m	Momento de inércia da carga
T_c	Torque de carga
f_ω	Coeficiente de atrito viscoso da máquina
v	Pulso de tensão aplicado em duas fases

i	Corrente circulante em duas fases
q_i	Chaves do conversor de potência, i=1,2,3
$\overline{q_i}$	Chaves complementares do conversor de potência, i=1,2,3
$i_{s\alpha}$	Componente α da corrente do estator
$i_{s\beta}$	Componente β da corrente do estator
N	Número de amostras
E	Tensão no Barramento CC
Ts	Período de amostragem
e_{mq}	Erro médio quadrático
v_{med}	Valor medido
v_{calc}	Valor calculado
pu	Representação por unidade

Sumário

Capítulo 1

Introdução

O uso de *softwares* de simulação proporcionam ferramentas para o estudo e análises de sistema de controle e sistemas físicos, a saber, controle de motores elétricos, desempenho de conversores estáticos de potência, análise de redes elétricas. No entanto, a utilização dessas ferramentas está limitada a condições de estudo onde não há restrição de tempo máximo para geração de cada amostra dos sinais de interesse do sistema sob análise. Ou seja, não se trabalha com respostas em tempo real, visto que, tais programas não operam em tempo-real. O tempo de resposta necessário para a CPU computar o sistema modelado pode levar de segundos a minutos, ordem de magnitude maior do que um sistema real.

O modo de operação destes *softwares* impõe limites para que equipamentos externos sejam interligados com as plantas/sistemas simulados, objetivando, por exemplo, avaliar o comportamento de controladores implementados em um dispositivo externo real.

As simulações em tempo real permitem testar um controlador em malha fechada quando a prática proíbe a utilização de plantas reais, que possa trazer risco aos equipamentos e as pessoas. Para isso é necessário que o simulador seja capaz de aceitar entradas de informações a partir do dispositivo sobre teste e incorporar essas informações continuamente na execução da simulação. Além disso, as equações do sistema devem ser resolvidas em tempo hábil para que as saídas estejam disponíveis antes da chegada de uma próxima amostra (Matar, Abdel-Rahman e Soliman, FPGA - Based Real-Time Digital Simulation 2005), ou matematicamente:

$$t_c = \Delta t$$

onde t_cé o tempo de computação e Δt é o passo de tempo da simulação.

Os controladores de motores eram tipicamente desenvolvidos e testados com o uso de motores reais. Entretanto, testar os controladores utilizando o modelo do motor simulado em

tempo-real é uma alternativa que desperta o interesse pelas suas características. Por exemplo, utilizar um motor simulado permite que os testes sejam feitos em um ambiente controlado permitindo atingir condições extremas, que poderiam danificar um motor real ou o sistema de potência, afetando o custo do protótipo. Para realizar estes testes o controlador é interligado a um motor simulado em tempo real através de entradas e saídas apropriadas, o que caracteriza a simulação *hardware-in-the-loop*.

Simulações em HIL são usualmente utilizadas em um nível de projeto em que seja possível a modelagem da dinâmica do sistema com alto grau de precisão e fidelidade. Essa categoria importante de simuladores baseados em *hardware* digital, os chamados "Simuladores em tempo real", são utilizados para treinamento de usuários e sistemas, prototipagem virtual e ensaios em malha-fechada com dispositivos externos reais para projeto, desenvolvimento e comissionamento (Pellini 2010).

Dentre as técnicas de simulação em tempo real duas podem ser destacadas:

- *Rapid Control Prototyping* (RCP): em um determinado sistema o *hardware* dedicado de controle é substituído por um modelo simulado do controlador executado em um computador de propósito geral. Este tipo de abordagem é utilizado com frequência quando se deseja testar novos algoritmos de controle diretamente na planta física.

- *Hardware-in-the-loop* (HIL): em um determinado sistema o *hardware* dedicado de controle é utilizado no experimento e a planta é substituída por um modelo de simulação em tempo real. Esta abordagem é clássica em testes e análises de comportamento e desempenho de sistemas dedicados de controle.

Entretanto, (Albuquerque 2007) descreve que o termo *Hardware-in-the-Loop* (HIL) também é utilizado quando o mesmo se apresenta de uma forma híbrida, que inclui a simulação de um controlador e/ou uma planta. A Figura 1 apresenta as possíveis combinações de uma simulação HIL, onde em um sistema de testes tanto os atuadores, planta ou sensores podem ser simulados.

É importante ressaltar que nessas aplicações é imprescindível a sintonia e o sincronismo entre o tempo simulado e o tempo real, além do papel das interfaces de comunicação entre o mundo exterior e o simulador (e vice-versa) para o intercâmbio das informações e sinais pertinentes.

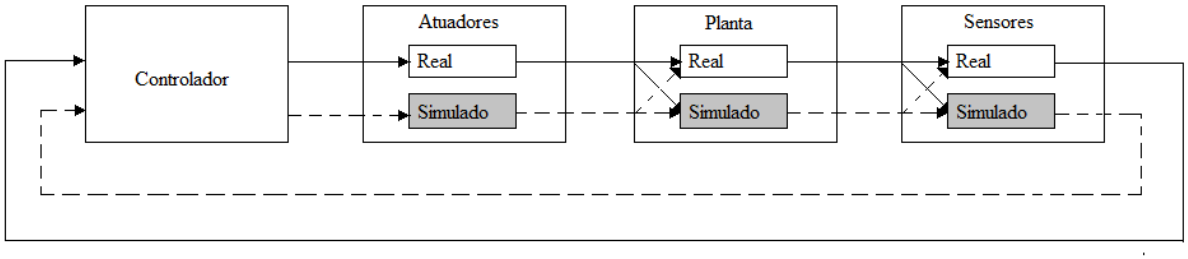


Figura 1. Combinações possíveis de uma aplicação HIL
Fonte: (Albuquerque 2007).

Em (Bouscayrol 2008) o autor descreve que as simulações HIL são encontradas em muitos campos da engenharia: na indústria automotiva, avaliação de unidades de controle, eletrônica de potência, servo acionamentos e robótica, sistemas de tração para trens e metrôs, aplicações educacionais, conversão de energia eólica, entre outros.

A prática das simulações em HIL se iniciou com o uso de computadores de propósito geral, com processadores operando em paralelo. Porém, os simuladores baseados nesta plataforma, mesmo com os processadores atuais, apresentam altas latências de cálculo, o que pode restringir a aplicações que não necessitem de passos de cálculo reduzidos.

Com o advento dos dispositivos FPGA (*Field-Programmable Gate Array*), houve mudanças no cenário dos simuladores. Antes, estes dispositivos eram usados apenas como interface entre equipamentos, como nas placas de conversores A/D, captura de sinais, ou como geradores de função para a geração de sinais senoidais, sinais *resolver*, PWM, etc.

Entretanto, nos dias atuais, são capazes de incorporar modelos matemáticos mais complexos e com baixa latência nos cálculos. O que desperta o interesse em utilizá-los em tarefas que antes eram executadas por um computador de propósito geral.

Apesar de tudo, ainda existem grandes desafios em se utilizar os FPGA's como simuladores de tempo real. O principal deles está relacionado às limitações de *hardware*. Os FPGA, principalmente os de baixo custo, possuem uma quantidade de elementos lógicos e blocos multiplicadores reduzidos. Esses elementos são combinados entre si para representar em *hardware* do modelo da planta a ser emulado.

A aritmética predominante nos dispositivos FPGA é o de ponto fixo. A aritmética de ponto fixo, se mal condicionada, pode inserir erros graves no sistema pela falta de precisão e arredondamentos. Vale a ressalva que, quanto maior o número de bits utilizado nas representações de ponto fixo, maior é o uso de elementos lógicos do FPGA.

Os elementos lógicos, ou blocos lógicos, são as menores unidades lógicas do dispositivo compostos por uma reunião de componentes lógicos, entre eles flip-flop, cuja

combinação de suas entradas e saídas permitem a implementação das funções lógicas, programar registradores, etc.

Os dispositivos FPGA variam em termos de capacidade e preço, quanto maior o número de elementos lógicos e blocos especiais (multiplicadores, memórias, etc) maior o seu valor. Mas, quando comparados aos sistemas comerciais, possuem um custo mais baixo. Neste sentido, são mais atrativos que os simuladores de tempo real comerciais. Entretanto, os simuladores comerciais, como dSPACE® e *RT-Lab®,* possuem um conjunto de ferramentas que tornam o desenvolvimento do projeto mais ágil.

O método de discretização também influencia no número de utilização dos elementos lógicos e multiplicadores. Quanto mais sofisticado o método, geralmente, maior é o número de operações matemáticas, fazendo com que um número maior de blocos multiplicadores e elementos lógicos seja necessário. Uma planta discretizada por um método complexo, utilizando um número de bits elevado para a representação em ponto fixo, poderá exigir uma disponibilidade de recursos maior do que a disponibilidade de um determinado FPGA.

Portanto, encontrar a melhor relação entre o consumo de recursos e a precisão do sistema considerando o método de discretização e a resolução dos bits das variáveis de ponto fixo; e ainda assim manter a fidelidade ao sistema original; é indispensável para garantir a viabilidade de uma aplicação. O modelo deve ser representado satisfatoriamente com um consumo de recursos inferior a capacidade do FPGA.

Simuladores em tempo real são ferramentas importantes para o desenvolvimento de controladores das máquinas elétricas, como a máquina síncrona a imã permanente. Para o controle destas máquinas é necessário o conhecimento da velocidade e posição do rotor. Isto normalmente é feito com a utilização de sensores de posição e de velocidade acoplados ao eixo da máquina. Entretanto, também há o desenvolvimento de técnicas de controle sem a utilização de sensores (*sensorless*) utilizando as grandezas da máquina como sensor de posição.

Através de uma plataforma HIL, podem-se utilizar as técnicas *sensorless* de estimação da posição neste ambiente controlado, sem riscos as pessoas e equipamentos. A partir disto, simular e avaliar efeitos encontrados em um acionamento real, a exemplo, os apontados por (E. M. Fernandes 2011), que trata a necessidade de alterar a sintonia dos controladores, avaliar a relação $\frac{L_d}{L_q}$ da máquina, a influência da quantidade de bits de um conversor A/D, além de estudos sobre os efeitos de tempo morto do inversor de tensão. A partir deste sistema é

possível simular estas dificuldades e outros problemas sob quaisquer condições, a fim de testar novos métodos e minimizar os erros até a aplicação final.

1.1 Objetivos

Este trabalho tem como objetivo elaborar uma plataforma de simulação em tempo real para permitir que os usuários possam desenvolver ou aprimorar soluções de controle diretamente no dispositivo que controlará a máquina real. Esta plataforma, conta com um FPGA simulando o modelo da máquina, o conversor de potência e sensores. O FPGA é conectado com um controlador baseado em DSP através de uma interface que emula um sistema real. Deste modo, avalia-se o desempenho dos algoritmos de controle no tempo de desenvolvimento e em um ambiente seguro.

A fim de atingir o objetivo geral, traçam-se os seguintes objetivos específicos:

- Estudar as características de máquinas síncronas a ímã permanente com imãs montados na superfície do rotor e máquinas síncronas a imã permanente com ímãs inseridos no interior do rotor;
- Implementar modelos matemáticos eficientes e precisos discretizados para um passo de cálculo de 1μs, utilizado no dispositivo FPGA.

1.2 Organização do Trabalho

Neste trabalho foi desenvolvida uma plataforma de simulação em tempo real com *hardware in-the-loop* para máquinas elétricas. Para tanto, foram implementados dois modelos de máquinas síncronas a imã permanente. Uma com um perfil de indutâncias e força contra eletromotriz senoidal, sem saliências, e outra máquina que apresenta saliências significativas, com um perfil de indutâncias e força contra eletromotriz bastante distorcidas.

O trabalho está organizado em 7 capítulos:

- No Capítulo 2 “Revisão Bibliográfica” foi realizada a contextualização do tema abordando o tipo de discretização, passo de cálculo e aritmética de simuladores HIL baseados em FPGA e em máquinas elétricas.

- No Capítulo 3 “Máquina Síncrona a Íma Permanente” são apresentadas as características das duas máquinas, o modelo *dq* utilizado e o sistema típico de acionamento que foi replicado no decorrer do trabalho.
- No Capítulo 4 “Plataforma de Simulação em Tempo Real” é tratado o desenvolvimento da plataforma, onde são relatados os passos da implementação, características e configuração geral do simulador.
- O Capítulo 5 “Implementação da Máquina Síncrona a Imã Permanente SMPM no Simulador HIL” discute a metodologia utilizada para a implementação do modelo matemático de uma máquina síncrona a imã permanente com imãs na superfície do rotor.
- O Capítulo 6 “Implementação da Máquina Síncrona a Imã Permanente IPMS no Simulador HIL” discute a metodologia utilizada para a implementação do modelo matemático de uma máquina síncrona a imã permanente com imãs no interior do rotor.
- O Capítulo 7 apresenta as conclusões e sugestões para trabalhos futuros.

Capítulo 2

Revisão Bibliográfica

Nas últimas duas décadas os computadores de propósito geral à medida que se tornam cada vez mais potentes também se tornam mais acessíveis. Como consequência, novos recursos computacionais surgem e permitem novas formas mais sofisticadas de simulações para a aplicação de sistemas e controles dinâmicos com alta fidelidade e aplicabilidade. Os sistemas de simulação de tempo real permitem a implementação de algoritmos avançados para o acionamento e controle de motores e avaliar a sua performance em tempo real (Bouscayrol 2008).

Existe uma ampla gama de produtos e tecnologias de simuladores comerciais disponíveis para os mais diversos setores: sistemas de potência, tração, veículos híbridos, eletrônicos para a indústria e o lar, automotivos, sistemas navais e aeroespaciais (Menghal e Jaya Laxmi 2010). Os sistemas geralmente utilizam computadores de propósito geral, combinados com módulos equipados com dispositivos como: interfaces digitais e analógicas para a conexão entre os sistemas simulado e real; assim como microprocessadores, DSP e FPGA. Na literatura são encontrados diversos trabalhos que utilizam as plataformas comerciais, como o *RT-Lab* (Abourida, Bélanger e Dufour 2005), (Dufour, Dumur, et al. 2008), (Bachir e David 2010); e *dSPACE* (Song, Weituo e Luo 2008), (Bouscayrol 2008), (Gebregergis e Pillay 2010), (Lucía, et al. 2010), (Lucía, et al. 2011).

A principal vantagem dos sistemas comerciais é a facilidade de implementação do sistema simulado, que pode ser desenvolvido através de diagramas de blocos intuitivos, com geração automática dos códigos dos modelos matemáticos utilizados através de ferramentas como o MATLAB/SIMULINK, o que resulta na agilidade de implementação e testes (Menghal and Jaya Laxmi 2010). Entretanto, (Abourida, Bélanger and Dufour 2005) aponta uma dificuldade encontrada nas plataformas HIL comerciais como o *RT-Lab* e *dSPACE* da

época. Mesmo com os processadores de última geração o menor passo de cálculo destas plataformas dificilmente era inferior do que 25µs, devido as latências entre sistemas de comunicação, entradas e saídas, e também a resolução dos modelos. Essa característica dava ênfase ao seu trabalho onde a simulação operava com um passo de cálculo de 10 µs.

A simulação de tempo real de sistemas de potência e sistemas eletromecânicos necessitam, para serem precisos e realistas, um passo de cálculo na ordem de microssegundos e sub-microssegundos. Em (Bachir e David 2010) destaca que modernos processadores de propósito geral têm capacidades de atingir passos de cálculo de nada menos que 2µs e 20µs, utilizados na simulação de sistemas com tamanho moderado, o que os limita a uma estreita faixa de aplicações.

A popularização dos sistemas HIL vem acompanhada com o aumento de sua complexidade fazendo com que ocorra a substituição das simulações tradicionais baseadas em *softwares* específicos. Para suprir essa necessidade surge o conceito de empregar simulação em tempo real *Hardware-in-the-Loop* baseado em FPGA (Washington e Doman 2010). A utilização dos sistemas de testes em HIL com FPGA possui uma velocidade de cálculo superior, na ordem de nano segundos, permitindo a simulação realista de componentes de uma planta, onde não se poderia alcançar no modo tradicional.

O FPGA pode ser definido como um dispositivo lógico que contém um array bidimensional de células lógicas genéricas, formado por blocos de entrada e saída, e conectado por fios de interconexão, onde todos esses itens são configuráveis. A computação reconfigurável é uma solução intermediária na resolução de problemas complexos, possibilitando combinar a velocidade do *hardware* com a flexibilidade do *software* (Ferreira 2011).

Uma comparação entre um sistema baseado em PC e um baseado em FPGA (*System on Chip*) é apresentada em (Ma e Kennel 2012). Neste trabalho, é demonstrada uma estratégia de controle *sensorless* de uma PMSM baseada em injeção de sinal de alta frequência. As estratégias de controle foram implementadas em um PC com vários cartões de interface (conversores A/D de 12bits, gerador PWM, etc) e em uma plataforma FPGA com as respectivas interfaces. Cada sistema foi testado aplicando-o a um motor real. O resultado dos testes demonstraram que a plataforma baseada em FPGA apresenta melhor performance com menores ondulações de corrente e posição.

Os desafios para adotar amplamente a tecnologia de FPGA para simulações em tempo real ainda são muitos. O predomínio do formato em ponto fixo é uma das maiores preocupações. A aritmética de ponto fixo implica em um trabalho de pré-processamento

tedioso nas tarefas de modelagem matemática, desta forma, técnicas que utilizam aritmética em ponto flutuante estão sendo cada vez mais estudadas.

Os dois tipos de representação numérica possuem vantagens e desvantagens. As características de cada tipo de aritmética são bem documentadas na literatura e podem ser resumidas como:

1) O formato em ponto fixo é uma representação de um conjunto de valores eqüidistantes representados por um número fixo de dígitos antes e depois do ponto decimal. Os FPGAs normalmente possuem operadores aritméticos básicos implementados em hardware com um clock de alta freqüência e latências baixas.

2) O formato em ponto flutuante é uma aritmética digital que traz praticidade e está padronizada pela IEEE-754. Um número real x é representado pelo trio (s,e,m), assim $x = (-1)^s . 2^e . m$, onde *"s"* é o bit de sinal, *"e"* o expoente e *"m"* a mantissa. O formato admite precisão simples (8 bits para *"e"*, 24 bits para *"m"*) e dupla precisão (11 bits para *"e"*, 53 bits para *"m"*). A vantagem do ponto flutuante é que os modelos propostos neste formato possuem a correspondência de 1 para 1 de todas as quantidades físicas de um modelo. Entretanto, como desvantagem em relação ao ponto fixo, por serem mais complexos e com formato de 32 ou 64 bits, possuem uma latência maior nos cálculos, e consomem mais recursos de um FPGA para realizar uma mesma operação aritmética.

Em (Bachir e David 2010) apresentam um comparativo de como o passo de cálculo é afetado devido o uso de ponto fixo e ponto flutuante. Foi constatado que o passo de cálculo com trabalhos utilizando a aritmética de ponto fixo alcançam passos de cálculo menores. Sendo assim, uma das maiores preocupações quanto ao uso do ponto flutuante está na latência, a utilização de operações em ponto flutuante influencia diretamente no passo de cálculo.

Um caso que comprova a eficácia do ponto fixo é encontrado em (Matar e Iravani, FPGA Implementation of the Power Electronic Converter Model for Real-Time Simulation of Electromagnetic Transients 2010) na simulação baseada em FPGA de transientes eletromagnéticos de um conversor eletrônico de potencia. A exigência de trabalhar com um passo de cálculo de 500ns e o consumo dos recursos de hardware nas operações em ponto flutuante fez com que este tipo de operação fosse descartado. Para isto então o código foi baseado em ponto fixo, com uma escolha apropriada do número de bits para que os cálculos

tivessem a precisão desejada. Neste caso, o formato escolhido para a representação do sistema de equações em ponto fixo foi de 20 bits para a parte fracionária (Q-20) e as variáveis foram dimensionadas para 35 bits com sinal: 14 para a parte inteira e 20 para a parte fracionária. Com a implementação em ponto fixo foi possível trabalhar em tempo real, com passo de cálculo menor que os 500ns requeridos para este tipo de aplicação.

No trabalho de (Duman, Can e Akin 2007) foi realizada a simulação de uma máquina de indução em tempo real. Os autores optaram por trabalhar em ponto flutuante. Porém, para alcançar os requisitos de passo de cálculo, foi adotado o formato de ponto flutuante com precisão simples ao invés da precisão dupla.

Como observado, a escolha entre a aritmética de ponto fixo e a aritmética de ponto flutuante depende da aplicação a que se destina. Nesse sentido, há interesse no estudo e avaliação do comportamento de ambas as aritméticas, objetivando atingir melhores índices de precisão nas simulações em tempo real, bem como, passos de cálculo de curta duração, que atendam aos requisitos de uma simulação em tempo real.

Outra dificuldade inerente ao uso de FPGA esta no método de discretização. Quanto mais sofisticado o método, normalmente, maior é o numero de operações matemáticas. Com um grande número de operações matemáticas, sendo executadas em um *software* um processador genérico, leva mais tempo para resolver o problema. Entretanto, no caso do FPGA, um número maior de equações ocasiona no aumento do consumo de recursos, são necessários números maiores de elementos lógicos e blocos multiplicadores para realizar as tarefas. Como os recursos do FPGA são limitados, a escolha de um método mais complexo de discretização pode extrapolar a disponibilidade de recursos. Por este motivo, na literatura, são encontrados basicamente três métodos de integração: Euler, *Backward* Euler e Tustin. Métodos de integração baseados em Euler são encontrados (Dufour, Abourida, et al. 2006), (Charaabi, Monmasson e Slama-Belkhodja 2008), (Bachir e David 2010) e (Darba, et al. 2012). Métodos baseados em *Backward* Euler são descritos em em (Le-Huy, et al. 2006) e (Matar e Iravani, FPGA Implementation of the Power Electronic Converter Model for Real-Time Simulation of Electromagnetic Transients 2010) Métodos de integração baseados em Tustin são descritos em (Matar, Abdel-Rahman e Soliman, FPGA - Based Real-Time Digital Simulation 2005), (Bahri, et al. 2008) e (Myaing e Dinavahi 2011).

No trabalho de (Bahri, et al. 2008) é realizada uma implementação de um simulador de tempo real para um sistema elétrico típico, com a simulação de um inversor de tensão aplicado a uma carga do tipo *RLE*. Para discretizar o modelo da carga, o trabalho considerou a utilização dos métodos de *backward* Euler e de Tustin, apresentando um comparativo entre os

dois. Pode-se verificar um erro maior no método de *backward* Euler para passos de integração maiores. Contudo, foi observado que o erro é minimizado para passos de integração menores. No caso de 2.5µs, apresentaram resultados precisos. Entretanto, verificaram que, nos dois casos, quanto menor o período de amostragem maior deve ser o comprimento de palavra das variáveis em ponto fixo, o que pode ser um problema nos ambientes em que se trabalhe com tamanhos de palavra relativamente curtos.

A implementação de um projeto em um FPGA é feito a partir de um arquivo descritivo que deve ser criado a partir de uma linguagem de descrição de hardware (do inglês, *Hardware Description Language- HDL*) como o *Verilog* ou o VHDL. As ferramentas de desenvolvimento são disponibilizadas pelos fabricantes, um exemplo é o *Quartus II* da Altera. Para facilitar o trabalho, são disponibilizadas também, ferramentas para criação do projeto de forma gráfica, que libera o desenvolvedor do uso de código VHDL como mostrado em (Wang, Tran e Andrade 2010), que utiliza recursos do LabVIEW FPGA para configuração. Assim como em (Darba, et al. 2012) que utilizaram um FPGA da marca Xilinx® que disponibiliza uma ferramenta para o MATLAB/SIMULINK chamada *Xilinx System Generator* (XSG), sem a necessidade de códigos mais complexos como o VHDL. Exemplos de utilização de HDL podem ser encontrados no trabalho de (Le-Huy, et al. 2006), e o uso de interfaces gráficas no trabalho de (Dufour, Abourida, et al. 2006).

De acordo com (Bachir e David 2010) o uso de FPGA em simulações de tempo real podem servir para os seguintes objetivos:

1) Na interface de placas ADC/DAC para interagir com o hardware a ser testado;
2) Para serem utilizados como geradores de função para a geração de sinais senoidais, sinais *resolver*, PWM;
3) Embarcar modelos matemáticos que alcançam alta fidelidade na simulação.

Em (Washington and Doman 2010) é demonstrado como os FPGA podem ser utilizados nas simulações em tempo real na área de interface e sinais. No mesmo trabalho, é discutido como é vantajoso utilizar o FPGA para simular sensores como termoacopladores, *Linear Variable Differencial Transformers* e resolvers.

2.1 Sistemas de Tempo Real Aplicados a Máquinas Elétricas

O uso de sistemas de tempo real aplicado a máquinas elétricas vem se difundindo nos últimos anos devido aos benefícios: reduz o tempo de estudos, utilizam componentes reais em um ambiente controlado, melhorando o sistema até o propósito final. A sua aplicação pode ser

encontrada nas variantes destes sistemas: planta simulada e controlador real; ou controlador simulado e planta real. Referências encontradas na literatura serão comentadas nesta seção.

Um dos primeiros trabalhos envolvendo máquinas síncronas a ímã permanente e *hardware-in-the-Loop* pode ser encontrado em (Abourida, Bélanger e Dufour 2005) que utiliza o sistema comercial RT-Lab como plataforma de simulação. Neste caso, a planta simulada contempla o acionamento completo da PMSM, com inversor trifásico a IGBT, barramento CC a capacitor e ponte trifásica de diodos, tudo com passo de cálculo de 10µs. O objetivo do sistema é validar e otimizar o controlador. Entretanto, como os sinais PWM operam em alta frequência, na ordem de kHz, eles não poderiam ser amostrados de forma precisa pelo simulador a base de processadores em paralelo pela falta de resolução. Portanto, foi utilizado uma placa FPGA com um *clock* de 100MHz para amostrar os sinais PWM com uma resolução de 10ns, a placa captura as bordas de subida e descida e a informação de tempo realimenta o modelo do inversor, que é baseada em uma interpolação do tempo dos pulsos aplicados no inversor de tensão e gerador de PWM. Com essas características foi possível avaliar os efeitos de tempo morto que é menor do que o passo de cálculo da simulação. No trabalho são mostrados ainda os resultados de simulação que comprovam a proposta apresentada.

No trabalho proposto por (Dufour, Abourida, et al. 2006) é apresentada uma nova estrutura de simulação HIL. O modelo da máquina a imã permanente foi discretizada com base no método de Euler e aritmética de ponto fixo. As equações foram embarcadas em um chip FPGA Xilinx ao invés de se utilizar um computador de propósito geral. O sistema foi implementado na plataforma RT-Lab através de diagramas gráficos do simulink utilizando o bloco *Xilinx System Generator* (XSG). Foi mostrada a simplicidade de se usar o bloco XSG, mas em contrapartida, os autores detectaram problemas devido ao dimensionamento em ponto fixo. O bloco não fazia o auto dimensionamento de ponto fixo como o *blockset* Fixed-Point do Simulink®, a representação utilizada foi com Q15. A solução encontrada foi fazer o dimensionamento de ponto fixo manualmente baseado nos resultados do bloco Fixed-Point do Simulink®. A latência total da simulação é de 1,31µs. Os resultados da compilação e síntese lógica foram apresentados e indicam uma latência de 310ns para o FPGA calcular o modelo matemático da máquina. Atrasos nos conversores A/D somam 1µs. Entretanto, são apresentados resultados com controle em malha aberta considerando um passo de cálculo da máquina de 10ns, o que demonstra certa incoerência técnica considerando os requisitos de tempo. Resultados em malha fechada foram obtidos com um controlador de velocidade sendo executado em um microcomputador de propósito geral. Outros trabalhos com esta estrutura

estão em (Dufour, Bélanger, et al. 2007) que utilizam técnicas de elementos finitos para a resolução das equações. Já em (Dufour, Lapointe, et al. 2008) os autores abrangem a simulação, e além de simular a máquina PMSM também utilizam da arquitetura RCP para desenvolvimento de um controlador convencional de corrente.

Um trabalho envolvendo FPGA e máquina de indução é apresentado em (Duman, Can e Akin 2007). O equipamento utilizado para embarcar o modelo matemático do motor é um cartão FPGA PCI (*peripheral component InterConnect*) da Altera®. O modelo trifásico do motor de indução é definido na representação no espaço de estados de sistemas de equações diferenciais, com sistema numérico em ponto flutuante de precisão simples. O objeto principal do trabalho é apresentar a técnica de utilização de ponto flutuante com baixas latências na utilização HIL com máquinas de indução. Segundo os autores, foram alcançados passos de cálculo menores que 1µs, entretanto, os resultados de simulação não foram apresentados. O sistema implementado foi desenvolvido pela linguagem de *hardware* VHDL através da ferramenta Quartus-II.

Outro trabalho envolvendo simulação de tempo real de uma máquina de indução é encontrado em (Charaabi, Monmasson e Slama-Belkhodja 2008). O sistema consiste de um ambiente de simulação totalmente emulado em um FPGA. O trabalho oferece uma metodologia para a obtenção do tamanho das palavras em ponto fixo considerando uma relação com o passo de cálculo. Um modelo resultante do motor de indução em ponto fixo é testado com algoritmos de controle baseados em DTC (*Direct Torque Control*). Resultados de simulação são demonstrados com o passo de controle em 100µs, entretanto, o passo de cálculo da máquina não foi comentado. A Figura 2 apresenta a estrutura da simulação embarcada no FPGA. Percebe-se que através de um módulo de interface serial, as variáveis de interesse são enviadas a uma interface homem-máquina em um computador padrão para a obtenção dos gráficos.

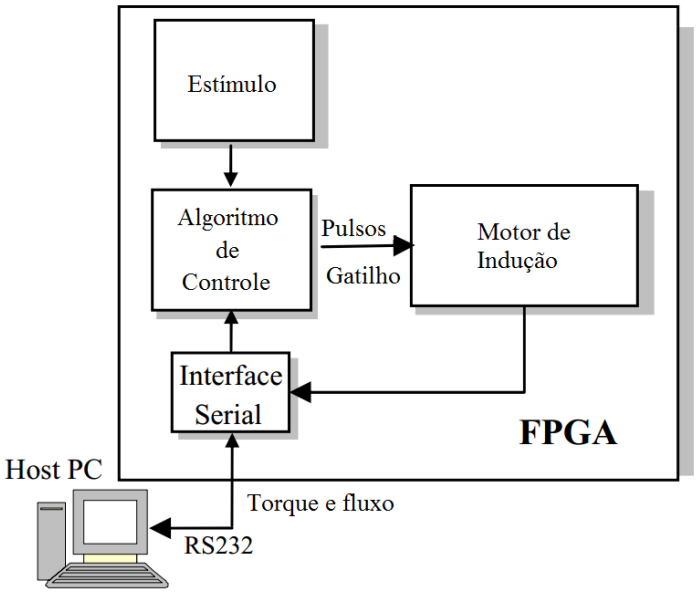


Figura 2. Estrutura de testes de uma simulação em tempo-real.
Fonte: (Charaabi, Monmasson e Slama-Belkhodja 2008)

Em (Bachir e David 2010) é encontrado um sistema HiL baseado em FPGA utilizando o sistema comercial RT-Lab. O objetivo principal do trabalho é o desenvolvimento do ambiente utilizando aritmética de ponto flutuante com modelagem no espaço de estados respeitando os requisitos de tempo e consumo de recursos do FPGA Vrtex 5 da Xilinx. São discutidas técnicas de implementação de acumuladores em ponto flutuante e por fim a simulação de um motor síncrono a imã permanente com força contra-eletromotriz trapezoidal. Para poupar recursos, neste sistema o perfil da força contra-eletromotriz foi implemento diretamente em uma tabela (*look-up table*) no FPGA. Durante a simulação cada valor de fase da força contra eletromotriz é obtida através da multiplicação de um valor da tabela com a velocidade elétrica. Este método permite que o perfil da força contra-eletromotriz seja ajustada pelo usuário para incluir, por exemplo, harmônicos. O passo de cálculo alcançado com o método atingiu 160ns, entretanto, o autor aponta que a velocidade angular apresentou problemas devido ao arredondamento da aritmética de ponto flutuante. O problema foi solucionado desacoplando o modelo mecânico do elétrico, e o executando o modelo mecânico com passo de cálculo em 1µs.

Nos trabalhos de (Colli, Stefano, et al. 2007) e (Colli, Stefano e Marignetti 2010) os autores apresentam uma plataforma HIL para o desenvolvimento de um controlador baseado em FPGA que utiliza técnicas para a estimação da posição do rotor de uma máquina síncrona a imã permanente. As estratégias são baseadas em observadores de fluxo da força contra-

eletromotriz e técnicas de injeção de sinais de alta freqüência para a estimação da posição e/ou velocidade do rotor da máquina.

Tabela 1. Uso do dispositivo FPGA embarcar um sistema de controle.

Descrição	Quantidade
Elementos lógicos	22852 (47%)
Blocos de Memória	265884 (10%)
Blocos DSP (multiplicadores 9 bits)	56 (19%)

Fonte: (Colli, Stefano e Marignetti 2010).

O controlador é embarcado em uma placa FPGA Altera Startix II e a planta é simulada em um computador de propósito geral através do Simulink, a comunicação entre as partes é feita através da interface JTAG. Entretanto, essa plataforma possui uma latência muito alta e na sua configuração padrão não é executada em tempo real. Por isso os autores utilizaram a ferramenta da Altera DSP Builder que possui duas técnicas de comunicação JTAG que tem como objetivo aumentar a velocidade da simulação HIL rodando em malha aberta, baseada em um método chamado *off-line-recorder stimuli*. Entretanto, maiores detalhes desta operação não foram apresentados. O consumo dos recursos do FPGA para utilizá-lo como controlador está exposto na Tabela 1.

No trabalho de (Myaing e Dinavahi 2011) os autores apresentam um simulador de tempo real que tem como principal característica a emulação do sistema de potência com características detalhadas das chaves do conversor. É descrito um modelo do conversor fonte de tensão trifásica que utiliza, ao invés de chaves ideais, a representação detalhada de uma chave IGBT. Este modelo possui características realistas, que refletem os efeitos da não linearidades do inversor, rejeição de corrente e comportamento do diodo reverso de um conversor real. O levantamento dos transitórios de tensão e corrente das chaves IGBT foi realizado com uma resolução de 12,5ns, permitindo que o modelo fosse executado com este passo de integração. Efeitos de tempo morto foram incluídos e representam 2µs. O sistema total de simulação no FPGA contempla o inversor trifásico, um motor de indução, um controlador baseado na orientação pelo campo e o PWM. O modelo da máquina foi discretizado pelo método de *Tustin* a um passo de cálculo de 10µs. O FPGA utilizado foi o Stratix EP 1S80 da Altera®. Os recursos utilizados, assim como a porcentagem correspondente do FPGA, estão demonstrados na Tabela 2. Percebe-se que foi necessária a utilização de um dispositivo com boa disponibilidade de recursos para comportar todo o sistema. Os resultados apresentados mostram o comportamento das tensões e correntes no

motor e, através da análise harmônica, é possível verificar que está presente a influência imposta pelo modelo do inversor.

Tabela 2. Recursos utilizados do FPGA.

Componente	Elementos Lógicos	Blocos DSP (multiplicadores de 9bits ou PLL)	Bits de Memória
Inversor trifásico	20699 (26.2%)	40 (22.7%)	26880 (0.4%)
Motor de Indução	21881 (27.7%)	128 (72.7%)	0
Controlador	9676 (12.2%)	8 (4.6%)	0
PWM	250 (0.3%)	0	0
Conexões	350 (0.4%)	0	0
Referencias	118 (0.2%)	0	0
Interface	99 (0.1%)	0	0
Total	53043 (67.1%)	176 (100%)	26880 (0.4%)

Fonte: (Myaing e Dinavahi 2011).

Um trabalho que envolve acionamento *sensorless* de uma PMSM e simulação em tempo real baseada em FPGA é encontrado em (Darba, et al. 2012). O trabalho se baseia em simular em tempo real o acionamento da máquina com velocidade nula. Para a estimação da posição do rotor foi utilizada a técnica de injeção de sinal de corrente de alta-frequência pulsante. O controlador, gerador PWM, fonte de tensão e a máquina foram emulados em uma placa FPGA Xilinx Spartan 3E XC3S500E. Toda a programação foi realizada pelo Simulink utilizando a ferramenta de blocos *Xilinx System Generator* (XSG), sem a utilização de códigos de baixo nível como o VHDL. Para garantir toda a simulação em uma placa FPGA, os autores utilizaram técnicas que combinam o reuso de recursos considerando o paralelismo e a otimização dos módulos denominado *Algorithm Architecture Adequation (A³)*. O passo de cálculo alcançado para o controlador foi de 125µs enquanto que para o modelo da máquina foi de 2,5µs, sendo 50 vezes menor que o período do PWM. Os resultados de simulação mostraram a precisão da plataforma, que demonstrou o erro máximo da posição em 0,5°. A obtensão dos dados também é discutida, e foi realizado através de um software chamado *Chipscope Pro Analyzer*, que realiza as medições dos sinais internos do FPGA e faz um *upload* dessas informações para o computador, tornando possível a visualização das variáveis.

Tabela 3. Comparativo entre propostas baseadas em HIL e FPGA para máquinas elétricas

Artigo	Maquina	Metodo de integracao	Passo de calculo	Aritmetica	Plataforma	Objetivos
(Abourida, Bélanger e Dufour 2005)	PMSM	Não discutido	10us	Não discutido	RT-Lab	Simular um sistema HiL completo.
(Dufour, Abourida, et al. 2006)	PMSM	Euler	1.31us (latência)	Ponto Fixo	RT-Lab FPGA	Simular um sistema HiL com o blockset XSG.
(Charaabi, Monmasson e Slama-Belkhodja 2008)	Indução	Euler	Não Informado	Ponto fixo	FPGA	Estudo do tamanho das palavras em ponto-fixo para representar a máquina. Testes com controle DTC.
(Bachir e David 2010)	BLDC	Euler	160ns – elétrico; 1µs - mecânico	Ponto Flutuante	RT-Lab	Verificar erros com a simulação em ponto flutuante
(Myaing e Dinavahi 2011)	Indução	Tustin	10us	Ponto fixo	FPGA	Simular acionamento com sistema de potencia a IGBT emulado
(Darba, et al. 2012)	PMSM	Euler	2.5us	Ponto fixo	FPGA	Utilizar o HiL para estimar a posição com o motor parado.

A Tabela 3 apresenta um comparativo de alguns trabalhos supracitados, com o foco direcionado para a simulação de máquinas elétricas com HIL e FPGA, apresentando o método de integração passo de cálculo e aritmética utilizada. Tais parâmetros, motivaram as escolhas adotadas no decorrer da implementação neste trabalho, tais como: Euler como método de integração; passo de cálculo de 1µs; aritmética de ponto fixo. Estes foram importantes para desenvolver em FPGA os dois modelos de máquinas síncronas a ímã permanente.

Capítulo 3

Máquina Síncrona a Imã Permanente

A máquina síncrona a imã permanente surge a partir da substituição dos enrolamentos de campo de uma máquina síncrona tradicional, por um conjunto de ímãs permanentes, apropriadamente colados, incrustados ou implantados no rotor.

Com a utilização dos imãs permanentes e a ausência dos enrolamentos de campo evitam-se as perdas no cobre e as demais perdas se concentram basicamente nos enrolamentos do estator. Como consequência dessa substituição, tem-se a redução do peso, aumento da eficiência da máquina e potência de saída (conjugado), maior densidade de fluxo no entreferro melhorando a performance dinâmica, simplificação do projeto construtivo, redução da manutenção quanto ao custo, tempo de paralisação da máquina e periodicidade.

Máquinas a imãs permanentes possuem maior relação de densidade de potência e conjugado por volume, em relação a uma máquina de indução. São uma opção atrativa para a substituição de máquinas de corrente contínua em aplicações de servomecanismos. Além deste tipo de acionamento, as máquinas a ímãs permanentes são bastante utilizadas em acionamentos com operação em velocidade constante e acionamentos com velocidade variável.

As máquinas síncronas a imã permanente podem ser classificadas de acordo com a montagem do imã no interior do rotor, e pode apresentar diversas formas. Neste trabalho, será apresentado o modelo matemático e a implementação em um simulador FPGA (HIL) de dois tipos de máquinas a imã: uma máquina a imã com imãs montados na superfície (*Surface Mounted PM motor* - SMPM), e máquina a imã com imãs inseridos no rotor (*Interior PM motor* - IPM), ilustrados na Figura 3.

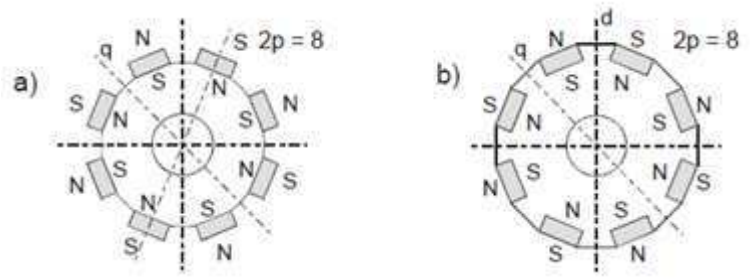


Figura 3. Configurações mais populares: a) ímãs localizados na superfície do rotor, e (b) na interior do rotor

Ambas as configurações possuem construção simples e são robustas. Entretanto, as máquinas PMSM tem a necessidade de utilizar ligas de alta densidade energética, como as ligas magnéticas de terras raras de samário cobalto ou neodímio-ferro-boro. Pelos motivos apresentados em (Costa 2013) o custo destas ligas magnéticas de alta densidade são muito altos. Por esta razão, busca-se o desenvolvimento de outros compostos independentes de terras raras para a construção de imãs permanentes. Os imãs permanentes de cerâmicas (conhecidos também como imãs de ferrite) possuem densidade de fluxo residual, coercitividade, e produto energético inferiores às dos materiais de terras raras, mas têm boas características mecânicas e um custo mais baixo.

Os imãs interiores têm boa resistência mecânica contra forças centrífugas criadas quando o rotor gira à velocidade nominal, os quais permitem o uso de imãs de cerâmica (ferrite) (Costa 2013). Porém, os motores com imãs interiores normalmente apresentam maiores ondulações de conjugado.

Ondulações de conjugado (*Cogging Torque*) (CT) são produzidas por máquinas a imã, especialmente com imãs interiores. O CT é produzido pela interação das ranhuras do estator e as bordas do rotor que provoca relutância magnética a medida que o rotor se movimenta. As ondulações de conjugado são dadas pela Força Magneto Motriz (FMM) produzida pela corrente do estator e ocasionam um conteúdo harmônico no conjugado. Com intuito de minimizar as ondulações de conjugado o campo produzido deve ser senoidal.

As máquinas síncronas a imã permanente podem ser classificadas também quanto a força contra eletromotriz, que pode ter característica da forma de onda puramente senoidal ou não-senoidal.

3.1 MÁQUINA A IMÃ COM IMÃS MONTADOS NA SUPERFÍCIE DO ROTOR (SMPM)

As máquinas síncronas a imã permanente possuem um estator com núcleo de aço laminado e ranhuras uniformes, assim sendo, os enrolamentos das fases são distribuídos uniformemente de forma semelhante às máquinas de indução. Essas características construtivas produzem uma força contra-eletromotriz senoidal de forma parecida àquelas produzidas nas máquinas de indução e nas máquinas síncronas convencionais.

As correntes nas fases da máquina devem estar em sincronismo com a posição do rotor, e deve ser determinada instantaneamente e de forma contínua. Isso faz com que a máquina SMPM exija a utilização de sensores de posição de alta resolução como *resolvers* ou *encoders* absolutos.

O conjugado produzido pela máquina SMPM apresenta oscilações, porém com uma intensidade muito menor que as oscilações observados no conjugado gerado por máquinas de força contra-eletromotriz não senoidal ou com imãs interiores, que será apresentada na sequencia. Por essa característica, este tipo de máquina é destinado a aplicações onde o controle de velocidade e o torque são críticos.

3.1.1 Modelo no Referencial 123

Nesta seção será apresentado o modelo matemático básico da máquina síncrona a imã permanente com distribuição senoidal de força eletromotriz. A máquina é representada basicamente por três enrolamentos trifásicos no estator, com uma armadura semelhante a da máquina assíncrona trifásica, e de um rotor com um imã permanente.

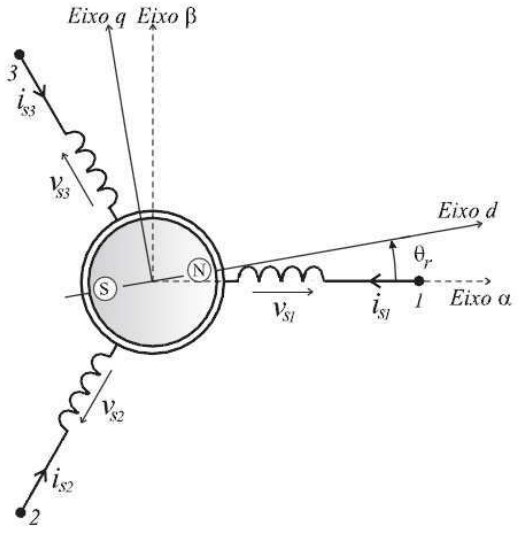


Figura 4 Representação de uma SMPM

O modelo da máquina com imãs montados na superfície do rotor é baseado nas seguintes suposições ou simplificações (E. d. Fernandes 2006):

- Não é considerada a saturação do circuito magnético;
- O material magnético permanente tem uma curva de desmagnetização linear e independente da temperatura;
- O fluxo dos imãs do rotor e o fluxo gerado pelos enrolamentos do estator são senoidais, dessa forma, a força eletromotriz induzida é senoidal;
- As perdas por histerese e as perdas produzidas pelas correntes parasitas são desprezíveis;
- Admite-se que as resistências e indutâncias da máquina são independentes da temperatura e da freqüência;
- O enrolamento trifásico do estator é distribuído de forma senoidal. O enrolamento é conectado em estrela, logo, não existe a componente de seqüência zero da corrente;
- O rotor não tem gaiola-de-esquilo.

Em (E. d. Fernandes 2006) é descrito que a indutância por fase pode ser dividida em duas parcelas: indutância própria e indutância mútua. A parcela correspondente a indutância própria possui os seguintes termos:

$$L_{s1}(\theta_r) = L_{s0} + L_{sm}\cos(2\theta_r) \tag{3.1}$$

$$L_{s2}(\theta_r) = L_{s0} + L_{sm}\cos\left(2\theta_r + {}^{2\pi}/_{3}\right) \tag{3.2}$$

$$L_{s3}(\theta_r) = L_{s0} + L_{sm}\cos\left(2\theta_r - {}^{2\pi}/_{3}\right) \tag{3.3}$$

A indutância mútua tem dois termos, um que varia de acordo com a posição elétrica do rotor (θ_r) e um termo constante:

$$M_{s12}(\theta_r) = M_{s0} + L_{sm}\cos\left(2\theta_r - {}^{2\pi}/_{3}\right) \tag{3.4}$$

$$M_{s23}(\theta_r) = M_{s0} + L_{sm}\cos(2\theta_r) \tag{3.5}$$

$$M_{s13}(\theta_r) = M_{s0} + L_{sm}\cos\left(2\theta_r - {}^{4\pi}/_{3}\right) \tag{3.6}$$

Com isso pode-se expressar a matriz de indutâncias L_{ss} da seguinte forma:

$$L_{ss} = \begin{bmatrix} L_{s1}(\theta_r) & M_{s12}(\theta_r) & M_{s13}(\theta_r) \\ M_{s12}(\theta_r) & L_{s2}(\theta_r) & M_{s23}(\theta_r) \\ M_{s13}(\theta_r) & M_{s23}(\theta_r) & L_{s3(\theta_r)} \end{bmatrix} \tag{3.7}$$

A equação da tensão trifásica da máquina pode ser representada na forma matricial como segue:

$$V_{s123} = R_s i_{s123} + \frac{d\emptyset_{s123}}{dt} \tag{3.8}$$

Onde:

- $V_{s123} = [V_{s1} \quad V_{s2} \quad V_{s3}]^T$ é o vetor das tensões de fase;
- $i_{s123} = [i_{s1} \quad i_{s2} \quad i_{s3}]^T$ é o vetor das correntes de fase;
- $R_s = r_s I_3$, onde I_3 é uma matriz identidade de ordem 3;
- $\emptyset_{s123} = L_{ss} i_{s123} + \emptyset_{r123}$ é o vetor dos fluxos totais das fases;

- $\emptyset_{r123} = \begin{bmatrix} \cos(\theta_r) \\ \cos(\theta_r - 2\pi/3) \\ \cos(\theta_r + 2\pi/3) \end{bmatrix} \emptyset_{pm}$, é a distribuição do fluxo produzido pelo imã permanente do rotor;
- $\emptyset_{pm}$ é o valor de pico do fluxo produzido pelo imã do rotor que enlaça os enrolamentos do estator.

A equação de tensão (3.8) pode ser escrita da seguinte forma:

$$V_{s123} = R_s i_{s123} + L_{ss} \frac{di_{s123}}{dt} + \omega_r \left[\frac{d\, L_{ss}}{d\, \theta_r}\right] i_{s123} + \omega_r \frac{d\emptyset_{r123}}{d\theta_r} \tag{3.9}$$

Onde

- $\omega_r = \frac{d\theta_r}{dt}$ é a velocidade elétrica do rotor em rad. elétricos/s;
- R_s é a resistência dos enrolamentos do estator.

O torque eletromagnético *Te* da máquina pode ser escrito como:

$$T_e = \frac{p}{2} i_{s123}^T \left[\frac{dL_{ss}}{d\theta_r}\right] i_{s123} + p i_{s123}^T \frac{d\emptyset_{r123}}{d\emptyset_r} \tag{3.10}$$

3.1.2 Modelo no Referencial *αβ*

O emprego de métodos de transformação como o *αβ* permite a obtenção de modelos mais simples de máquinas elétricas trifásicas. A utilização da transformação *αβ* converte o modelo trifásico original da máquina para um modelo bifásico equivalente, com mesma potência mecânica, torque, velocidade e número de pólos. A máquina trifásica é representada em um sistema de coordenadas *αβ*.

O sistema bifásico *αβ* é composto por dois eixos ortogonais entre si. Para a obtenção da representação em *αβ* utiliza-se a Transformação de Clarke.

$$X_{s\alpha\beta} = \mathrm{T} X_{s123} \tag{3.11}$$

$$\mathrm{T} = \sqrt{\frac{2}{3}} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \tag{3.12}$$

Aplicando (3.9) em (3.11) e assumindo que a máquina esteja conectada em estrela, as equações que descrevem o comportamento da máquina são reduzidas a:

$$\begin{bmatrix} V_{s\alpha} \\ V_{s\beta} \end{bmatrix} = R_s \begin{bmatrix} i_{s\alpha} \\ i_{s\beta} \end{bmatrix} + L_{s\alpha\beta}(\theta_r) \frac{d}{dt} \begin{bmatrix} i_{s\alpha} \\ i_{s\beta} \end{bmatrix} + \omega_r \left[\frac{dL_{s\alpha\beta}(\theta_r)}{d\theta_r} \right] \begin{bmatrix} i_{s\alpha} \\ i_{s\beta} \end{bmatrix} + \omega_r \frac{d}{d\theta_r} \begin{bmatrix} \emptyset_{r\alpha} \\ \emptyset_{r\beta} \end{bmatrix} \tag{3.13}$$

$$T_t = \frac{P}{2} \begin{bmatrix} i_{s\alpha} \\ i_{s\beta} \end{bmatrix}^T \left[\frac{dL_{s\alpha\beta}(\theta_r)}{d\theta_r} \right] \begin{bmatrix} i_{s\alpha} \\ i_{s\beta} \end{bmatrix} + P \begin{bmatrix} i_{s\alpha} \\ i_{s\beta} \end{bmatrix}^T \frac{d}{dt} \begin{bmatrix} \emptyset_{r\alpha} \\ \emptyset_{r\beta} \end{bmatrix} \tag{3.14}$$

onde:

$$L_{s\alpha\beta}(\theta_r) = \begin{bmatrix} L_{s\alpha\alpha}(\theta_r) & M_{s\alpha\beta}(\theta_r) \\ M_{s\alpha\beta}(\theta_r) & L_{s\beta\beta}(\theta_r) \end{bmatrix}$$

$$\begin{bmatrix} 6L_{s\alpha\alpha}(\theta_r) \\ 2\sqrt{3}M_{s\alpha\beta}(\theta_r) \\ 2L_{s\beta\beta}(\theta_r) \end{bmatrix} = \begin{bmatrix} 4 & -4 & -4 & 1 & 2 & 1 \\ 0 & 2 & -2 & -1 & 0 & 1 \\ 0 & 0 & 0 & 1 & -2 & 1 \end{bmatrix} \begin{bmatrix} L_{s1}(\theta_r) \\ M_{s12}(\theta_r) \\ M_{s13}(\theta_r) \\ L_{s2}(\theta_r) \\ M_{s23}(\theta_r) \\ L_{s3}(\theta_r) \end{bmatrix} \tag{3.15}$$

Neste referencial as indutâncias são expressas da seguinte forma:

$$L_{s\alpha\alpha}(\theta_r) = (L_{so} - M_{so}) + \frac{3}{2} L_{sm} \cos(2\theta_r) \tag{3.16}$$

$$L_{s\beta\beta}(\theta_r) = (L_{so} - M_{so}) - \frac{3}{2} L_{sm} \cos(2\theta_r) \tag{3.17}$$

$$M_{s\alpha\beta}(\theta_r) = \frac{3}{2} L_{sm} \sin(2\theta_r) \tag{3.18}$$

3.1.3 Modelo no Referencial *dq*

Este tipo de representação da máquina SMPM é feita através da transformação de coordenadas do referencial $\alpha\beta$ para o referencial dq. A utilização desta técnica é mais

adequada para a análise da máquina, visto que, as tensões, correntes e fluxos são representadas em um referencial síncrono girante acoplado ao rotor da máquina, e o sistema de coordenadas gira em sincronismo com a freqüência fundamental. A transformação é implementada da seguinte forma:

$$X_{sdq} = \mathrm{Q} X_{s\alpha\beta} \tag{3.19}$$

$$\mathrm{Q} = \begin{bmatrix} \cos\theta_r & \sin\theta_r \\ -\sin\theta_r & \cos\theta_r \end{bmatrix} \tag{3.20}$$

Aplicando a transformação de coordenadas, equação 3.20, nas equações de tensão (2.13), obtêm-se as equações de tensão da máquina no referencial *dq* apresentadas abaixo:

$$\begin{bmatrix} V_{sd} \\ V_{sq} \end{bmatrix} = \begin{bmatrix} r_s & -\omega_r L_{sq} \\ \omega_r L_{sd} & r_s \end{bmatrix} \begin{bmatrix} I_{sd} \\ I_{sq} \end{bmatrix} + \begin{bmatrix} L_{sd} & 0 \\ 0 & L_{sq} \end{bmatrix} \frac{d}{dt} \begin{bmatrix} I_{sd} \\ I_{sq} \end{bmatrix} + \omega_r \begin{bmatrix} 0 \\ \lambda_{pm} \end{bmatrix} \tag{3.21}$$

- L_{sd} é a indutância de eixo direto (*d*);
- L_{sq} é a indutância de eixo em quadratura (*q*);
- $\lambda_{pm} = \sqrt{\frac{3}{2}}\phi_{pm.}$

As indutâncias *d* e *q* são obtidas pelas transformações entre os referenciais e expressas como:

$$L_{sd} = (L_{so} - M_{so}) + \frac{3}{2} L_{sm} \tag{3.22}$$

$$L_{sq} = (L_{so} - M_{so}) - \frac{3}{2} L_{sm} \tag{3.23}$$

Em máquinas de pólos lisos o valor de L_{sd} é igual a L_{sq}, enquanto que para pólos salientes L_{sd} e L_{sq} são diferentes ($L_{sd} > L_{sq}$).

Utilizando a matriz de transformação (3.20) na equação (3.14), pode-se obter o torque total desenvolvido pela máquina:

$$T_t = P\left[\lambda_{pm} I_{sq} + \left(L_{sd} - L_{sq}\right) I_{sd} I_{sq}\right] \tag{3.24}$$

A equação (3.24) mostra que há duas parcelas de torque na máquina. A primeira é diretamente proporcional a I_{sq}, que é o torque eletromagnético. A segunda parcela é o torque de relutância que é proporcional ao produto de ($I_{sd}I_{sq}$) e a diferença entre as indutâncias $\left(L_{sd} - L_{sq}\right)$.

3.1.4 Equação mecânica de movimento

O comportamento dinâmico da máquina representado pela equação mecânica de movimento é:

$$(J + J_m)\frac{d\omega_m}{dt} = T_e - T_c - f_\omega \omega_m \tag{3.25}$$

onde:

- ω_m é a freqüência angular mecânica do rotor;
- *J* é o momento de inércia do rotor da máquina;
- *Jm* é o momento de inércia da carga;
- f_ω é o coeficiente de atrito viscoso da máquina;
- T_c é o torque de carga;
- T_e é o torque total desenvolvido pela máquina.

A relação entre a freqüência angular do sistema de eixos *dq* (ω_r) com a freqüência angular do rotor (ω_m) é dada por:

$$\omega_r = p\omega_m \tag{3.26}$$

3.2 MÁQUINA A IMÃ COM IMÃS INSERIDOS NO INTERIOR DO ROTOR (IPM)

Nesta seção será descrito o modelo de uma máquina a imã permanente de fcem não-senoidal quando comparada a máquina com imãs montados na superfície do rotor (SMPM), descrita na seção anterior. Essa característica faz com que a forma de onda da fcem apresente componentes harmônicas. Trata-se de um protótipo baseado em imãs interiores cuja a liga magnética para a fabricação dos imãs é de ferrite estrôncio dispostos transversalmente no

interior do rotor. O objetivo do projeto foi de melhorar a qualidade da forma de onda do fluxo magnético no entreferro. A máquina apresentada nesta seção corresponde a máquina estudada por (Costa 2013). Nesta seção, serão apresentados o modelo matemático da máquina IPM e suas principais características.

3.2.1 Modelo no Referencial 123

Conforme apresentado em (Costa 2013), a máquina estudada obedece algumas convenções e hipóteses como:

- O motor é considerado equilibrado;
- São consideradas três fases no estator defasadas em 120° ou $\frac{2\pi}{3}$ radianos elétricos e um imã permanente no interior do rotor (não é necessário excitação);
- Utilização de convenção passiva para o circuito;
- As indutâncias próprias e mútuas são dependentes da posição do rotor (θ_m);
- Máquina multipolar, ou seja, posição elétrica do rotor (θ_r) é múltiplo da posição mecânica do rotor (θ_m);
- A distribuição da densidade de fluxo magnético do imã possui harmônicos;
- Co-energia igual a energia (máquina não-saturada);
- O fluxo total é igual aos fluxos parciais;
- O conjugado eletromagnético é igual a variação da energia em relação a posição do rotor.

As características de não linearidades da máquina podem ser percebidas a partir do perfil das indutâncias, que são distorcidas devido a presença de harmônicos. As indutâncias dependem da posição do rotor (θ_r), as próprias são descritas como:

$$L_{s1}(\theta_r) = L_0 - \sum_{n=1}^{2} L_n \cos(2n\theta_r) + \sum_{n=3}^{4} L_n \cos(2n\theta_r) \tag{3.27}$$

$$L_{s2}(\theta_r) = L_0 - \sum_{n=1}^{2} L_n \cos(2n\left(\theta_r - \frac{2\pi}{3}\right)) + \sum_{n=3}^{4} L_n \cos(2n\left(\theta_r - \frac{2\pi}{3}\right)) \tag{3.28}$$

$$L_{s3}(\theta_r) = L_0 - \sum_{n=1}^{2} L_n \cos(2n\left(\theta_r + \frac{2\pi}{3}\right)) + \sum_{n=3}^{4} L_n \cos(2n\left(\theta_r + \frac{2\pi}{3}\right)) \quad (3.29)$$

e as indutâncias mútuas são:

$$M_{s12} = -M_0 - \sum_{n=1}^{4} M_n \cos(2n\left(\theta_r - \frac{\pi}{3}\right)) \quad (3.30)$$

$$M_{s23} = -M_0 - \sum_{n=1}^{4} M_n \cos(2n\theta_r) \quad (3.31)$$

$$M_{s13} = -M_0 - \sum_{n=1}^{4} M_n \cos(2n\left(\theta_r + \frac{\pi}{3}\right)) \quad (3.32)$$

nas quais L_0 e M_0 são amplitudes da componente contínua e L_n e M_n são amplitudes para as componentes alternadas, para n = 1, 2, 3, 4.

A partir destes dados é possível determinar a matriz de indutância L_{ss} da equação (3.7) para a IPM.

As expressões para o fluxo magnético do imã permanente é dado:

$$\phi_{r_{123}} = \begin{bmatrix} \sum_{n=0}^{6} k_{2n+1} \cos((2n+1)\theta_r) \\ \sum_{n=0}^{6} k_{2n+1} \cos((2n+1)(\theta_r - {}^{2\pi}/_{3})) \\ \sum_{n=0}^{6} k_{2n+1} \cos((2n+1)(\theta_r + {}^{2\pi}/_{3})) \end{bmatrix} \phi_{pm} \quad (3.33)$$

Nas quais ϕ_{pm} é a amplitude da fundamental do fluxo magnético do imã permanente, e k_{2n+1} são constantes das harmônicas do fluxo. A equação disposta desta forma representa a inclusão das componentes harmônicas na distribuição do fluxo do imã permanente do IPM.

Em termos práticos, as principais diferenças na representação trifásica da máquina IPM e a SMPM se encontram nas equações das indutâncias própria e mútua do estator bem como do fluxo magnético. As equações gerais trifásicas apresentadas para a SMPM se aplicam para a IPM. Desta forma, assim como na SMPM, a equação de tensão trifásica é representada pela equação (3.9), o torque eletromagnético por (3.10) e as equações mecânicas por (3.25) e (3.26).

3.2.2 Modelo no Referencial $\alpha\beta$

O modelo no referencial αβ é obtido do mesmo modo que o apresentado na seção 3.1.1 para a SMPM, sendo a equação de tensão representada por (3.13) e o torque eletromagnético pela equação (3.14). Devido a nova configuração, os termos da matriz de indutâncias $L_{s\alpha\beta}(\theta_r)$ (3.15) ficam como segue:

$$L_{s\alpha\alpha}(\theta_r) = L_0 + M_0 - \left(\frac{L_1}{2} + M_1\right)\cos 2\theta_r - \left(\frac{L_2}{2} + M_2\right)\cos 4\theta_r + (L_3 + M_3)\cos 6\theta_r + \left(\frac{L_4}{2} + M_4\right)\cos 8\theta_r \tag{3.34}$$

$$M_{s\alpha\beta}(\theta_r) = -\left(\frac{L_1}{2} + M_1\right)\sin 2\theta_r - \left(\frac{L_2}{2} + M_2\right)\sin 4\theta_r - \left(\frac{L_4}{2} + M_4\right)\sin 8\theta_r \tag{3.35}$$

$$L_{s\beta\beta}(\theta_r) = L_0 + M_0 + \left(\frac{L_1}{2} + M_1\right)\cos 2\theta_r + \left(\frac{L_2}{2} + M_2\right)\cos 4\theta_r + (L_3 + M_3)\cos 6\theta_r - \left(\frac{L_4}{2} + M_4\right)\cos 8\theta_r \tag{3.36}$$

sendo o termo $\frac{dL_{s\alpha\beta}(\theta_r)}{d\theta_r} = L'_{s\alpha\beta}(\theta_r)$ das equações (3.13) e (3.14), tem-se:

$$L'_{s\alpha\alpha}(\theta_r) = (L_1 + 2M_1)\sin 2\theta_r + (2L_2 + 4M_2)\sin 4\theta_r - (6L_3 + 6M_3)\sin 6\theta_r - (4L_4 + 8M_4)\sin 8\theta_r \tag{3.37}$$

$$M'_{s\alpha\beta}(\theta_r) = -(L_1 + 2M_1)\cos 2\theta_r - (2L_2 + 4M_2)\cos 4\theta_r - (4L_4 + 8M_4)\cos 8\theta_r \tag{3.38}$$

$$L'_{s\beta\beta}(\theta_r) = -(L_1 + 2M_1)\sin 2\theta_r - (2L_2 + 4M_2)\sin 4\theta_r - (6L_3 + 6M_3)\sin 6\theta_r + (4L_4 + 8M_4)\sin 8\theta_r \tag{3.39}$$

A transformação de Clarke do fluxo do ímã permanente é:

$$\begin{gathered}\phi_{r\alpha\beta}(\theta_r) = \mathrm{T}\phi_{r123}(\theta_r) \\ \phi_{r\alpha\beta}(\theta_r) = [\phi_{r\alpha}(\theta_r) \quad \phi_{r\beta}(\theta_r)]^t\end{gathered} \tag{3.40}$$

na qual $\emptyset_{r\alpha\beta}(\theta_r)$ é a matriz do fluxo do ímã permanente nas coordenadas αβ. Aplicando a transformação de coordenadas da equação (3.40):

$$\phi_{r\alpha\beta}(\theta_r) = \begin{bmatrix} \lambda_{PM}(k_1 \cos\theta_r + k_5 \cos 5\theta_r + k_7 \cos 7\theta_r + k_{11} \cos 11\theta_r + k_{13} \cos 13\theta_r) \\ \lambda_{PM}(k_1 \sin\theta_r - k_5 \sin 5\theta_r + k_7 \sin 7\theta_r - k_{11} \sin 11\theta_r + k_{13} \sin 13\theta_r) \end{bmatrix} \quad (3.41)$$

na qual $\lambda_{PM} = \sqrt{\frac{2}{3}}\phi_{pm}$.

Sendo $\frac{d}{d\theta_r}\left[\frac{\phi_{r\alpha}}{\phi_{r\beta}}\right] = \frac{d\phi_{r\alpha\beta}(\theta_r)}{d\theta_r} = \phi'_{r\alpha\beta}(\theta_r)$ o termo constante nas equações (3.13) e (3.14), sua derivada neste referencial:

$$\phi'_{r\alpha\beta}(\theta_r) = \begin{bmatrix} -\lambda_{PM}(k_1 \sin\theta_r + k_5 \sin 5\theta_r + k_7 \sin 7\theta_r + k_{11} \sin 11\theta_r + k_{13} \sin 13\theta_r) \\ \lambda_{PM}(k_1 \cos\theta_r - k_5 \cos 5\theta_r + k_7 \cos 7\theta_r - k_{11} \cos 11\theta_r + k_{13} \cos 13\theta_r) \end{bmatrix} \quad (3.42)$$

3.2.3 Modelo no Referencial *dq*

O modelo neste referencial segue os mesmos passos apresentados na seção 3.1.2, onde para a obtenção do equacionamento em *dq* as equações são multiplicadas pela matriz de transformação expressa em (3.20).

$$V_{dq} = QV_{\alpha\beta}, I_{dq} = QI_{\alpha\beta} \quad (3.43)$$

$$V_{dq} = [v_d \quad v_q]^t, I_{dq} = [i_d \quad i_q]^t$$

nas quais V_{dq} e I_{dq} são matrizes de tensões e correntes nas coordenadas *dq*.

O fluxo do imã pode ser representado por:

$$\phi'_{rdq}(\theta_r) = \begin{bmatrix} \lambda'_{r_d}(\theta_r) \\ \lambda'_{r_q}(\theta_r) \end{bmatrix} = \begin{bmatrix} \lambda_{da} \sin 6\theta_r + \lambda_{db} \sin 12\theta_r \\ \lambda_{PM} + \lambda_{qa} \cos 6\theta_r - \lambda_{qb} \cos 12\theta_r \end{bmatrix} \quad (3.44)$$

nas quais, $\lambda_{da} = \lambda_{PM}(K_5 - K_7)$, $\lambda_{db} = \lambda_{PM}(K_{13} - K_{11})$, $\lambda_{qa} = \lambda_{PM}(K_5 + K_7)$, $\lambda_{qb} = \lambda_{PM}(K_{13} + K_{11})$, sendo $K_{2n+1} = (2n+1)k_{2n+1}$.

A equação das tensões fica da seguinte forma:

$$V_{dq} = R_{dq}I_{dq} + L_{dq}(\theta_r)\frac{dI_{dq}}{dt} + \omega_r \frac{dL_{dq}(\theta_r)}{d\theta_r} I_{dq} + \omega_r \phi'_{rdq}(\theta_r) \quad (3.45)$$

A expressão do conjugado:

$$c_e = \frac{p}{2} I_{dq}^t \frac{dL_{dq}(\theta_r)}{d\theta_r} I_{dq} + p I_{dq}^t \phi'_{rdq}(\theta_r) \tag{3.46}$$

3.2.4 Representação dinâmica do motor

Pode-se verificar que o modelo dinâmico do IPM pode ser expresso pelas seguintes equações (o termo (θ_r) que indica a dependência de L_{dq} e ϕ_{rdq} fica implícito no decorrer do trabalho):

$$V_{dq} = R_{dq} I_{dq} + L_{dq}(\theta_r) \frac{d}{dt} I_{dq} + U_{dq} \tag{3.47}$$

$$U_{dq} = \omega_r \left(L'_{dq} I_{dq} + \phi'_{rdq} \right)$$

$$c_e = \frac{p}{2} I_{dq}^T L'_{dq} I_{dq} + p I_{dq}^T \phi'_{rdq} \tag{3.48}$$

$$J \frac{d}{dt} \omega_r = p(c_e - cm) - f_\omega \omega_r \tag{3.49}$$

$$V_{dq} = [v_d \quad v_q]^T, \quad I_{dq} = [i_d \quad i_q]^T,$$

$$R_{dq} = \begin{bmatrix} r_s & 0 \\ 0 & r_s \end{bmatrix}, \qquad L_{dq} = \begin{bmatrix} l_d(\theta_r) & 0 \\ 0 & l_q(\theta_r) \end{bmatrix},$$

$$L'_{dq} = \frac{dL_{dq}}{d\theta_r} = \begin{bmatrix} l_{dh} \cos 6\theta_r & l_{ac}(\theta_r) \\ l_{ac}(\theta_r) & l_{qh} \cos 6\theta_r \end{bmatrix},$$

$$\phi_{r_{dq}} = \begin{bmatrix} \lambda_{r_d}(\theta_r) \\ \lambda_{r_q}(\theta_r) \end{bmatrix}, \qquad \phi'_{r_{dq}} = \frac{d\phi_{r_{dq}}}{d\phi_r},$$

$$l_d(\theta_r) = L_d + l_{d1} \cos(6\theta_r), \qquad l_q(\theta_r) = L_q + l_{q1} \cos(6\theta_r)$$

$$L_d = L_0 + M_0 - \frac{1}{2} L_1 + M_1, L_q = L_0 + M_0 + \frac{1}{2} L_1 - M_1,$$

$$l_{ac}(\theta_r) = l_{ac0} + l_{ac1} \cos 6\theta_r,$$

$$l_{d1} = -\frac{1}{2} L_2 + M_2 + L_3 - M_3 + \frac{1}{2} L_4 + M_4, \qquad l_{q1} = \frac{1}{2} L_2 - M_2 + L_3 - M_3 - \frac{1}{2} L_4 - M_4,$$

$$l_{dh} = 2L_2 - 4M_2 - 6L_3 + 6M_3 - 4L_4 - 8M_4,$$

$$l_{qh} = -2L_2 + 4M_2 - 6L_3 + 6M_3 + 4L_4 + 8M_4,$$

$$l_{ac0} = L_d - L_q, \qquad l_{ac1} = 2L_2 - 4M_2 + 4L_4 + 8M_4,$$

$$\lambda'_{r_d}(\theta_r) = \lambda_{da} \sin 6\theta_r + \lambda_{db} \sin 12\theta_r,$$

$$\lambda'_{r_q}(\theta_r) = \lambda_{PM} + \lambda_{qa} \cos 6\theta_r + \lambda_{qb} \cos 12\theta_r.$$

Segundo (Costa 2013) os parâmetros de l_{dh} e l_{qh} não são relevantes para as equações do modelo dinâmico, logo, serão desconsideradas. As equações (3.50), (3.51) e (3.52) representam uma visão expandida das equações (3.47) e (3.48) do modelo dinâmico do motor.

$$v_d = r_s i_d + l_d(\theta_r)\frac{di_d}{dt} + l_{ac}(\theta_r) i_q + \lambda'_{r_d}(\theta_r)\omega_r \tag{3.50}$$

$$v_q = r_s i_q + l_q(\theta_r)\frac{di_q}{dt} + l_{ac}(\theta_r) i_d + \lambda'_{r_q}(\theta_r)\omega_r \tag{3.51}$$

$$c_e = p\left(i_q \lambda'_{r_q}(\theta_r) + i_d \lambda'_{r_d}(\theta_r) + l_{ac}(\theta_r) i_d i_q\right) \tag{3.52}$$

A Figura 5 mostra uma representação geral simplificada para o IPM.

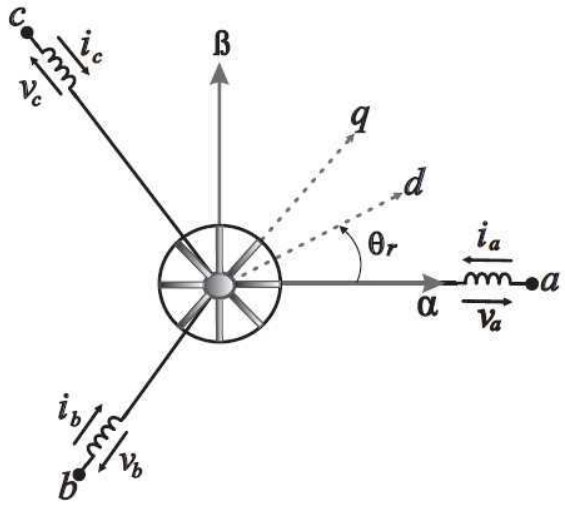


Figura 5. Representação simplificada da IPM

3.3 ACIONAMENTO DA MÁQUINA SÍNCRONA A IMÃ PERMANENTE

Além do motor síncrono a imã permanente, outro dispositivo será simulado no FPGA: o inversor Fonte de Tensão. Com o objetivo de testar estes dispositivos simulados, optou-se em utilizar um sistema de controle baseado na orientação pelo campo. Este sistema de controle, juntamente com as características de um acionamento típico, é descrito nesta seção.

A maioria das máquinas síncronas a imã permanente de corrente alternada apresentam características em comum para operação: malhas de controle de velocidade; enfraquecimento de campo e controle vetorial; controle de malha de corrente. A Figura 6 apresenta o esquema que contém as malhas de controle citadas. O sistema de controle recebe do operador a velocidade de referência, e as medidas das grandezas elétricas e mecânicas são oferecidas pelos seus respectivos sensores. A ação do algoritmo de controle e a correta ação de chaveamento do conversor de potência controla o fluxo de potência da fonte para o motor. O conjugado de referência é definido pela malha de velocidade. As correntes de referência são obtidas utilizando a estratégia de orientação pelo campo. O controlador de corrente força as correntes medidas a rastrearem os sinais de referência gerando os estados de chaveamento do conversor.

Os algoritmos de controle que representam as malhas de controle de posição, velocidade e torque, além das funções de aquisição das medições e geração dos sinais de comando são, tipicamente, implementados digitalmente em microprocessadores ou processadores digitais de sinais (DSP).

O sistema de medição normalmente engloba sensores para a aquisição das grandezas elétricas e mecânicas. As grandezas elétricas amostradas normalmente são a tensão do barramento do conversor de potência, as correntes das fases e tensões das máquinas com base em circuitos específicos utilizando sensores de efeito Hall e conversores A/D. No caso das grandezas mecânicas, como a velocidade e posição do rotor, são utilizados sensores de posição, como *resolvers* ou *encoders*, instalados no eixo da máquina.

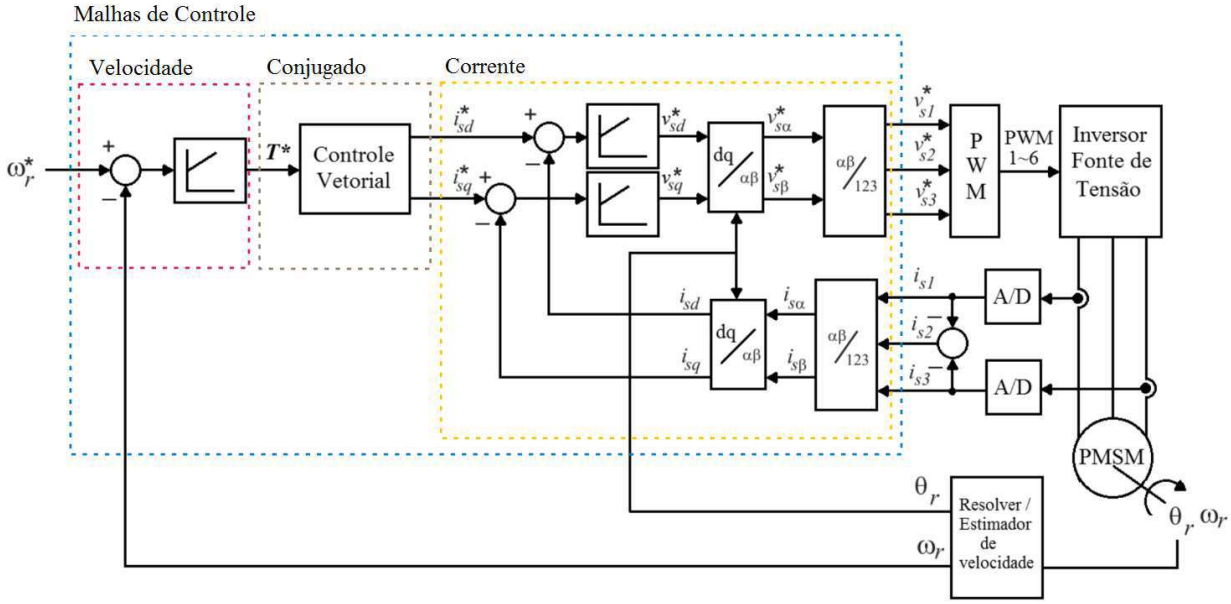


Figura 6. Diagrama de controle da máquina SMPM

3.3.1 Controle por orientação pelo campo

A estratégia de orientação pelo campo é geralmente implementada mantendo a componente de eixo direto I_{sd} nula ($I_{sd} = 0$) e controlando a componente em quadratura. Assim o torque elétrico da máquina é dado por:

$$T_t = p\lambda_{pm} I_{sq} \tag{3.53}$$

Através da equação (3.53) pode se perceber que o torque da máquina é diretamente proporcional ao fluxo magnético dos imãs permanentes (fluxo produzido pelo rotor) e a componente da corrente de eixo em quadratura I_{sq}, que é a corrente responsável pela produção do torque. Portanto, para a ontenção de um torque constante é necessário manter a componente I_{sq} constante.

A operação com $I_{sd} = 0$ é adequada para acionamentos até a velocidade nominal, na região de torque constante, e onde a tensão real fornecida pela fonte é suficiente. Entretanto, para velocidades acima da nominal, na faixa de operação de potência constante, a fcem aumenta proporcional à velocidade do rotor. Esse aumento da fcem no estator exige um aumento da tensão terminal. Entretanto, pode-se reduzir o valor da força contra-eletromotriz utilizando uma técnica de enfraquecimento de campo. Este enfraquecimento de campo é

conseguido pela introdução de uma componente de eixo direto I_{sd} na direção oposta ao do fluxo ($I_{sd} = 0$).

Na condição de enfraquecimento de campo a introdução de valores negativos de I_{sd} provoca um efeito desmagnetizante, entretanto, analisando-se a expressão (3.24), verifica-se que os valores negativos de I_{sd} contribuem para um aumento no torque desenvolvido ($L_{sq} > L_{sd}$). Assim sendo, um incremento de torque é acompanhado por uma redução do fluxo do estator.

No caso de valores positivos de I_{sd} há um aumento do fluxo magnético e redução considerável do torque. Para que a máquina SMPM opere com corrente nominal é necessário a redução de I_{sq}, de modo que a corrente resultante i_s não exceda o limite da corrente da máquina, ou seja:

$$i_s = \sqrt{I_{sd}^2 + I_{sq}^2} < i_{s\,max} \tag{3.54}$$

3.3.2 Modulação por largura de pulsos (*Pulse-Width Modulation* - PWM)

O inversor trifásico da Figura 7 é composto por seis chaves q_1, q_2, q_3, $\overline{q_1}$, $\overline{q_2}$, $\overline{q_3}$ e seus respectivos diodos. As chaves $\overline{q_1}$, $\overline{q_2}$, $\overline{q_3}$ são comandadas de forma complementar as chaves q_1, q_2, q_3.

As tensões aplicadas a máquina dependem da configuração das chaves do inversor. As chaves assumem valores binários (0 ou 1). As tensões de saída do inversor são dadas por:

$$v_{s1} = v_{s10} + v_{0N} \tag{3.55}$$

$$v_{s2} = v_{s20} + v_{0N} \tag{3.56}$$

$$v_{s3} = v_{s30} + v_{0N} \tag{3.57}$$

Onde v_{0N} é a diferença de tensão do ponto médio do barramento CC (“0”) para o neutro da máquina. As tensões de polo v_{s10}, v_{s20}, v_{s30} são dadas por:

$$v_{s10} = q_1 \frac{E}{2} - \overline{q_1} \frac{E}{2} = (2q_1 - 1) \frac{E}{2} \tag{3.58}$$

$$v_{s20} = q_2 \frac{E}{2} - \overline{q_2} \frac{E}{2} = (2q_2 - 1) \frac{E}{2} \tag{3.59}$$

$$v_{s30} = q_3 \frac{E}{2} - \overline{q_3} \frac{E}{2} = (2q_3 - 1) \frac{E}{2} \tag{3.60}$$

Sejam v_{s1}^*, v_{s2}^*, v_{s3}^* as tensões trifásicas de referência que se deseja aplicar a máquina, pode-se utilizar as tensões de referência de polo v_{s10}^*, v_{s20}^*, v_{s30}^*, para determinar as relações para as larguras de pulso τ_1, τ_2, τ_3:

$$\tau_1 = \left(\frac{v_{s10}^*}{E} + \frac{1}{2}\right) T \tag{3.61}$$

$$\tau_2 = \left(\frac{v_{s20}^*}{E} + \frac{1}{2}\right) T \tag{3.62}$$

$$\tau_3 = \left(\frac{v_{s30}^*}{E} + \frac{1}{2}\right) T \tag{3.63}$$

3.3.3 Inversor fonte de tensão

O inversor fonte de tensão é o conversor estático empregado em sistemas de acionamento de máquinas elétricas de corrente alternada. A topologia básica deste conversor é apresentada na Figura 7. Esta estrutura será simulada em conjunto com o motor no FPGA. A tensão no barramento CC é obtida por um circuito retificador. O inversor é constituído de três ramos (braços) que são constituídos de duas chaves de potência cada, a carga é acoplada ao ponto central dos braços.

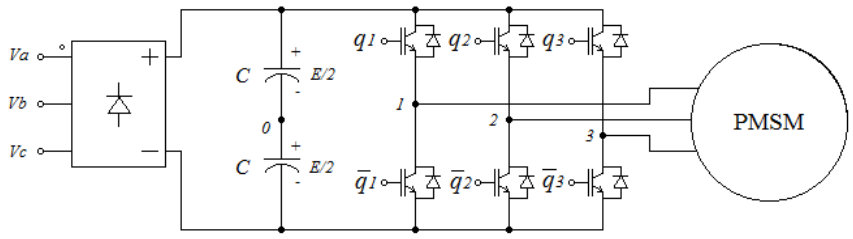


Figura 7. Sistema de alimentação da Máquina SMPM

As tensões de saída do inversor são produzidas a partir de uma técnica de modulação por largura de pulsos (*Pulse With Modulation – PWM*), aplicados as chaves de cada braço de forma complementar. As tensões geradas têm valores instantâneos cujo valor médio, em um intervalo de tempo *T*, é igual a tensão de referencia. As tensões de saída estão representadas nas equações (3.55), (3.56) e (3.57).

3.4 CONCLUSÕES

Este capítulo apresentou as características das síncronas a imã permanente utilizadas neste trabalho, considerando o modelo matemático de cada máquina e o sistema genérico de controle. A primeira máquina (SMPM) tem o formato da fcem praticamente senoidal, sem apresentar não linearidades. Já a segunda máquina (IPM) apresenta um grande conteúdo harmônico nas indutâncias e fluxos, o que representa as não linearidades da máquina, tornando-a mais complexa e exigindo mais do sistema de controle.

No FPGA será simulado o modelo da máquina no referencial síncrono do rotor (dq) juntamente com o conversor de potência. O sistema de controle apresentado serve de referência para os testes de simulação e será executado a partir de um DSP.

O sistema de controle é composto por malhas de controle em cascata, onde as malhas de controle de velocidade antecedem a malha de controle de torque/corrente. Os valores de referência a serem aplicados no algoritmo PWM são obtidos pelo processamento destas malhas. Os sinais PWM são utilizados para o chaveamento do inversor fonte de tensão que alimenta a máquina.

O controle é baseado na orientação pelo campo. O seu princípio é a orientação da componente de eixo q da corrente estatórica I_{sq}, em quadratura com o fluxo do rotor λ_{pm}, mantendo a componente I_{sd} com valor nulo. Assim, o torque desenvolvido pela máquina tem relação direta com a componente I_{sq}, o que simplifica a implementação do controle por orientação pelo campo.

Capítulo 4

Plataforma De Simulação Em Tempo Real

O desenvolvimento da plataforma HIL segue o esquema demonstrado na Figura 1 da seguinte forma: controlador real (baseado em DSP); atuadores simulados; planta simulada; e sensores simulados. O objetivo é realizar estas atividades utilizando dois dispositivos: um para embarcar as estratégias de controle e PWM; e uma placa de desenvolvimento baseada em FPGA para simular o modelo da máquina, o modelo do conversor de potência e sensores ideais. A conexão entre os dois dispositivos é feita mediante entradas e saídas digitais.

Inicialmente, se considerou em utilizar um segundo FPGA para realizar a tarefa do controlador, que executa o esquema de controle e o comando PWM antes de se iniciar as atividades com o DSP. Uma plataforma baseada neste conceito foi desenvolvida. Entretanto, devido a dificuldade em se trabalhar em ponto fixo e sintaxe Verilog, frente à linguagem C e ponto flutuante com DSP, o conceito foi descontinuado. Porém, a plataforma desenvolvida serviu como base para a adaptação do DSP.

Ao longo do trabalho, os testes em FPGA foram implementados e simulados nas ferramentas do *software* Quartus II, versão 10.1. O Quartus II foi utilizado para a (i) entrada de projeto, (ii) simulação funcional/tempo (ModelSim *Starter Edition*), (iii) síntese do projeto, (iv) ajuste do projeto para o FPGA, (v) programação do FPGA.

4.1 MODELO MATEMÁTICO DA MÁQUINA DE TESTES

Nesta etapa do projeto, para a concepção das plataformas o modelo da máquina utilizado foi de uma SMPM discretizada por Euler e passo de cálculo de 1µs.

As equações na forma matricial, descritas em (3.21), foram divididas em duas partes como segue:

1) Equacionamento da tensão Vsd:

$$V_{sd} = R_s.I_{sd} - W_r.L_q.I_{sq} + L_d.\frac{dI_{sd}}{dt} \quad (4.1)$$

Aplicando o teorema de Laplace obtêm-se a seguinte expressão de corrente:

$$I_{sd}(s) = \frac{V_{sd}(s)}{sL_d + R_s} + \frac{E_d(s)L_q}{sL_d + R_s} \quad (4.2)$$

onde $E_d(s) = W_r(s).I_{sq}(s)$.

A equação (5.2) pode ser reescrita da seguinte forma:

$$I_{sd}(s) = X_{isd1}(s) + X_{isd2}(s) \quad (4.3)$$

onde

$$X_{isd1}(s) = \frac{V_{sd}(s)}{sL_d + R_s} \quad (4.4)$$

$$X_{isd2}(s) = \frac{E_d(s)L_q}{sL_d + R_s} \quad (4.5)$$

2) Equacionamento da tensão Vsq:

$$V_{sq} = R_s.I_{sq} + W_r L_d I_{sd} + L_q \frac{dI_{sq}}{dt} + f_{lm}W_r \quad (4.6)$$

Aplicando o teorema de Laplace chega-se a seguinte expressão de corrente:

$$I_{sq}(s) = \frac{V_{sq}(s)}{sL_q + R_s} - \frac{E_q(s)L_d}{sL_q + R_s} - \frac{f_{lm}W_r(s)}{sL_q + R_s} \quad (4.7)$$

onde $E_q(s) = w_r(s)I_{sd}(s)$

A equação de corrente (4.7) pode ser reescrita da seguinte forma:

$$I_{sq}(s) = X_{isq1}(s) - X_{isq2}(s) - X_{isq3}(s) \quad (4.8)$$

onde:

$$X_{isq1}(s) = \frac{V_{sq}(s)}{sL_q + Rs} \tag{4.9}$$

$$X_{isq2}(s) = \frac{L_d E_q(s)}{sL_q + R_s} \tag{4.10}$$

$$X_{isq3}(s) = \frac{f_{lm} w_r(s)}{sL_q + R_s} \tag{4.11}$$

Com relação a equação mecânica de movimento, aplicando Laplace em (3.25) obtém-se a seguinte função de transferência:

$$\frac{w_m(s)}{Torque(s)} = \frac{1}{s(j + j_m) + f_w} \tag{4.12}$$

onde $Torque(s) = T_t(s) - T_c(s)$

O método de Euler corresponde a aproximar o mapeamento exato entre **z** e **s** por uma expansão em série truncada, ou seja:

$$z = e^{sT} \approx 1 + sT \tag{4.13}$$

Manipulando a expressão, tem-se:

$$s = \frac{z - 1}{T} \tag{4.14}$$

De acordo com as equações de corrente (4.3) e (4.8), pode-se determinar as equações discretas equivalentes pelo método de Euler:

$$I_{sd}[k] = X_{isd1}[k] + X_{isd2}[k] \tag{4.15}$$

$$I_{sq}[k] = X_{isq1}[k] - X_{isq2}[k] - X_{isq3}[k] \tag{4.16}$$

Da mesma forma a equação (3.24) que representa o conjugado da máquina pode ser reescrita como visto na equação (4.17).

$$T_t[k] = P\{\lambda_{pm} I_{sq}[k] + (L_d - L_q) I_{sd}[k] I_{sq}[k]\} \quad (4.17)$$

Utilizando-se das equações (4.4), (4.5), (4.9) a (4.12) e aplicando o método de Euler, com passo de simulação de h, chegam-se às respectivas equações de modelo discreto descritas abaixo.

Para a equação de corrente I_{sd}:

$$X_{isd1}[k] = Cd_1 . X_{isd1}[k-1] + Cd_3 . V_{sd}[k] \quad (4.18)$$

$$X_{isd2}[k] = Cd_2 . X_{isd2}[k-1] + Cd_4 . w_r[k-1] I_{sq}[k-1] \quad (4.19)$$

Para a equação de corrente I_{sq}:

$$X_{isq1}[k] = Cq_1 . X_{isq1}[k-1] + Cq_2 . V_{sq}[k] \quad (4.20)$$

$$X_{isq2}[k] = Cq_1 . X_{isq2}[k-1] + Cq_3 . I_{sd}[k-1] . w_r[k-1] \quad (4.21)$$

$$X_{isq3}[k] = Cq_1 . X_{isq3}[k-1] + Cq_4 . w_r[k-1] \quad (4.22)$$

Para a equação de movimento:

$$w_m[k] = w_m[k-1] + Cwm . T_t[k] \quad (4.23)$$

O w_r é encontrado a partir da relação de w_m com o número de pólos:

$$w_r[k] = P . w_m[k] \quad (4.24)$$

E por fim, a posição angular do rotor é encontrada por:

$$\theta_r[k] = \theta_r[k-1] + w_r[k] . h \quad (4.25)$$

Onde h é o passo de simulação, as constantes descritas por $Cd_n, Cq_n \; e \; Cwm$ (Tabela 5) são resultantes da aplicação do método relacionado com os parâmetros do motor (Tabela 4). A representação das constantes em ponto fixo foi feita com Q-21.

Tabela 4 Servomotor WEG SWA 40-1,6-30

Potência Nominal	0,45 KW
Rs	6,187 Ω
Lq	0,033 H
Ld	0,024 H
V/krpm	56,16
Velocidade	3000 RPM
Corrente Nominal	2 A
Inércia	0,000084 Kg.m²
Pares de Pólos	4

Tabela 5. Constantes do modelo da máquina SMPM de testes.

Parâmetro	**Valor**
$\boldsymbol{Cd_1}$	0,999742208333
$\boldsymbol{Cd_2}$	0,999742274772749
$\boldsymbol{Cd_3}$	$4{,}167741075584777 \times 10^{-5}$
$\boldsymbol{Cd_4}$	$1{,}37464562781253 \times 10^{-6}$
$\boldsymbol{Cq_1}$	0,99981251515151
$\boldsymbol{Cq_2}$	$3{,}0303030303 \times 10^{-5}$
$\boldsymbol{Cq_3}$	$7{,}272727272727 \times 10^{-7}$
$\boldsymbol{Cq_4}$	$2{,}872121212121212 \times 10^{-6}$

4.2 PLATAFORMA DE SIMULAÇÃO 1

A partir das combinações possíveis de um sistema HIL dado pela Figura 1, diz-se que neste sistema os atuadores, a planta e os sensores são simulados. A Figura 8 demonstra o acionamento típico de uma PMSM. Nota-se que o FPGA simula o equivalente do Inversor Fonte de Tensão e a máquina síncrona a imã permanente, e um *hardware* dedicado (DSP) embarca os algoritmos de controle. A interface entre os dois dispositivos que corresponde aos sensores A/D, sensores de posição e velocidade e sinais PWM são emulados mediante um canal de comunicação digital.

4.2.1 Implementação da plataforma HIL

Esta seção descreve o desenvolvimento do sistema que utiliza dois FPGA para a simulação em tempo real, como utilizado no trabalho de (Huller, Fernandes e Oliveira 2012) e (Fernandes, et al. 2012).

O módulo de controle da máquina, representado pelas malhas de controle no diagrama de blocos indicado na Figura 8, foi implementado em um segundo FPGA (placa DE-0 da Altera com o FPGA Cyclone III). Este módulo simula o equivalente de um controlador, que

estaria implementado em um dispositivo DSP, em um sistema real. O dispositivo que simula o modelo da máquina, conversor de frequência e sensores é uma placa Altera DE-2 equipado com um FPGA Cyclone II 2C35. A Figura 9 mostra a estrutura básica da plataforma utilizada para os testes. Os dois FPGA estão interligados por conectores de expansão (denominados GPIO) presentes nas duas placas. Os pinos GPIO oferecem acesso externo aos pinos correspondentes no FPGA.

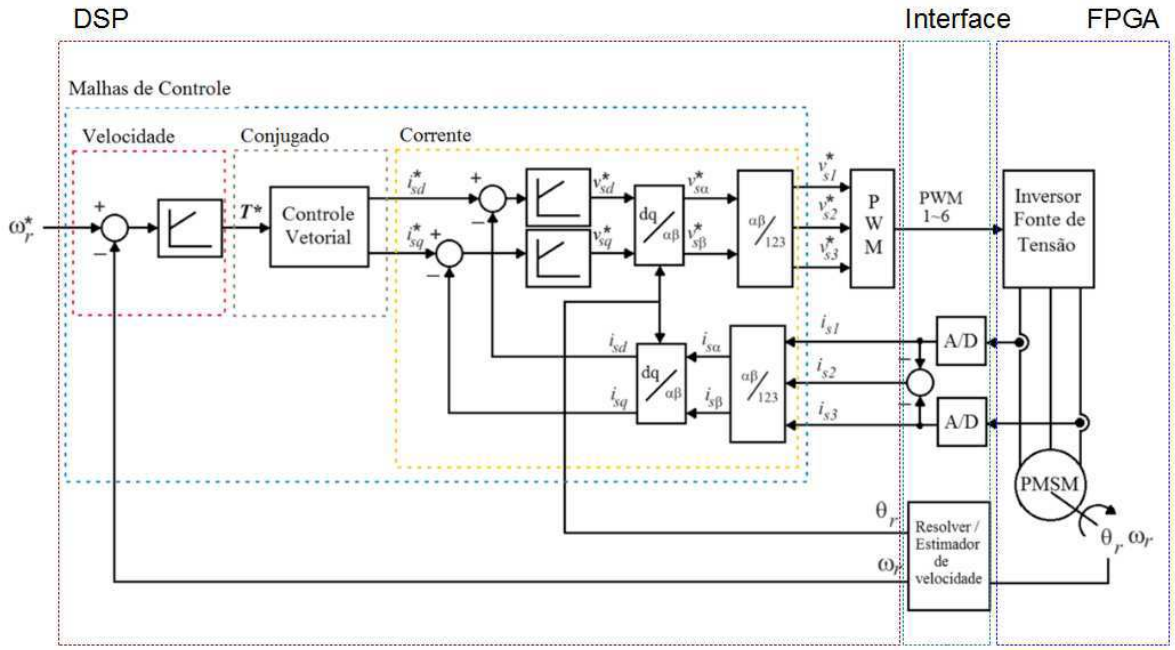


Figura 8. Interface entre Simulador e Controlador

O microcomputador conectado pela interface USB-Serial RS-232 à placa DE-2 executa um software de monitoramento e parametrização da simulação. Com ele é possível visualizar todas as variáveis de interesse e modificar de modo *on-line* asa características da planta, como a carga aplicada no eixo do motor simulado.

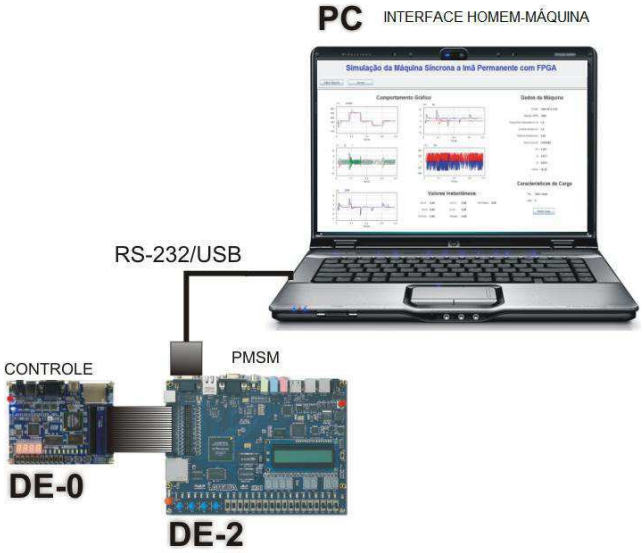


Figura 9. Plataforma 2 de simulação

A Figura 10 apresenta o diagrama de blocos detalhado da plataforma. Na placa FPGA DE2 foram implementados cinco módulos. As entradas e saídas de cada módulo estão representadas no diagrama através das setas.

- O módulo “Máquina PMSM” contém o modelo matemático da máquina.
- As tensões trifásicas de alimentação (Vs1, Vs2, Vs3) são obtidas pelo equacionamento do “Inversor Fonte de Tensão” que calcula os valores de tensão com base no estado das chaves (q1, q2 e q3) provenientes do módulo “INTERFACE”.
- O bloco “Funções” fornece ao módulo da máquina os valores de seno e cosseno necessários para as transformações de coordenadas.
- O módulo “INTERFACE” realiza a troca de dados entre os dispositivos.
- "Divisor de Frequência" é o módulo responsável pela aplicação do sinal de *clock*.

Na Figura 10 são apresentados os módulos de Controle e PWM, Interface, Inversor Fonte de Tensão e divisor de frequência, com seus respectivos sinais de entrada/saída. Todos os módulos foram desenvolvidos usando linguagem de descrição de *hardware* Verilog. A estratégia de controle representada no diagrama da Figura 8 foi reestruturada para aritmética de ponto fixo e descrita na linguagem de *hardware* para ser embarcada na placa DE-0.

O sistema é comandado por um sinal de *clock* (Clk). O módulo Máquina PMSM calcula o equacionamento da máquina e o equacionamento do inversor ideal através do

chaveamento deste sinal. A cada novo ciclo de *clock,* é acionado um bloco que calcula as equações em ordem sequencial. Ao término da operação são disponibilizados nas saídas do módulo os valores calculados das grandezas da máquina. Desta forma, é importante garantir que a frequência do sinal de *clock* aplicado seja maior que o tempo necessário para computar todas as equações no módulo.

O sinal de *clock* (Clk), presente no diagrama é obtido pelo módulo divisor de frequência através de um *clock* nativo de placa de 50MHz. A conexão destes módulos conforme observada na Figura 10 implementa o sistema HIL de uma máquina SMPM, onde avaliações de estratégias de controle das mesmas podem ser simuladas.

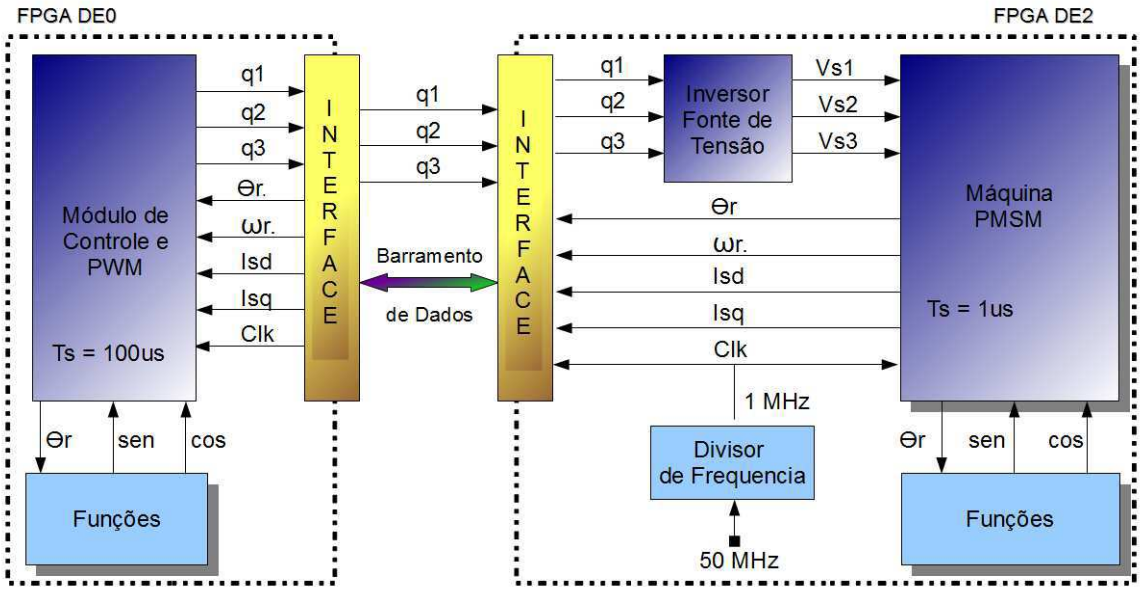


Figura 10. Módulos implementados nos dispositivos FPGA da Plataforma 2

Na Figura 10 também é possível observar a distribuição dos módulos do sistema de simulação da máquina SMPM entre as duas placas FPGA. Na placa DE-0 são implementados os módulos: "Módulo de controle e PWM", o módulo "Interface", e o módulo "Funções".

O módulo "Funções" retorna valores de seno e cosseno a partir de um ângulo de entrada. No módulo são armazenados valores tabelados de seno e cosseno de 0° a 359° com resolução de 1°. A obtenção dos valores intermediários é calculada como mostra a equação (4.26) para o caso do seno, o cosseno é calculado de forma similar.

$$\sin[\theta_r] = \left(1 + \theta_{r_{int}} - \theta_r\right) * \sin[\theta_{r_{int}}] + \left(\theta_r - \theta_{r_{int}}\right) * \sin[\theta_{r_{int}} + 1] \qquad (4.26)$$

onde θ_r é o ângulo em graus e $\theta_{r_{int}}$ é a parte inteira do ângulo em graus.

O “Módulo de Controle e PWM” é o responsável por gerar os sinais PWM que controlarão as chaves do conversor estático de potência simulado pelo módulo “Inversor Fonte de Tensão”. O equacionamento do “Inversor Fonte de Tensão”, gera em sua saída, as tensões trifásicas que alimentam a máquina simulada através do módulo “Máquina SMPM”, ambos localizados na placa DE-2. A conexão entre as placas FPGA é realizada através dos módulos “Interface”. Estes módulos são responsáveis pela conexão e troca de informações entre as placas FPGA, conforme ilustrado na Figura 10**.**

O “barramento de dados”, presente na “Interface” de cada um dos módulos, emula o que seriam os sensores de posição, velocidade e correntes trifásicas que estariam sendo medidos diretamente na máquina. Também são representados os canais PWM, que propagam os sinais de controle das chaves de potência do conversor estático simulado no módulo “Inversor Fonte de Tensão”.

Nesta plataforma não foi necessário utilizar nenhum artifício para a sincronização entre os dispositivos. Uma vez que, ambos FPGA são operados pelo mesmo sinal de *clock*.

3.2.2 Descrição da interface HIL

A obtenção das grandezas da máquina pelo controlador é feito mediante uma transação iniciada pelo módulo “Interface” da placa DE-0, o módulo “Interface” na placa DE-2 transmite as grandezas de forma multiplexada no barramento de dados. Entretanto, se trata de um envio sequencial, é preciso atentar para que todos os dados estejam disponíveis ao controlador antes do inicio de um novo período de controle.

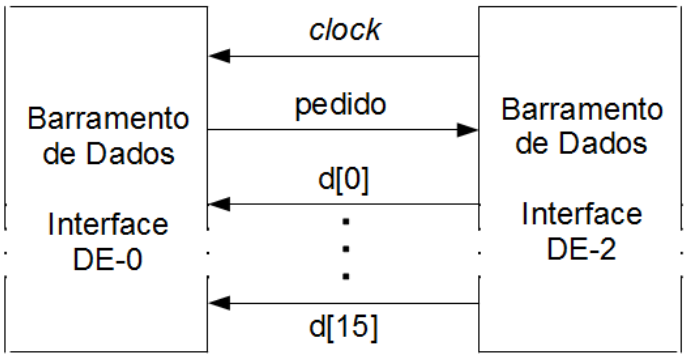


Figura 11. Esquema do barramento de dados da simulação HiL

O esquema do barramento de dados pode ser visto na Figura 11**.** É composto por um sinal de "*clock"*, um sinal de "pedido" e uma via de dados de 16 bits d[15:0], o que determina que todas as grandezas a serem transmitidas apresentem uma mesma resolução.

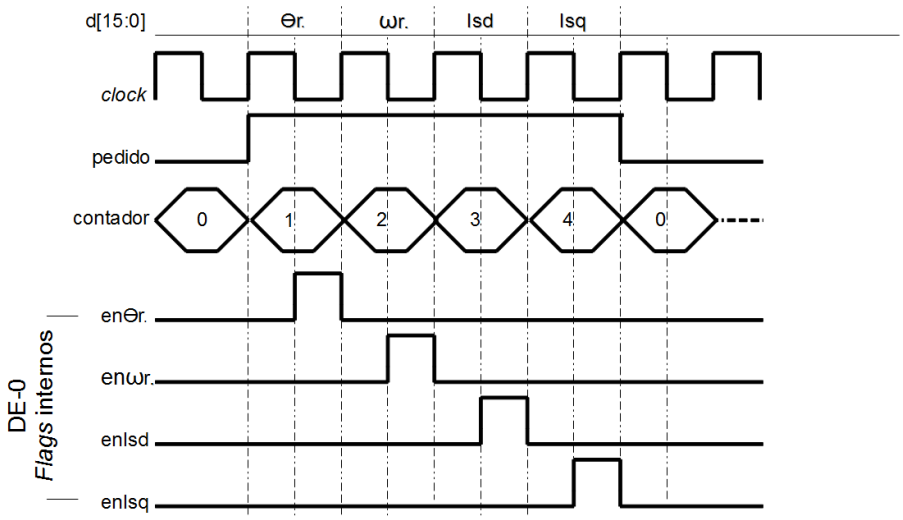


Figura 12. Aquisição de dados pela DE-0

A Figura 12 apresenta um diagrama de tempo da sequencia de envio dos dados da placa DE-2 (Módulo Inversor de Tensão e módulo Máquina PMSM) para a placa DE-0 (Módulo de Controle). São quatro grandezas a serem transmitidas: Ângulo rotórico, velocidade, corrente de eixo d e corrente de eixo q. Os flags internos da "Interface" DE-0 mostrados na figura operam como um gatilho. Estes gatilhos permitem que a variável disponibilizada na via de dados (d[15:0]) seja devidamente armazenada no seu respectivo registrador de palavra. Por exemplo, quando o bit "enIsd" estiver em 1 o valor que estiver no barramento de dados será armazenado na variável correspondente a corrente de eixo d ("Isd") do módulo de controle.

O procedimento seguido para a transmissão dos dados ocorre da seguinte forma:

- A cada intervalo de 100µs o controlador realiza a amostragem das variáveis. O inicio da amostragem é caracterizado pelo bit "pedido", fazendo seu valor variar de 0 para 1 é caracterizando o "inicio da transação" da aquisição de dados.
- As duas interfaces possuem uma variável denominada contador que é incrementado a cada pulso de *clock* quando o bit de pedido for 1. Este contador

identifica qual variável será disponibilizada no barramento de dados. Quando o bit de pedido for 0 o contador é zerado.

- A "Interface" DE-2 reconhece o evento de "inicio da transação" e imediatamente disponibiliza na via de dados a variável correspondente ao contador. A variável é mantida até que o valor do contador seja mudado.
- Com o valor de pedido em 1, a Interface DE-0 identifica qual variável deve armazenar e aguarda o pulso de *clock* ir a zero para habilitar o gatilho correspondente.
- Este ciclo se repete até que o valor do contador seja igual a 4 e todas as variáveis sejam transmitidas. Quando o contador tiver valor 4 e o próximo ciclo de *clock* se iniciar é caracterizado o fim da transação, e o bit de pedido é posto em 0.

O tempo total da amostragem dos sinais é de 4µs, considerando a frequência do *clock* imposta de 1MHz. O tempo total corresponde ao produto do numero de variáveis e o período de *clock*.

4.2.3 Requisitos de tempo e *hardware* do FPGA

A compilação do projeto no ambiente Quartus II inclui uma análise de tempo (*Timing Analisys and Simulation*). Dentre os objetos de análise se encontra as vias de *clock*. Por exemplo, a via "Clk" (Figura 10) do módulo "Máquina PMSM" dispara um bloco interno do módulo que realiza os cálculos do modelo matemático da máquina. A ferramenta de análise verifica a latência e qual a frequência máxima de *clock* que pode ser imposta na via para que toda a rotina seja calculada e, suas saídas sejam atualizadas antes que ocorra um novo ciclo. A Tabela 6 descreve todas as vias de *clock* identificadas pela ferramenta de análise e qual a frequência máxima que pode ser imposta a cada via. De acordo com os resultados, embora o modelo matemático da máquina seja discretizado a um passo de cálculo de 1µs (1MHz), percebe-se que nesta configuração é possível chegar a modelos com um passo de integração de aproximadamente 111 ns (9,47 MHz), o que pode ser visto como uma reserva caso seja necessário um passo de cálculo menor.

Tabela 6. Frequencias Máximas das vias de clock

Nome do clock	Frequência Máxima
Clk	9,47 MHz
Clock_50	210 MHz
PortaSerial:TX	124,58 MHz
PortaSerial:RxD_data_ready	274,8 MHz

A Tabela 7 apresenta o consumo de recursos obtidos pela compilação no Quartus II do projeto do FPGA DE-2. Mesmo contendo mais módulos que a plataforma 1, a utilização dos bits de memória para armazenar os valores de senos e cossenos possibilitou uma redução do consumo de recursos necessários para embarcar o projeto no dispositivo. A Tabela 8 apresenta a relação dos recursos para o projeto da placa DE-0. Devido a menor quantidade de elementos lógicos, quase todos os recursos foram utilizados para embarcar o sistema de controle.

Tabela 7. Consumo de recursos do FPGA DE-2 da plataforma 1

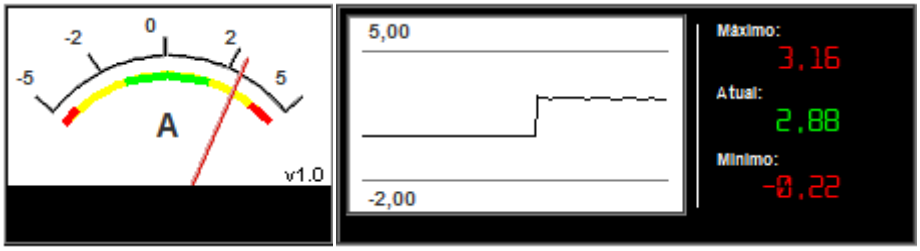
Descrição	Necessário	Disponível	(%)
Elementos lógicos (LEs)	22379	33216	67
Bits de memória	30120	483840	7
Multiplicadores 9bits	70	70	100
PLL	0	4	0

Tabela 8. Consumo de recursos do FPGA DE-0 da plataforma 1

Descrição	Necessário	Disponível	(%)
Elementos lógicos (LEs)	13378	15408	87
Bits de memória	0	516096	0
Multiplicadores 9bits	112	112	100
PLL	0	4	0

4.2.4 Interface homem-máquina

Nesta etapa do projeto a interface Homem-máquina foi aperfeiçoada (Figura 14). É possível verificar graficamente e em tempo real o comportamento das variáveis. Existem dois tipos de medidores: analógicos e gráficos. Os medidores analógicos (Figura 13-a) mostram os valores instantâneos da grandeza em um ponteiro, enquanto que os digitais (Figura 13-b) apresentam o comportamento gráfico no tempo da variável, mostrando, inclusive, o valor instantâneo, máximo e mínimo registrado.



(a) (b)

Figura 13. Medidores - (a) Analógico; (b) Gráfico.

O campo “Características da carga” (Figura 14) permite que o usuário altere o valor da carga aplicada no eixo da máquina de modo *on-line*. A Figura 14 também apresenta o resultado de uma simulação. A máquina foi acionada e ajustada a uma velocidade mecânica de 125 rad/s. Em regime, uma carga de 1.25 N.m foi aplicada no eixo. Através dos medidores gráficos é possível acompanhar os transitórios desta aplicação.

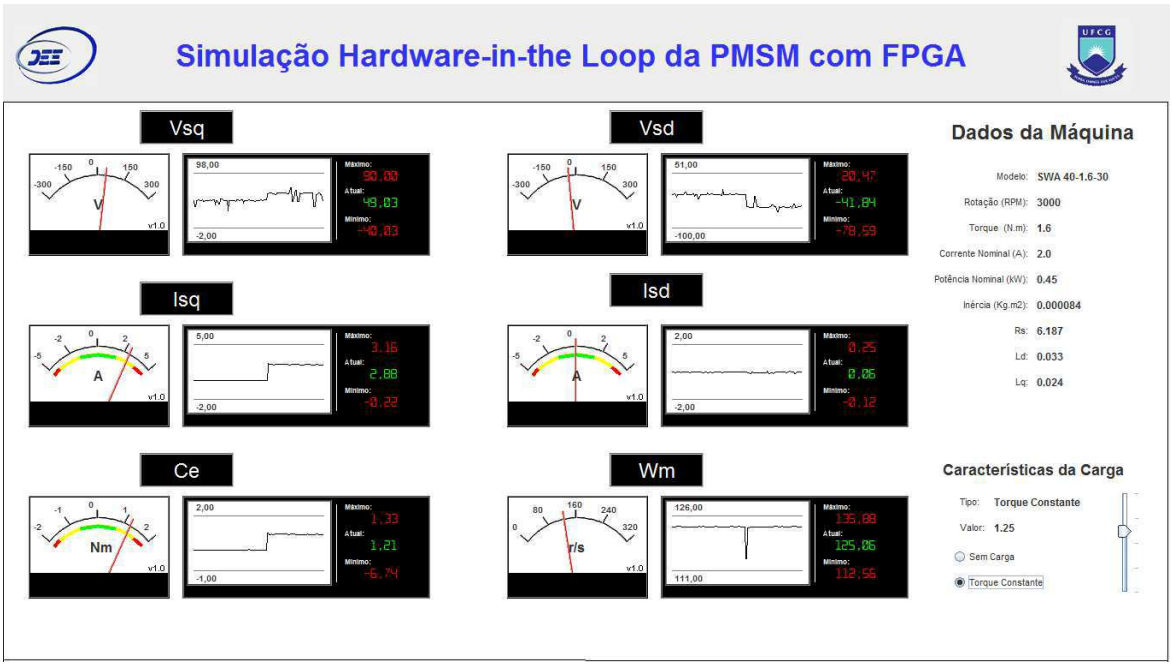


Figura 14. Interface Homem-Máquina do Sistema

4.3 PLATAFORMA DE SIMULAÇÃO *HARDWARE-IN-THE LOOP*

4.3.1 Integração com o controlador real

Para a integração com o controlador real o sistema proposto na plataforma 2 teve algumas adequações. Como pode ser visto na Figura 15, o controlador passa a ser um dispositivo DSP TMS320F28335 da Texas Instruments.

Ao migrar o controlador para o DSP foi constatado problemas devido a metaestabilidade. As entradas digitais do FPGA, blocos de elementos lógicos, blocos multiplicadores, têm em sua composição elementos flip-flop. Um flip-flop entra em estado metaestável quando não se respeita seus tempos de *setup* e *hold.* Em termos práticos, ocorre quando uma entrada é assíncrona ou quando se está interfaceando dois domínios que operam em diferentes frequências ou na mesma frequência, mas em diferentes fases, que ocasionam a violação dos tempos.

Um flip-flop em estado metaestável tem sua tensão de saída com valor diferente do nível 0 ou 1, ou seja, o seu valor de saída é imprevisível. Como a saída de um flip-flop está ligada a outros componentes, como portas lógicas e outros flip-flop, este valor incomum de tensão poderá causar valores estranhos de saída em outros componentes prejudicando a informação que se deseja t ransmitir. O autor (Vahid 2008) descreve que a metaestabilidade é basicamente um problema que ocorre quando um flip-flop tem entradas que não estão sincronizadas com o *clock* do circuito; sendo estas entradas ditas assíncronas; e desta forma quando se inicia o ciclo do *clock* os valores de saída ainda não se encontram em níveis estáveis propagando erros de dados para os outros dispositivos.

Como na plataforma 1 o sinal de *clock* que comandava os módulos das duas placas era o mesmo, os dispositivos já se encontravam em sincronia, desta forma, estava garantida a leitura da variáveis de simulação. Entretanto, com o DSP a fonte de *clock* dos dispositivos é distinta.

Usualmente, o modo utilizado para solucionar o problema é sincronizar a entrada assíncrona (DSP) com o seu relógio (*clock* do FPGA) antes que essa entrada se propague até os componentes do circuito. Este sincronismo foi feito com base nos sinais de PWM vindos do DSP. O módulo “Divisor de Frequência” (Figura 15), recebe o sinal PWM da chave “q3” e faz a contagem de tempo com base na borda de subida deste sinal. Esta ação sincroniza a borda de subida dos dois dispositivos. Porém, na leitura da borda de descida dos PWM o problema continuava.

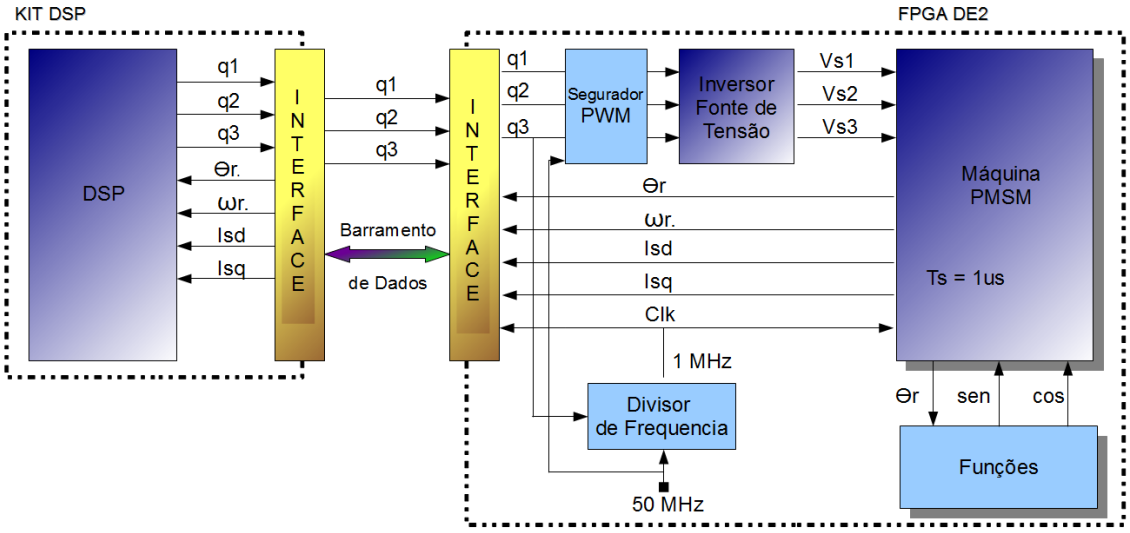


Figura 15. Diagrama representativo da Plataforma HIL

Por esse motivo foi necessário inserir um segurador de estado nos sinais PWM vindos do DSP. Como o *clock* das duas interfaces divergem em frequência – DSP com 150MHz e FPGA 50MHz – a borda de descida dos sinais PWM, eventualmente, coincidem com a proximidade borda de subida do *clock* do módulo da "Máquina PMSM", deixando a entrada metaestável exatamente no ponto de coleta da informação. Neste momento, ocorrem erros de leitura que induzem à instabilidade da simulação. Como o sincronismo é feito a partir da borda de subida e não há como prever quando os sinais PWM irão do estado 1 para 0, as informações destes sinais somente são validados a partir da análise de uma janela de dados amostrados; que foi chamado de "Segurador PWM".

O segurador consiste em realizar 50 amostras dos sinais PWM, nesta janela de dados o estado que mais se repetir é repassado ao "Inversor Fonte de Tensão".

Interface

A interface para a troca de informações entre o DSP e a placa DE-2 ocorre de forma similar à encontrada na plataforma 1. Entretanto, ao invés de quatro variáveis, são sete variáveis amostradas. Foram incluídas as tensões v_{sd} e v_{sq} médias e o conjugado eletromagnético na amostragem. Mesmo sendo as variáveis de controle amostradas de forma digital, o controlador DSP as obtém de modo análogo às variáveis analógicas (através de interrupções programadas e tempo definido) e, sendo assim, do ponto de vista de controle, não

há distinção, as variáveis podem ser consideradas como de um sistema real, obtidas através de conversores A/D.

Monitoramento das Grandezas

Para o monitoramento das grandezas foram utilizados os recursos disponíveis pela placa DSP e pelo *software* CCStudio v3.3. A memória interna da placa DSP, capaz de armazenar 114688 palavras de 16 bits, foi dividida em sete vetores inteiros de 16 bits com 16384 posições. O armazenamento dos dados pode ser realizado pelos múltiplos do intervalo de amostragem (100µs), desta forma, consegue-se armazenar períodos de simulação maiores ou iguais que 1,6384s. Os resultados obtidos podem ser exportados para posterior análise.

4.4 RESULTADOS DE SIMULAÇÃO PARA A PLATAFORMA HIL

Para a verificação da plataforma foram realizadas simulações em malha fechada de acordo com a estrutura do diagrama da Figura 6, onde as malhas de controle e o PWM são executados em um controlador real baseado em DSP. O conversor de tensão e o modelo da máquina PSMSM são simulados no FPGA em tempo real com *Hardware-in-the Loop*. Os parâmetros dos controladores e do conversor de potência são mostrados na Tabela 9. O objetivo da simulação é verificar se os componentes da plataforma operam de modo adequado.

Tabela 9. Parâmetros dos controladores e conversor de potência para a simulação da plataforma HiL.

Parâmetro	**Valor**
Largura de faixa do controlador de corrente (*d* e *q*)	250 Hz
Largura de faixa do controlador de velocidade (PI)	20 Hz
Tensão do barramento CC	300 V
Período de amostragem	100µs
Frequência de chaveamento	10 kHz

A simulação consiste em partir o motor e ao alcançar o regime permanente, aplica-se uma carga ao eixo da máquina. A velocidade mecânica de referência é de 125 rad/s. A carga de 1N.m é aplicada no instante de tempo de 0.325s. As características da simulação executada na plataforma HIL foram replicadas no ambiente do PSIM para a comparação de resultados. O passo de cálculo da simulação foi de 1µs para ambos os ambientes e a amostragem dos sinais foi realizada a cada 100µs. Nas figuras foram apresentados 0,7s de simulação.

A Figura 16 mostra o comportamento da velocidade mecânica executada nos dois ambientes. Na Figura 17 é mostrada a comparação do ângulo elétrico na faixa de tempo que compreende a aplicação da carga. A Figura 18 e a Figura 19 mostram as correntes i_{sq} e i_{sd} respectivamente.

Como as duas simulações são executadas sob os mesmos quesitos: mesma sintonia dos controladores, mesmas características do motor características do motor e conversor em condições ideais; espera-se que o comportamento das variáveis sejam próximo. Analisando as figuras, é possível perceber que essa correlação existe. Portanto, o objetivo foi alcançado, a troca de informações entre os dispositivos ocorre de forma correta permitindo que sistemas de controle sejam testados.

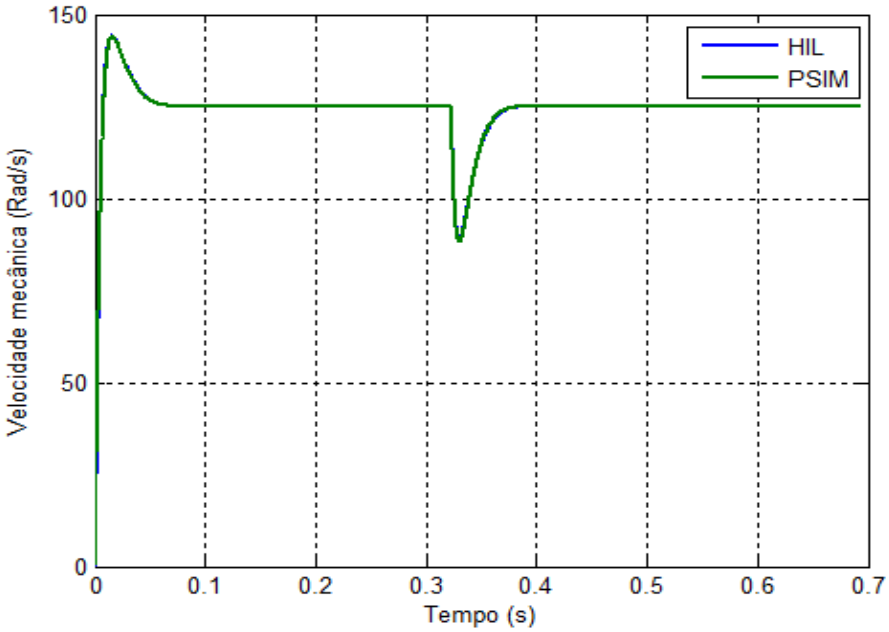


Figura 16. Comparação da Velocidade entre a Simulação HIL e o PSIM

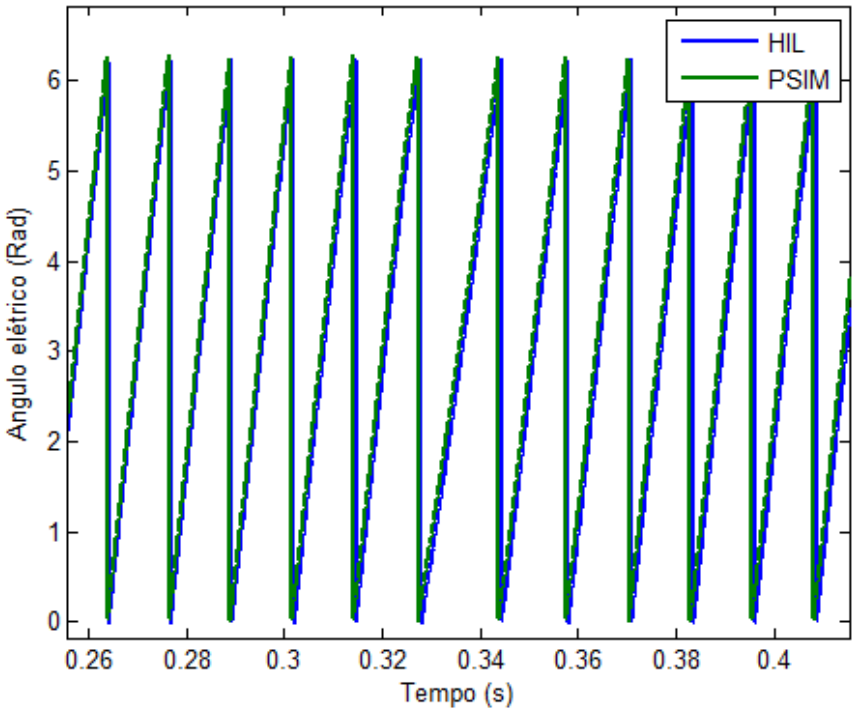


Figura 17. Comparação do angulo elétrico entre a Simulação HIL e o PSIM no instante de aplicação de carga (t = 0,325s).

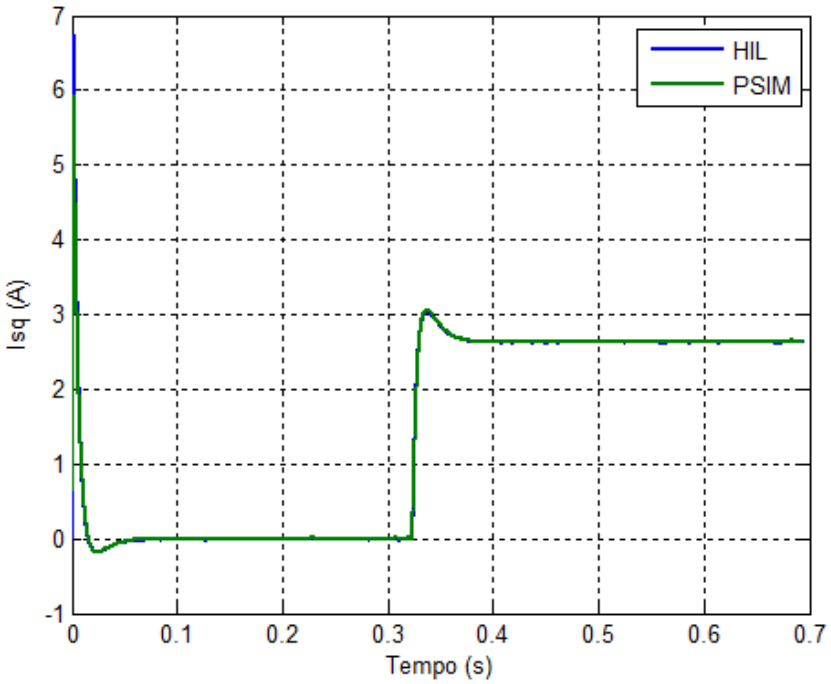


Figura 18.Comparação da Corrente i_{sq} entre a Simulação HIL e o PSIM.

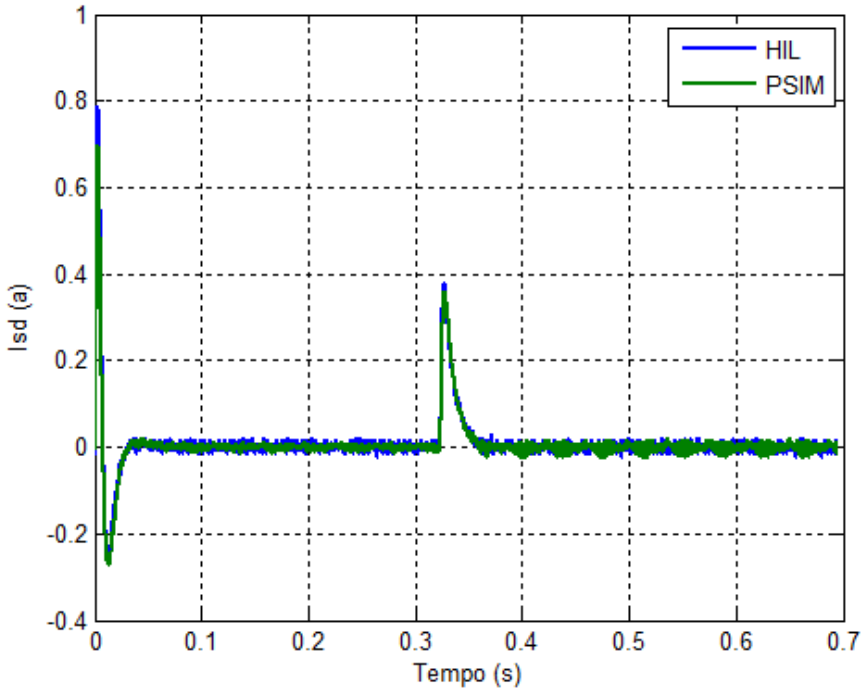


Figura 19. Comparação da Corrente i_{sd} entre a Simulação HIL e o PSIM.

4.5 CONCLUSÕES

O capítulo apresentou o processo de desenvolvimento da plataforma HIL. Foram descritas duas plataformas do processo de implementação com o objetivo de testar separadamente partes especificas do sistema final.

A plataforma 1 de simulação conta com um sistema *Hardware-in-the Loop* emulado. Uma segunda placa FPGA (DE-0) foi utilizada como controlador da máquina e proporcionou o desenvolvimento da interface entre controlador e planta simulada. Os dois dispositivos (DE-0 e DE-2) são comandados pelo mesmo pulso de *clock*, permitindo algumas flexibilidades para a realização dos testes. Foi possível reduzir a frequência do *clock* e manter as mesmas características de simulação, a fim de, com o sistema em execução, visualizar passo a passo o comportamento das variáveis e os bits que trafegam na interface dos dispositivos, o que não é possível quando executado em tempo real. Essa monitoração do comportamento das variáveis e da interface entre as placas FPGA permite a identificação rápida de problemas e correção, tais como, o correto procedimento da transmissão de dados, ordem das variáveis transmitidas, se os dados transmitidos e recebidos correspondem ao esperado.

Ainda no desenvolvimento da plataforma 1, a interface Homem-Máquina foi aprimorada e permite acompanhar e interferir na simulação em tempo real. Essa IHM também está disponível para a plataforma HIL.

Com os módulos da plataforma testados, houve a integração com o controlador real baseado em DSP, caracterizando o sistema HIL proposto. Foram realizadas simulações em malha fechada onde os resultados apresentados foram comparados com os resultados do *software* PSIM. Os resultados demonstram a equivalência entre as duas simulações, a partir disto, conclui-se que a plataforma opera da forma desejada.

Capítulo 5

Implementação da SMPM no Simulador HiL

Após o desenvolvimento da plataforma, buscou-se aprimorar o método para a implementação de uma máquina no simulador de tempo real. Com isso, os passos de implementação da máquina SMPM foram reestruturados e também replicados para a IPM. De um modo geral, a implementação é feita em duas fases: simulação como gerador; e simulação como motor.

Dentro de cada fase ocorrem atividades específicas que vão desde a avaliação inicial do modelo sendo executado em *software*, simulador de *hardware* até sua implementação no FPGA. Estas atividades serão descritas na sequencia, sendo aplicadas em cada máquina.

Nesta etapa do projeto, o FPGA utilizado contou com a placa de desenvolvimento Altera DE2-115 com o dispositivo FPGA Cyclone IV E FPGA com 114.480 elementos lógicos. O *layout* e dispositivos encontrados nesta placa são praticamente idênticos à placa anterior, assim, a migração dos módulos em Verilog desenvolvidos para o módulo anterior ocorreu sem problemas.

5.1 INTERAÇÃO DO MODELO DA MÁQUINA COM O SISTEMA

As equações da máquina são resolvidas no referencial síncrono do rotor dq. Entretanto, o sistema de potência é calculado a partir do referencial trifásico abc. Deste modo, é necessário interfacear o modelo da máquina com o restante do sistema para adequar as referencias e garantir o correto fluxo de dados com o controlador e o sistema de potência.

Como discutido no Capítulo 4, todo o sistema de equações é calculado em um único ciclo de *clock*. A cada pulso de *clock* são calculadas as tensões trifásicas de acordo com o estado das chaves PWM, e estes valores de tensão são utilizados nos cálculos do modelo da máquina.

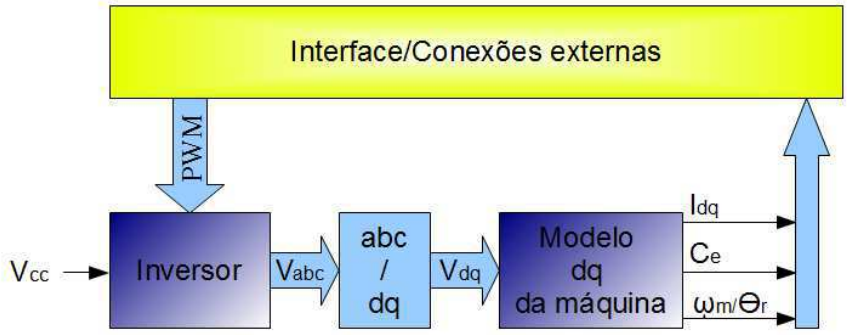


Figura 20. Interação do modelo da máquina com o restante do sistema.

A Figura 20 ilustra como é o processo de interação do modelo da máquina com o controlador e o sistema de potência. A solução segue os seguintes passos:

- O modelo do inversor recebe o estado das chaves PWM, recebidas do controlador;
- Através de um valor escolhido para a tensão do barramento cc (Vcc) são calculadas as tensões no referencial abc.
- As tensões de fase são transformadas do referencial abc para o dq.
- O modelo dq da máquina é calculado para se obter as correntes em dq e o equacionamento mecânico.
- As correntes dq e as variáveis mecânicas (C_e, ω_m e θ_r) calculadas são disponibilizadas no módulo de Interface (FPGA/DSP) para amostragem.

Como o sistema opera exclusivamente de modo digital, por conveniência, optou-se em disponibilizar as correntes dq para a amostragem. Visto que, uma transformação para o referencial trifásico aumenta o uso de recursos do FPGA para transmitir um mesmo teor de informação. Entretanto, pode-se transformar os valores de corrente para outro referencial e disponibilizá-los na interface para a amostragem de acordo com a necessidade da aplicação.

A Figura 21 mostra a representação funcional do bloco que implementa as equações. As entradas do bloco correspondem às chaves PWM e as saídas correspondem às variáveis da máquina que são calculadas a cada borda de subida do sinal de *clock* (Clk).

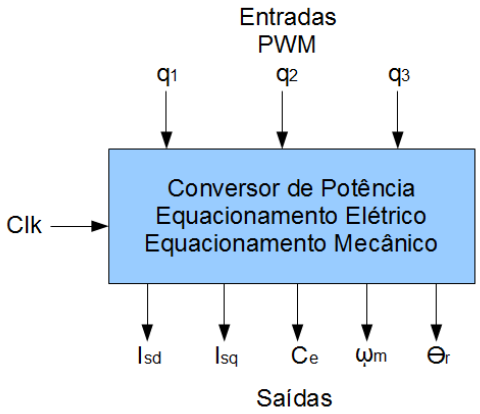


Figura 21. Representação do bloco com as equações do sistema.

O método de interação utilizado é simples. Porém, todos os cálculos são realizados sequencialmente dentro de um único bloco (em Verilog denominado bloco *always*) a cada pulso de *clock*. Como frequência do *clock* corresponde ao passo de cálculo do sistema, deve-se atentar para que o passo de cálculo seja maior que o tempo necessário para o FPGA resolver o sistema de equações. Portanto:

- Após cada alteração na descrição de *hardware*, ao compilar, deve-se verificar a indicação da frequência máxima das vias de *clock* obtidas pela ferramenta de análise de tempo do Quartus II (*TimeQuest Timing Analizer*). A frequência máxima indica o tempo de cálculo.
- Caso o tempo necessário para a solução das equações seja maior que o passo de cálculo, deve-se aumentar o tempo do passo de cálculo.
- Se este aumento do passo de cálculo ocasionar em perda de precisão, pode-se utilizar outro método que separa as equações do modelo da máquina, equacionamento mecânico e sistema de potência para blocos diferentes e assim dividir o tempo de cálculo.

Os detalhes do método que separa em blocos o equacionamento da máquina estão apresentados no Apêndice C.

5.2 MÁQUINA OPERANDO COMO GERADOR

Como dito anteriormente, o primeiro passo para a implementação da SMPM é a simulação da máquina como gerador. Os dados da máquina apresentada nesta seção estão listados na Tabela 10. Simular primeiro a máquina como gerador permite avaliar o

equacionamento da máquina de forma gradativa, analisando um conjunto reduzido de equações por vez. O objetivo da simulação da máquina como gerador é avaliar os seguintes perfis:

1) perfil da força contra-eletromotriz;

2) perfil da corrente da máquina sob carga.

Tabela 10 Servomotor WEG SWA 56-2,5-60

Potência Nominal	1,13 kW
Rs	0,7465 Ω
Lq	2,54 mH
Ld	2,28 mH
λ_{pm}	0,068 Wb
Velocidade	6000 rpm
Corrente Nominal	7,2 A
Inércia	0,00022 Kg.m²
Pares de Pólos	4

O equacionamento de corrente da máquina operando como gerador no modelo *dq* segue o mesmo princípio da máquina operando como motor. A operação da máquina como motor pode ser descrita a partir deste equacionamento:

Equacionamento para o eixo d:

A equação (4.2) pode ser reescrita da seguinte forma (5.1):

$$I_{sd} = \frac{V_{sd} + \omega_r I_{sq} L_q + E_d}{sL_d + R_s} \tag{5.1}$$

$$E_d = 0 \tag{5.3}$$

onde E_d é o termo da fcem no eixo d, que nesta máquina tem valor nulo e $R_s = r_s$.

Aplicando a equação (4.14), substituindo T por h, na equação (5.1), chega-se na seguinte representação discreta:

$$I_{sd} = \frac{I_{sd}(L_d - R_s h) + h(V_{sd} + \omega_r I_{sq} L_q + E_d)}{L_d} \tag{5.4}$$

Equacionamento para o eixo q:

Para o eixo q a equação de corrente (4.7) pode ser reescrita como em (5.5):

$$I_{sq} = \frac{V_{sq} - \omega_r I_{sd} L_d - E_q}{sL_q + R_s} \tag{5.5}$$

$$E_q = \omega_r \lambda_{pm} \tag{5.6}$$

onde E_q é o termo da fcem no eixo q. Aplicando o método de Euller, chega-se na sua equivalente discreta:

$$I_{sq} = \frac{I_{sq}\left(L_q - R_s h\right) + h(V_{sq} - \omega_r I_{sd} L_d - E_q)}{L_q} \tag{5.7}$$

O equacionamento para o conjugado eletromagnético pode ser dado a partir da equação (3.24) e pode ser reescrita da seguinte forma:

$$C_e = P\{\lambda_{pm} I_{sq} + (L_d - L_q) I_{sd} I_{sq}\} \tag{5.8}$$

A equação do movimento pode ser obtida se aplicando Euller na equação (4.12):

$$\omega_m = \omega_m + \frac{h(C_e - f_\omega \omega_m - C_m)}{J} \tag{5.9}$$

Inicialmente, para o equacionamento em ponto fixo, optou-se em representar todo o sistema com um formato de Q-27 tanto para as variáveis como para as constantes, exceto o passo de cálculo (h = 1µs) com formato Q-40. Entretanto, nas primeiras simulações, observou-se que alterações nos formatos das constantes poderiam melhorar o resultado final. Por este motivo, foi adotado um método descrito na sequência para se minimizar os erros.

Para representar as constantes da máquina em ponto fixo, buscou-se evitar os chamados *Erros iniciais*, que são cometidos nos arredondamentos de constantes e dados iniciais. Através da seguinte definição:

- É dito que $\bar{x}$ tem *k* casas decimais corretas se: $\Delta_{\bar{x}} = |x - \bar{x}| \leq 0{,}5 \times 10^{-k}$;
- É dito que $\bar{x}$ tem *n* algarismos significativos corretos se: $r_{\bar{x}} \leq 0{,}5 \times 10^{-n}$.

onde $\bar{x}$ é uma aproximação de *x,* $\Delta_{\bar{x}}$ é o erro absoluto e $r_{\bar{x}}$ o erro relativo.

É importante também saber o número de dígitos significativos do sistema de representação que está sendo utilizada para que se tenha a noção da precisão do resultado obtido.

A representação base do sistema é de Q-27, que oferece uma precisão de $2 \times 10^{-27} = 7{,}45 \times 10^{-9}$. Seguindo o conceito, o número de algarismos significativos do sistema que devem ser considerados são de 7 casas decimais ($7{,}45 \times 10^{-9} \leq 0{,}5 \times 10^{-7}$) para a validação do método.

Após a análise dos erros, percebeu-se que seria necessária a adequação de várias constantes, sendo necessário aplicar valores maiores. A Tabela 11 mostra os valores das constantes da máquina e sua aproximação em ponto fixo. A Tabela 12 mostra a relação do erro absoluto com o número de casas decimais corretas e o erro relativo com o número de algarismos significativos corretos.

Percebe-se pelos valores de *n* na Tabela 12 que as constantes estão dentro do número de algarismos significativos esperados ($n \geq 7$).

Tabela 11. Valores das constantes da máquina SMPM e se valor aproximado em ponto fixo.

Constante	x	$\bar{x}$	Representação
Rs (Ω)	0,7465	0,74650000035	Q-27
Lq (H)	0,00254	0,002540000016	Q-32
Ld (H)	0,00228	0,002279999898	Q-32
λ_{pm} (Wb)	0,068	0,067999999970	Q-28
Inércia(Kg.m²)	0,00022	$2{,}20000001718 \times 10^{-4}$	Q-36

Tabela 12. Analise do erro absoluto e erro relativo para a validação da representação das constantes da máquina SMPM.

Constante	$\Delta_{\bar{x}}$	k	$r_{\bar{x}}$	n
Rs (Ω)	$3{,}576 \times 10^{-10}$	9	4.790×10^{-10}	9
Lq (H)	$1{,}586 \times 10^{-11}$	10	$6{,}247 \times 10^{-9}$	7
Ld (H)	$1{,}012 \times 10^{-10}$	9	$4{,}440 \times 10^{-8}$	7
λ_{pm} (Wb)	$2{,}980 \times 10^{-11}$	10	$4{,}382 \times 10^{-10}$	9
Inércia(Kg.m²)	$1{,}718 \times 10^{-12}$	11	$7{,}810 \times 10^{-9}$	7

Os registradores em Verilog foram dimensionados em ponto fixo com 45 bits com sinal, o que seria equivalente aos 27 bits para a parte fracionária e 18 bits como reserva para a parte inteira.

O angulo elétrico em Verilog quando calculado é convertido para pu levando em consideração o numero de bits do seu registrador. Desta forma, a faixa de valores do registrador que o armazena corresponde a faixa de valores compreendidos entre 0 e 2π. Quando o angulo atinge o valor máximo do registrador o estouro o leva de volta ao valor de θ_r-2π, se o ângulo tender a um valor negativo o resultado será de θ_r+2π. Assim, o valor ângulo fica condicionado ao tamanho do registrador. Através das simulações, comparando os modelos em ponto fixo e em ponto flutuante com precisão dupla, percebeu-se que quanto maior o tamanho do registrador do ângulo, mais próximos são os resultados. Para esta representação, verificou-se que 31 bits são suficientes para uma aproximação satisfatória.

5.2.1 Perfil da força contra-eletromotriz:

Para validar o modelo as grandezas simuladas foram comparadas com resultados experimentais da máquina. Os valores experimentais foram obtidos no trabalho de (E. d. Fernandes 2006) onde a máquina foi acionada como gerador por uma máquina auxiliar a 900 rpm onde foram medidas as forças eletromotrizes geradas pela máquina.

A Figura 22-(a) mostra o perfil da tensão induzida no referencial αβ medidas experimentalmente e calculadas. A Figura 22-(b) apresenta as tensões induzidas medidas no referencial dq, e a Figura 22-(c) demonstra o conteúdo harmônico presente nas tensões induzidas no referencial dq. Através da informação presente na FFT, foi possível calcular as tensões induzidas em αβ a parir do equacionamento em dq abaixo:

$$E_d = -\omega_r \lambda_{pm} \sum_{n=1}^{50} ked_n \sin\left(n\theta_r + \theta_{ked_n}\right) \tag{5.10}$$

$$E_q = \omega_r \left(\lambda_{pm} - \lambda_{pm} \sum_{n=1}^{50} keq_n \sin\left(n\theta_r + \theta_{keq_n}\right) \right) \tag{5.11}$$

Onde ked_n e keq_n são as constantes harmônicas de eixo d e q respectivamente; e θ_{ked_n} e θ_{keq_n} os ângulos de fase das respectivas constantes.

Apesar da correção dada pelas equações (5.10) e (5.11) melhorarem o modelo, optou-se em utilizar equacionamento genérico para a força eletromotriz no referencial dq pelas equações (5.3) e (5.6). Visto que, como será demonstrado na sequencia, as equações representam satisfatoriamente este modelo da máquina.

O modelo inicial foi simulado em *software* e transcrito para Verilog para ser testado na plataforma HIL. A Tabela 13 apresenta a quantidade de recursos utilizado no FPGA para simular o perfil da fcem da máquina. Como não houve a necessidade de se utilizar seno e cosseno, os bits de memória não foram ocupados. A conversão da fcem no referencial dq para o modelo trifásico foi executado pelo DSP. Com isso, os módulos "Inversor Fonte de Tensão", "Funções" e "Filtro PWM" constantes na Figura 15 não foram utilizados. A ferramenta Altera Quartus II *Timing analyzer* informou que este equacionamento implementado pode ser executado em 15,57 ns.

Tabela 13. Consumo de recursos do FPGA para simular a fcem do SMPM

Descrição	Necessário	Disponível	(%)
Elementos lógicos (LEs)	125	114480	<1
Bits de memória	0	3981312	0
Multiplicadores 9bits	4	532	<1

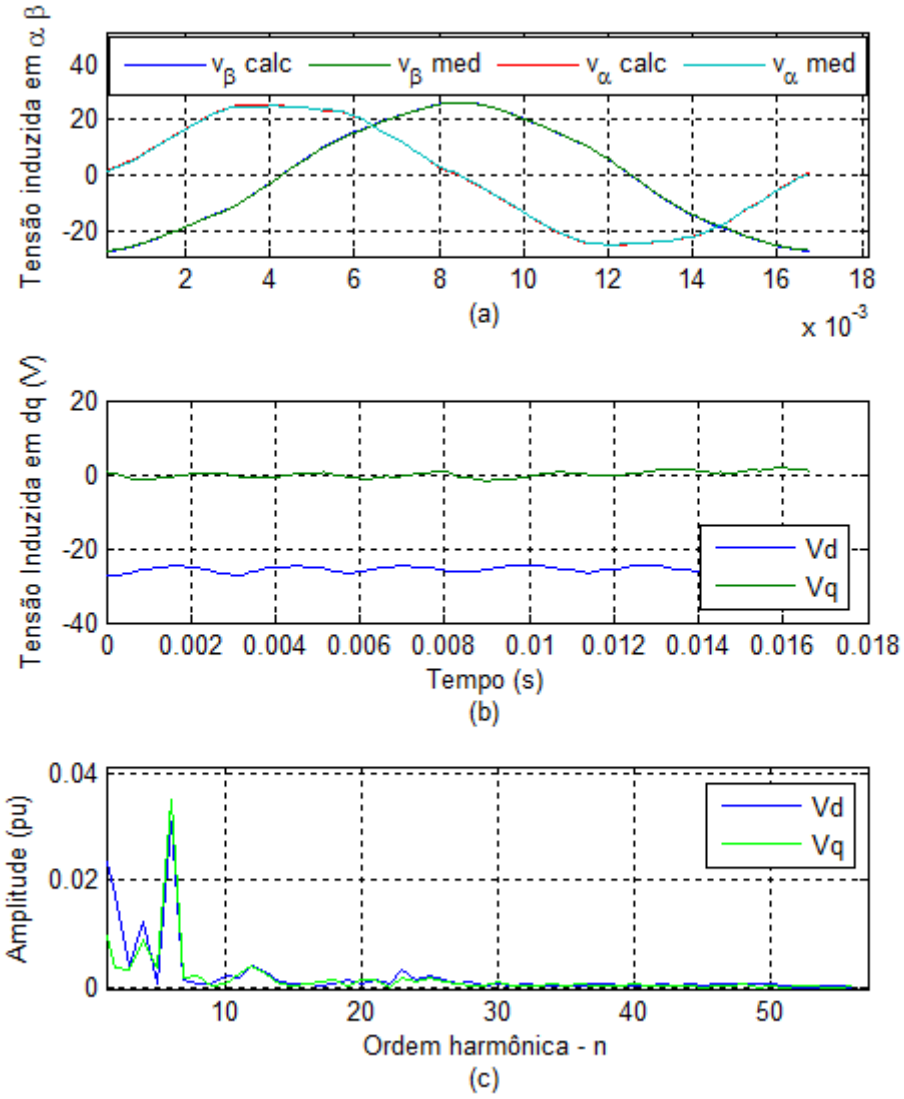


Figura 22. Perfil da Tensão induzida da SMPM: (a) Tensão induzida no referencial αβ medida e calculada; (b) Tensão induzida medida no referencial dq; (c) Espectro harmônico das tensões induzidas no referencial dq.

A Figura 23 apresenta a comparação dos dados obtidos pelo sistema HIL e a máquina da WEG, mostrando as curvas das fcem de linha e o erro instantâneo para v_α, enquanto que a Figura 24 apresenta para v_β.

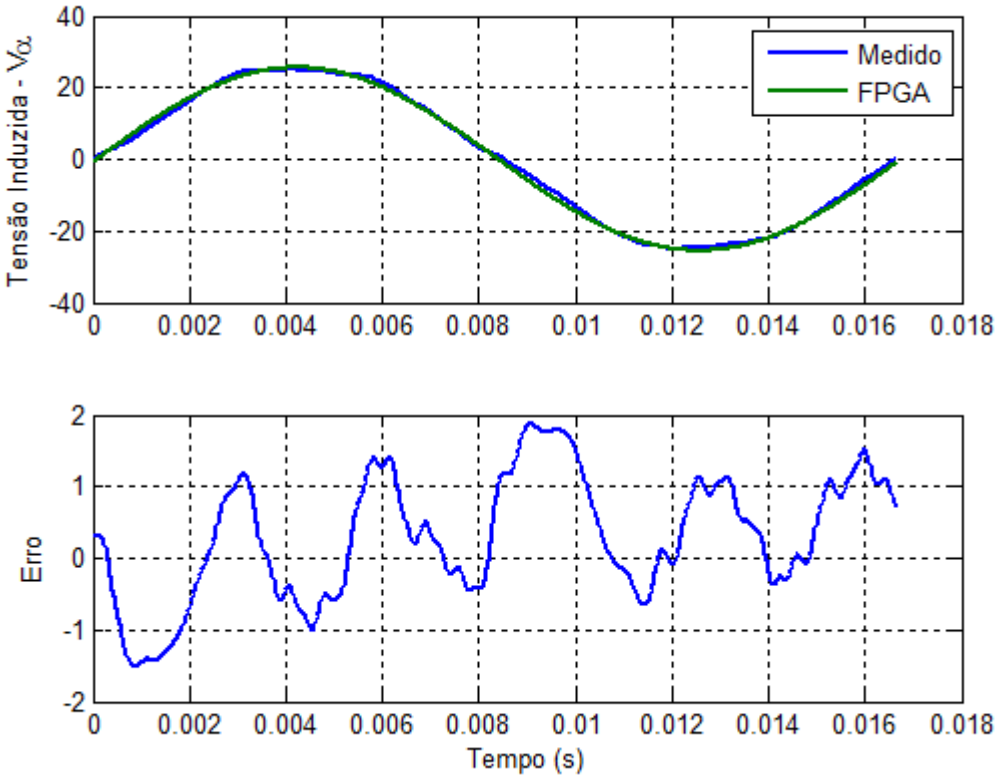


Figura 23. Perfil da tensão induzida $\boldsymbol{v_{s\alpha}}$: (acima) medido (WEG) e obtida pelo modelo em FPGA, (abaixo) Erro instantâneo entre a saída obtido pelo modelo FPGA e a medição em laboratório (WEG).

O critério para a avaliação adotado foi o erro médio quadrático entre o valor medido e o valor calculado pelo simulador. O erro médio quadrático é dado pela seguinte expressão:

$$e_{mq} = \frac{1}{N_a}\sum_{a=1}^{N_a}[v_{med}(a) - v_{calc}(a)]^2 \tag{5.12}$$

nas quais, a é a amostra instantânea, N_a o número de amostras, v_{med} o valor medido e v_{calc} o valor calculado pelo simulador.

Para os resultados da Figura 23, utilizou-se N_a= 688. O erro médio quadrático para a v_α foi de 0,303 $V^2/amostra$, representando 1,18% da amplitude do valor medido. Enquanto que v_β 0,3662 $V^2/amostra$, representando 1,39% da amplitude do valor medido.

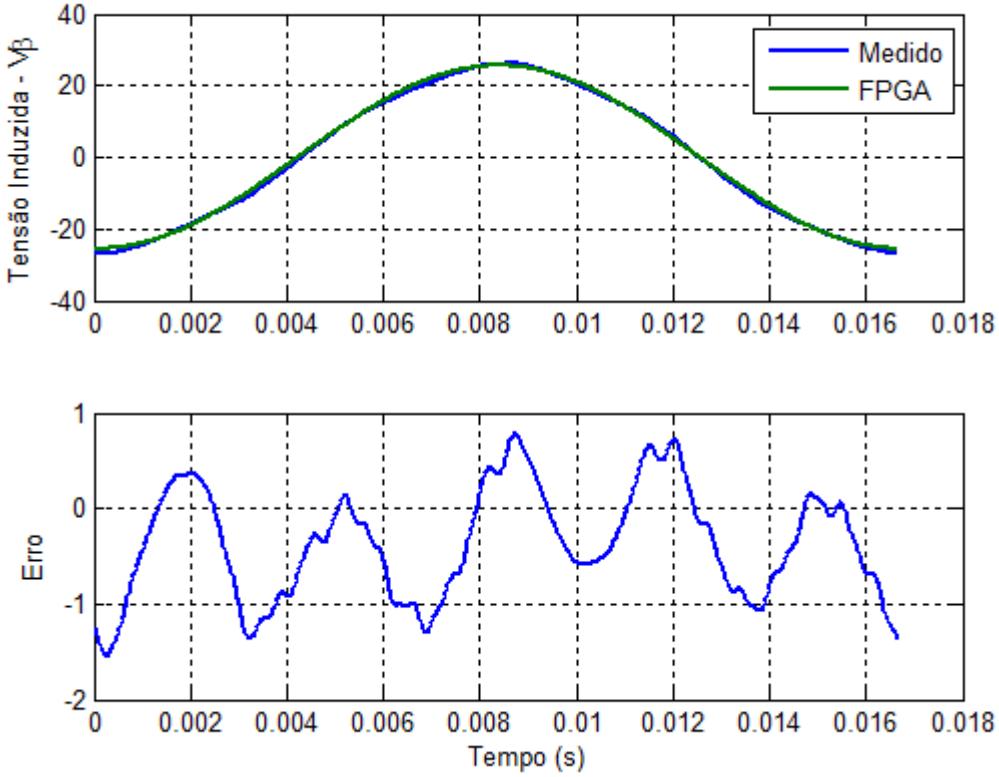


Figura 24. Perfil da tensão induzida $\boldsymbol{v_{s\beta}}$: (a) medido (WEG) e obtida pelo modelo em FPGA, (b) Erro instantâneo entre a saída obtido pelo modelo FPGA e a medição em laboratório (WEG).

5.2.2 Perfil da Corrente da Máquina sob Carga

A simulação do perfil da corrente já contém um número de equações significativos para serem simulados em FPGA. Para este caso, o modelo matemático passou pelos seguintes estágios na sequência:

- Simulação em ponto flutuante e ponto fixo em *software*;
- Simulação de *hardware* no ModelSim;
- Implementação no Quartus II e plataforma HiL.

Para os testes de simulação em ponto fixo e ponto flutuante foram executadas em simulações em *software* na linguagem JAVA. A aritmética em ponto fixo em JAVA tem sintaxe similar ao Verilog, o que facilita a transcrição. Desta forma, o esforço para adequar os códigos em Verilog é menor.

Para a simulação da máquina operando como gerador sob carga os termos das tensões de entrada foram anulados ($V_{sd} = V_{sq} = 0$). Para fins didáticos as equações de corrente (5.4) e (5.7) podem ser representadas da seguinte forma:

$$I_{sd} = \frac{I_{sd}(L_d - R_s h) + h(\omega_r I_{sq} L_q + E_d)}{L_d} \tag{5.13}$$

$$I_{sq} = \frac{I_{sq}\left(L_q - R_s h\right) + h(-\omega_r I_{sd} L_d - E_q)}{L_q} \tag{5.14}$$

onde $R_s = r_s + r_{carga}$.

Simulação de *hardware* no ModelSim**:**

O ModelSim é uma ferramenta disponibilizada pela Altera para a simulação interativa de projetos feitos para FPGA. É possível simular os módulos em linguagem de *hardware* passo a passo e acompanhar o comportamento de qualquer variável e visualizar graficamente sua forma de onda.

O modelo em ponto fixo obtido em *software* é transcrito para Verilog. Neste estágio os registradores do FPGA são dimensionados, onde se deve buscar valores que não comprometam as operações aritméticas por "estouro" das variáveis.

Em (Altera 2010) é sugerido simular os módulos Verilog no ModelSim antes de implementa-los no FPGA. O ModelSim é uma ferramenta que permite verificar se as variáveis foram dimensionadas de forma correta. A compilação é rápida se comparada a compilação do QUARTUS II e se ocorrer algum estouro o registrador é facilmente localizado. É possível monitorar todas as variáveis e averiguar a utilização bit a bit de cada registrador, identificando assim, a necessidade de se alterar alguma representação.

A Figura 25 mostra o ambiente ModelSim com a simulação da SMPM operando como gerador. Percebe-se o campo "Objects", que apresenta todas as variáveis envolvidas na simulação e permite acompanhar seu estado a cada novo passo de cálculo. A área "Wave" apresenta o comportamento gráfico das variáveis de interesse, que no caso foram I_{sd}, I_{sq} e θ_r. A partir destas curvas se pode verificar que o comportamento das grandezas está dentro do esperado.

A ferramenta permite que os dados dos gráficos sejam exportados, com isso os dados foram comparados com os valores em ponto fixo obtidos em JAVA. A Figura 26 mostra o resultado para a corrente I_{sq}, as outras grandezas tiveram o comportamento similar. Como é de se esperar, o resultado obtido entre o simulador em *hardware* e o JAVA foi exatamente o mesmo, com erro instantâneo nulo.

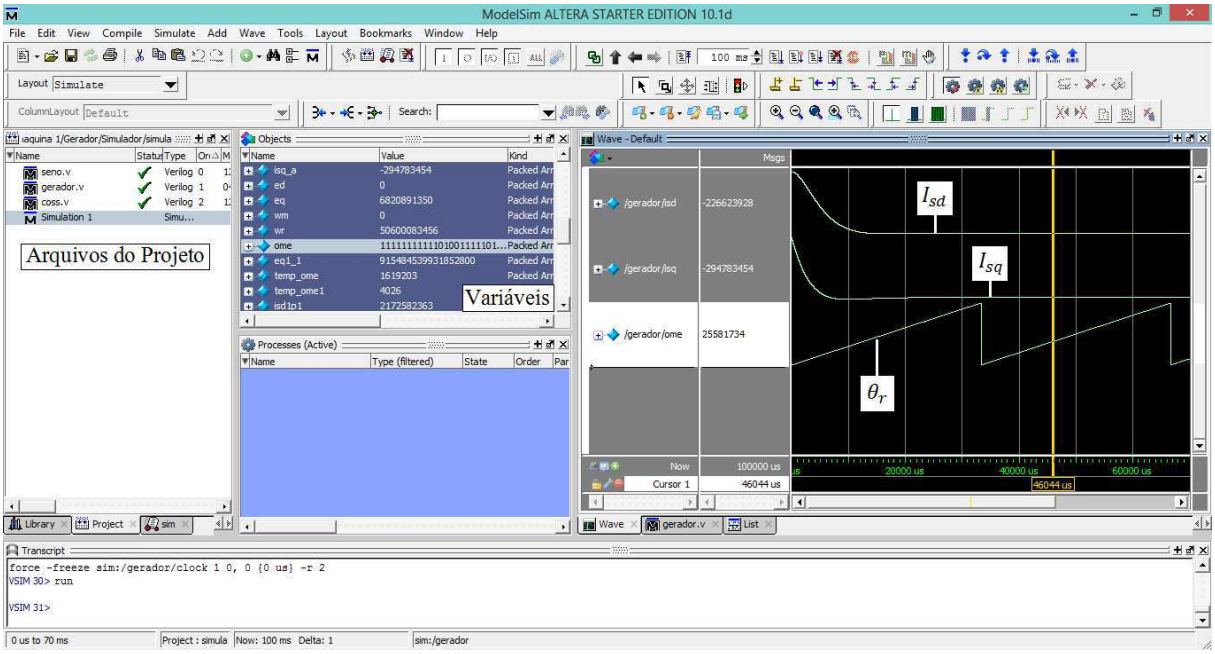


Figura 25. Simulação da SMPM como Gerador no ModelSim.

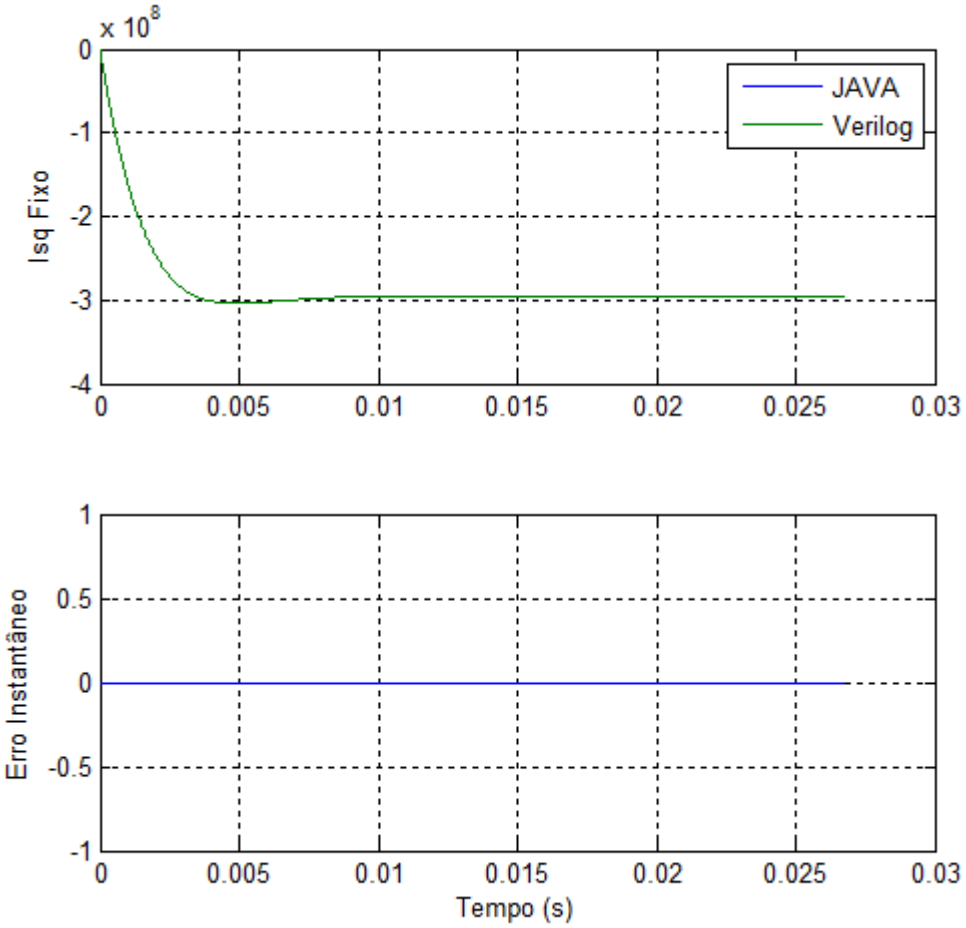


Figura 26. (a) Comparação da Corrente I_{sq} entre Verilog e JAVA; (b) Erro instantâneo entre as correntes I_{sq} obtidas em Verilog e JAVA.

Implementação no Quartus II e plataforma HIL:

O módulo em Verilog simulado no ModelSim foi adaptado para operar na plataforma HIL. A Tabela 14 apresenta o uso de recursos do FPGA para esta aplicação. Nota-se um aumento significativo do consumo dos Elementos lógicos e Multiplicadores em relação à fcem devido a utilização do equacionamento das correntes.

Neste caso, como na simulação da fcem, não foi necessária a utilização de senos e cossenos. Desta forma os bits de memória não foram ocupados. As conversões do referencial dq para o modelo trifásico que utilizam estas funções foram executadas pelo DSP.

Tabela 14. Consumo de recursos do FPGA Cyclone IV para simular a SMPM como gerador.

Descrição	**Necessário**	**Disponível**	**(%)**
Elementos lógicos (LEs)	6483	114480	6
Bits de memória	0	3981312	0
Multiplicadores 9bits	106	532	20

Após a compilação da máquina como gerador, o resultado da análise de tempo do QUARTUS II apontou que a frequência máxima que poderia ser aplicado ao *clock* do sistema é de 5,76 MHz, que representa a possibilidade de se utilizar um passo de cálculo de 0,17 µs.

A máquina foi simulada de acordo com os resultados experimentais obtidos por (E. d. Fernandes 2006). A máquina foi acionada por uma máquina auxiliar a 900 RPM e foi conectado aos seus terminais um banco resistivo de 10Ω/fase. Através deste procedimento obteve-se as correntes de carga $i_{s\alpha}$ e $i_{s\beta}$.

As figuras Figura 27 e Figura 28 apresentam os resultados dos valores das correntes obtidos experimentalmente e pelo FPGA. Os erros médios quadráticos para as correntes foram calculados a partir da equação (5.12). Para $i_{s\alpha}$ o erro foi de 0,00823 A²/amostra, representando 0,35% da amplitude medida. Para $i_{s\beta}$ o erro foi de 0,00608 A²/amostra, cerca de 0,24% da amplitude da corrente medida.

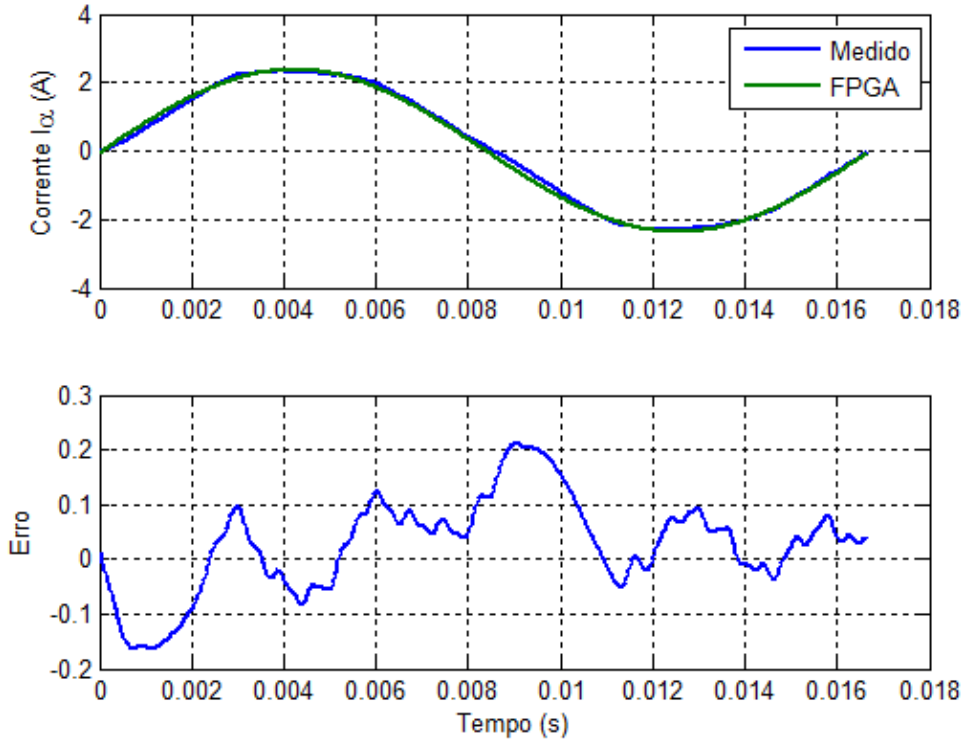


Figura 27. Comparação da corrente i_α da máquina SMPM operando como gerador.

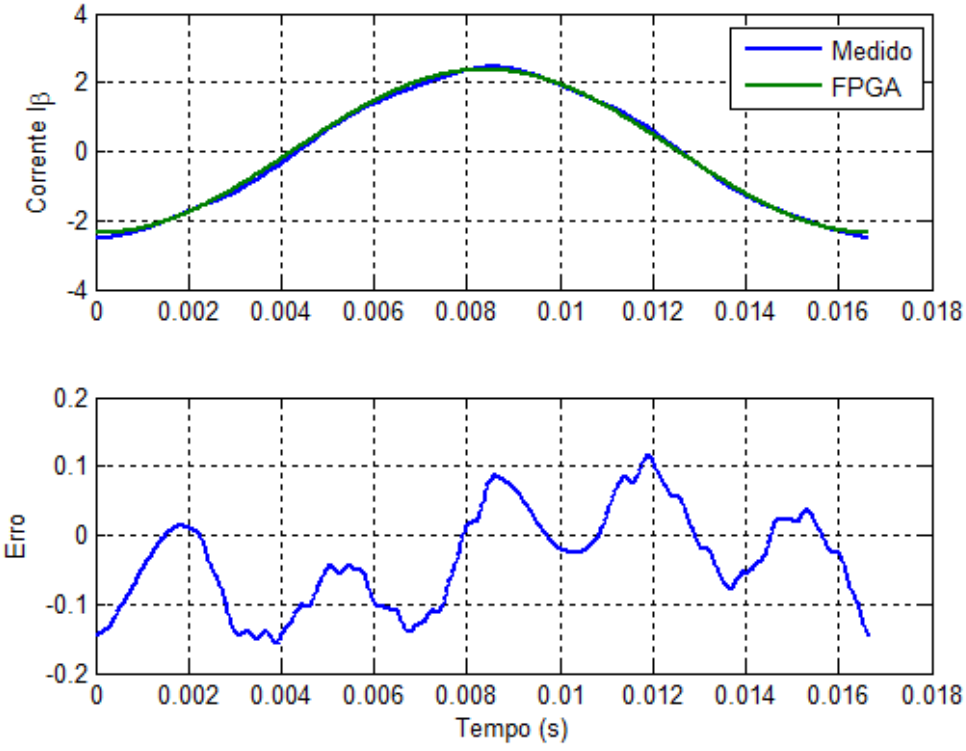


Figura 28. Comparação da corrente i_β da máquina SMPM operando como gerador

5.3 MÁQUINA OPERANDO COMO MOTOR

A implementação da máquina SMPM operando como gerador contém a base da simulação da máquina operando como motor. Nele está o equacionamento da fcem e das correntes I_{sd} e I_{sq}. Ou seja, o modelo matemático e o dimensionamento das variáveis garantem uma boa aproximação ao modelo de referência. Para a máquina operando como motor é compilado na simulação, além das equações de corrente (5.4) e (5.7), as equações de conjugado eletromagnético (5.8) e a equação de movimento (5.9).

Simulação em ponto flutuante, ponto fixo e no ModelSim:

Seguindo os passos da implementação da máquina como gerador, o modelo foi simulado em *software* para ponto flutuante e ponto fixo antes de ser simulado em Verilog no ModelSim. Pela equivalência entre o Verilog e ponto fixo, os resultados apresentados comparam o Verilog apenas com a simulação em ponto flutuante de precisão dupla.

A simulação é simples, não há controle de velocidade nem acionamento por inversor de tensão. O modelo matemático do motor é testado apenas se aplicando um valor de tensão nas correspondentes *dq,* com $V_{sd} = 0$ e $V_{sq} = 12\,V$. O objetivo é de se certificar que a resposta do modelo em Verilog corresponde ao modelo em ponto flutuante.

A Figura 29 mostra a comparação da velocidade elétrica do motor entre a simulação em ponto flutuante com a simulação realizada no ModelSim. A Figura 30 faz a mesma comparação para a corrente I_{sd}, e a Figura 31 para a corrente I_{sq}. É possível notar nas figuras a equivalência entre o modelo em Verilog e o modelo em ponto flutuante, e como o arredondamento inerente da aritmética de ponto fixo induz a uma pequena diferença de valor entre as grandezas. Para todas as grandezas o erro médio quadrático percentual é inferior a 0.00001%.

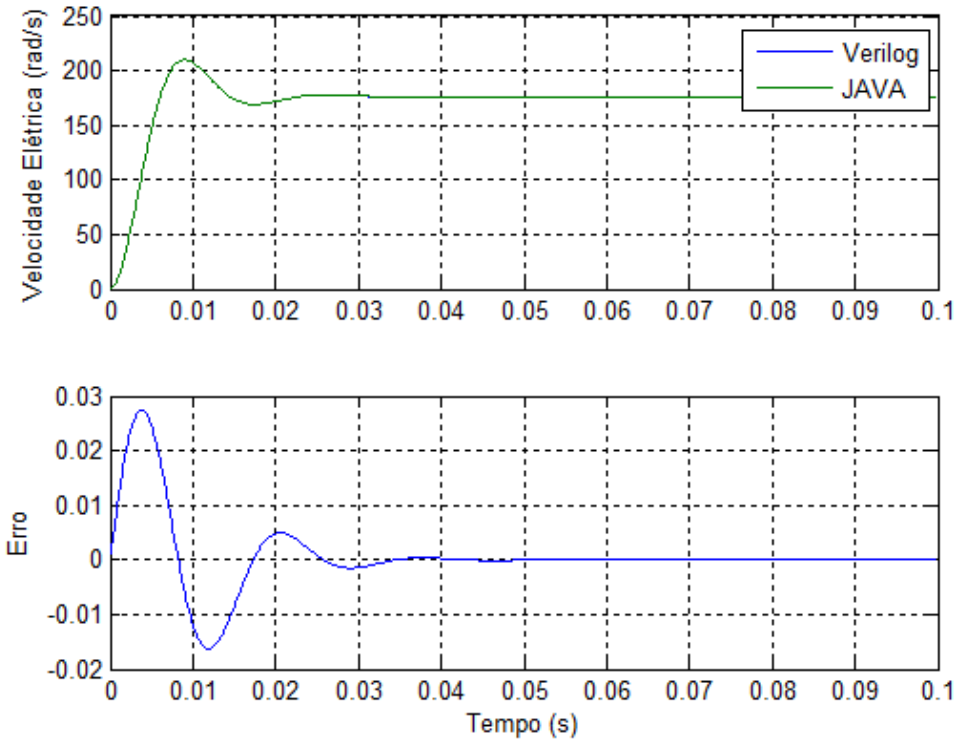


Figura 29. Comparação entre as velocidades obtidas pelos simuladores: (a) Verilog - linha azul, JAVA - linha verde; (b) Erro instantâneo entre as respostas Verilog e JAVA.

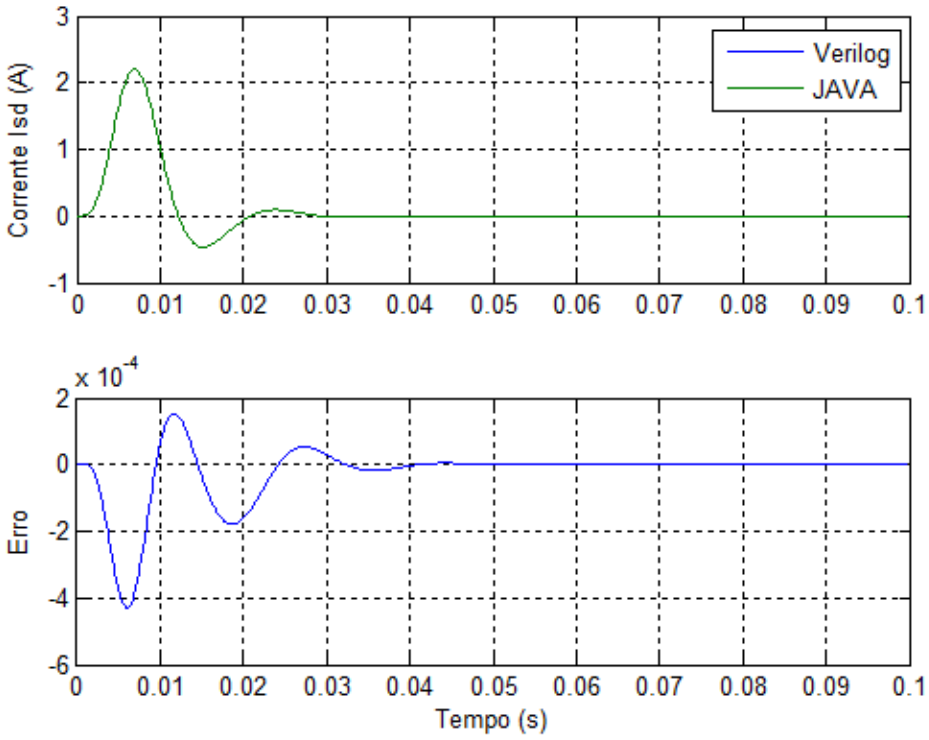


Figura 30. Comparação entre as correntes I_{sd} obtidas pelos simuladores: (a) Verilog - linha azul, JAVA - linha verde; (b) Erro instantâneo entre as respostas Verilog e JAVA.

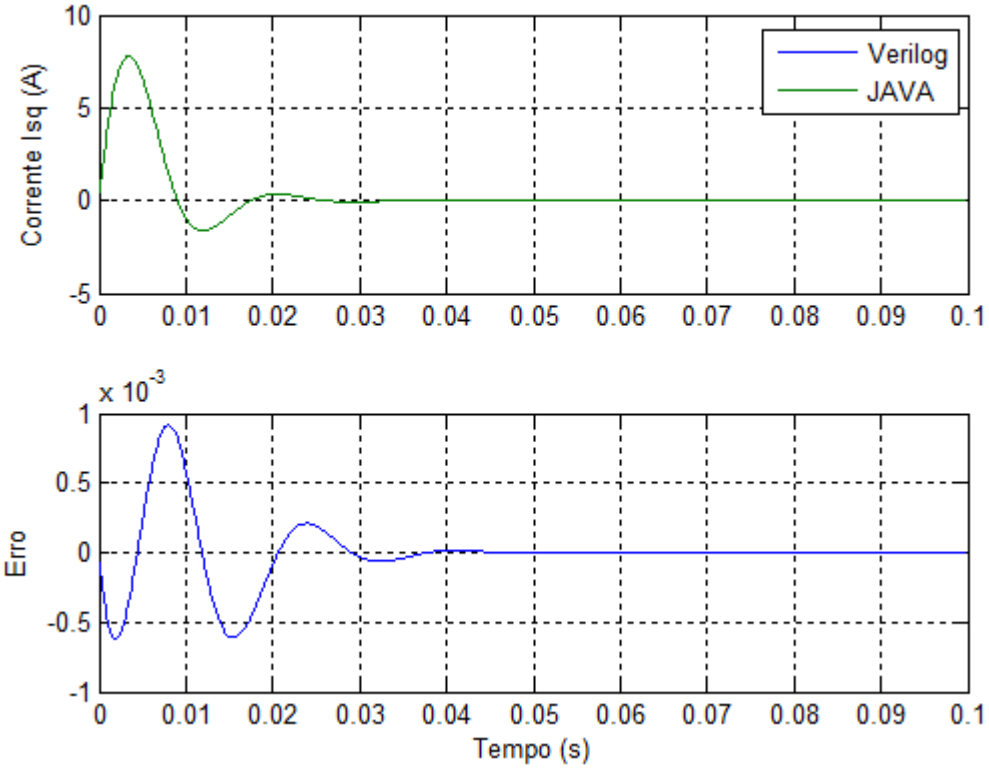


Figura 31 . Comparação entre as correntes I_{sq} obtidas pelos simuladores: (a) Verilog - linha azul, JAVA - linha verde; (b) Erro instantâneo entre as respostas Verilog e JAVA.

Simulação HIL do motor SMPM:

A simulação HiL para o motor SMPM é representado pela Figura 15, onde o DSP opera como controlador de velocidade e corrente da máquina simulada no FPGA. Nesta simulação todos os módulos representados na figura são utilizados. A Tabela 15 apresenta o consumo dos recursos do FPGA para simular a máquina como motor. Quanto aos requisitos de tempo, o módulo leva 0,42µs para ser calculado, permitindo um *clock* de 2,36 MHz.

Tabela 15.Consumo de recursos do FPGA Cyclone IV para simular a SMPM como Motor.

Descrição	**Necessário**	**Disponível**	**(%)**
Elementos lógicos (LEs)	12203	114480	11
Bits de memória	33120	3981312	<1
Multiplicadores 9bits	207	532	39

O esquema de controle está demonstrado na Figura 6. O controle segue a orientação pelo campo onde $i_{sd}^{*} = 0$ e $i_{sq}^{*} = \frac{T^{*}}{P\lambda_{pm}}$. A Tabela 16 apresenta os parâmetros de controle.

Tabela 16. Parâmetros dos controladores e conversor de potência para a simulação da plataforma HIL com a máquina SMPM.

Parâmetro	**Valor**
Largura de faixa do controlador de corrente (*d* e *q*)	250 Hz
Largura de faixa do controlador de velocidade (PI)	20 Hz
Tensão do barramento CC	300 V
Período de amostragem	100µs
Frequência de chaveamento	10 kHz

A forma de obtenção dos parâmetros dos controladores pode ser encontrada em apêndice. Na simulação o motor é acionado com uma velocidade de referência de 900 RPM. No período compreendido entre 0,75s e 1,2s é aplicada uma carga no seu eixo de 1 N.m. As figuras 32-33 mostram a resposta dinâmica do sistema de acionamento do motor. A simulação HIL é comparada com o modelo apresentado discretizado com Runge-Kutta de quarta ordem (E. d. Fernandes 2006).

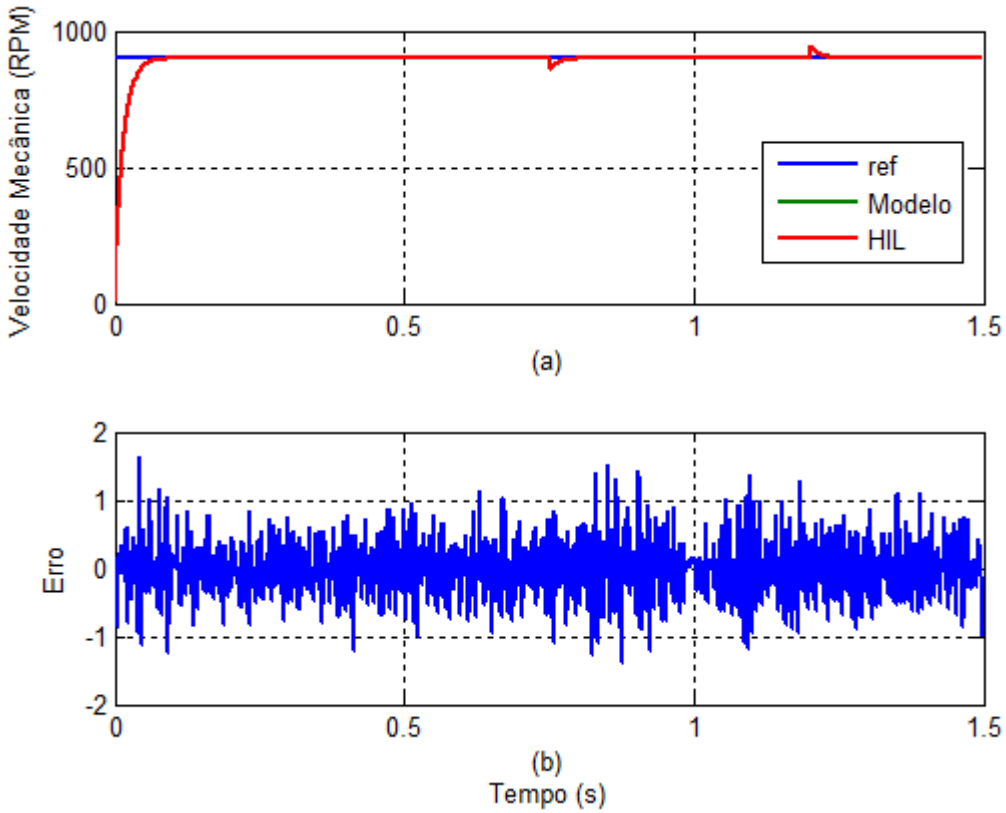


Figura 32. Comparação de Velocidade do motor SMPM: (a) Velocidades do Modelo e HIL a uma Referencia de 900 rpm; (b) Erro Instantâneo entre o Sistema HIL e o Modelo.

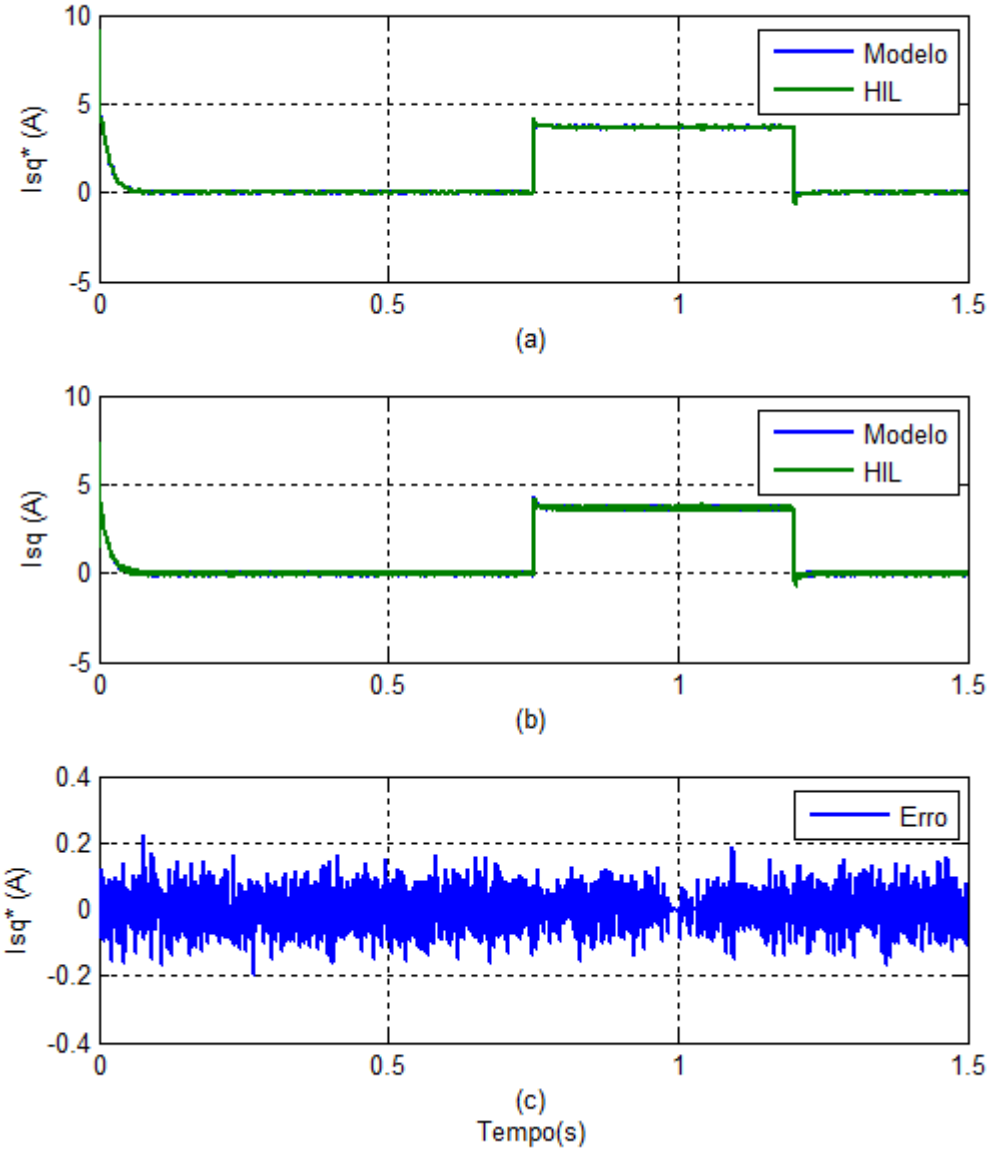


Figura 33. Comparação das correntes I_{sq}: (a) Correntes de referencia I_{sq}^{*}dos controladores; (b) correntes I_{sq}; (c) Erro instantâneo das correntes de referência dos controladores.

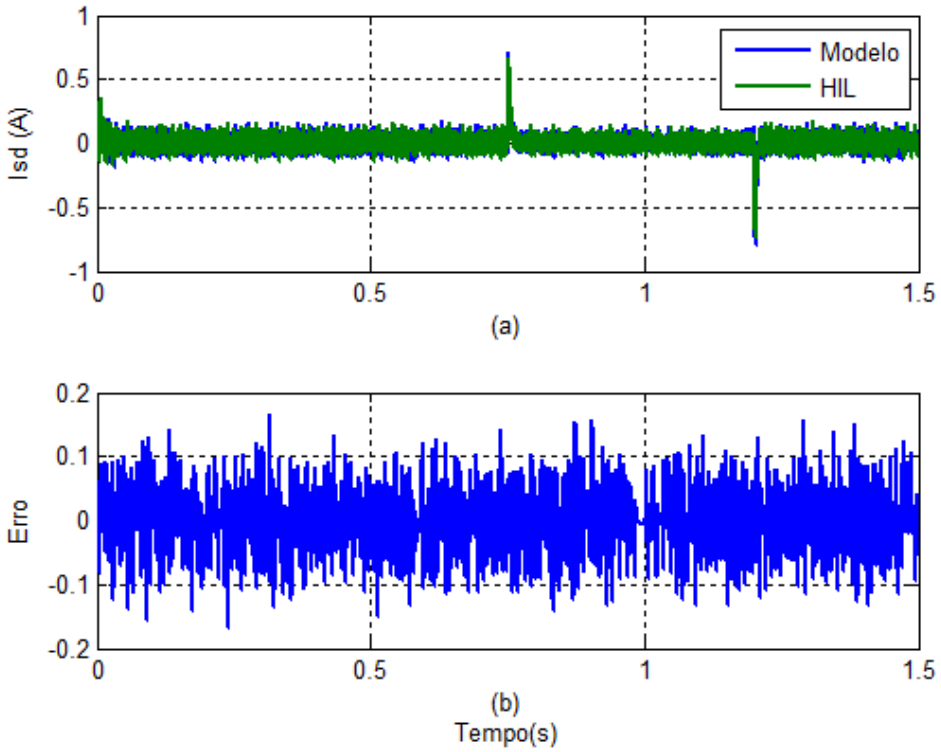


Figura 34. Comparação das correntes I_{sd}: (a) Modelo - linha azul, HIL - linha verde ; (b) Erro instantâneo.

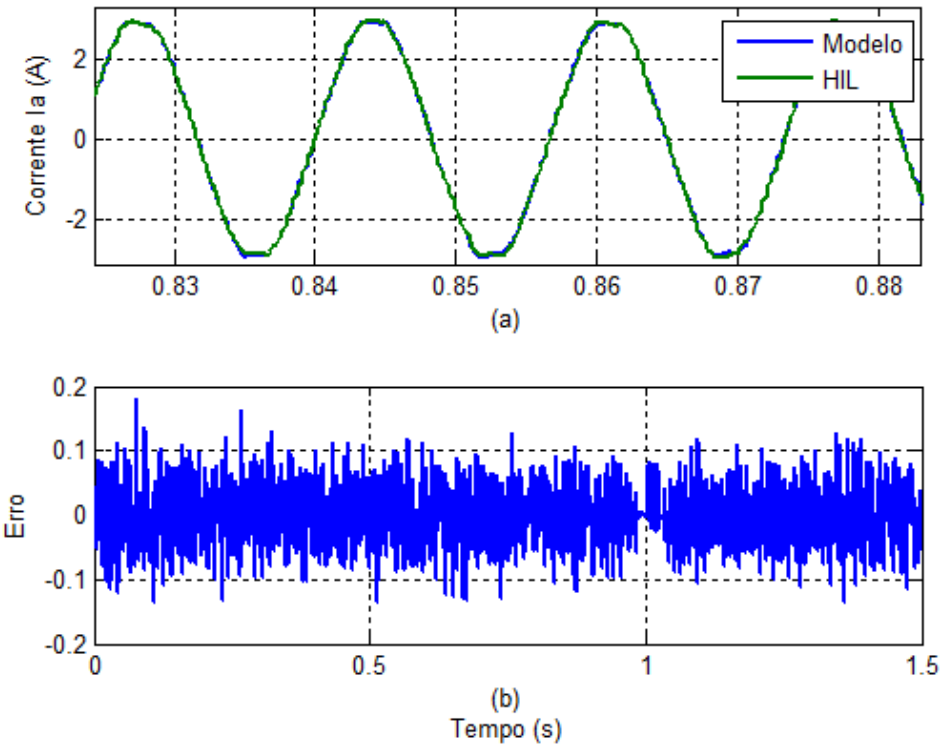


Figura 35. Corrente de fase Ia: (a) ampliação da corrente para período com carga de 1 N.m; (b) Erro Instantâneo.

A Figura 35 (a) compara a corrente de fase i_a com carga aplicada de 1 N.m. Com um número de amostras N_a= 600, o erro médio quadrático foi de 0,00175 A²/Amostra, que representa 0,0571% da amplitude fundamental.

Como se pode verificar nos resultados, o modelo do sistema HiL representa satisfatoriamente o modelo original obtido por (E. d. Fernandes 2006). Devido ao comportamento da máquina, com características praticamente senoidais, a aproximação do modelo por Euller e ponto fixo manteve-se próximo ao modelo de referencia, os erros existentes representam valores baixos.

5.3 CONCLUSÕES

Este capítulo apresentou a metodologia utilizada para a implementação da máquina SMPM na plataforma HIL. Os critérios adotados para a aproximação da máquina em ponto fixo foram demonstrados, onde se buscou o melhor dimensionamento das constantes para se trabalhar com uma representação em Q-27. A máquina foi simulada operando como gerador e como motor.

O modelo descrito na seção 4.1 abstrai os termos da equação geral da máquina. Apesar de simplificar o equacionamento, limita a simulação. Os termos da máquina se encontram pré-processados em constantes que não permitem a sua alteração. Por este motivo, o modelo foi reestruturado para permitir maior flexibilidade, com acesso direto aos parâmetros da máquina e, consequentemente, facilitando a alteração dos valores e a migração para outras máquinas similares.

O modelo da máquina discretizado pelo método de Euller, simulado na plataforma HIL, foi comparado com resultados experimentais e com o modelo de (E. d. Fernandes 2006), que é baseado em uma discretização por Runge-Kutta de 4ª ordem.

Na simulação como gerador, o erro médio quadrático da tensão gerada entre o simulador e a máquina real foi de apenas 1,18% para $v_{s\alpha}$ e 1,39% para $v_{s\beta}$. Para a máquina operando como motor, o sistema de controle via DSP foi utilizado. A corrente de fase i_a foi comparada com o simulador em *software* do modelo de (E. d. Fernandes 2006) e apresentou um erro médio quadrático de 0,0571% da amplitude fundamental.

Estes resultados comprovam que o modelo da máquina SMPM implementado no FPGA representa satisfatoriamente a máquina real e o modelo original. Em ambos os casos o modelo simulado pelo FPGA foi equivalente às referencias.

Capítulo 6

Implementação da IPM no Simulador HiL

A implementação da IPM ocorreu seguindo os mesmos princípios da SMPM. A máquina foi simulada como gerador e como motor. Dentro de cada operação houve a simulação em *software* (Matlab, JAVA em ponto flutuante e ponto fixo), simulação de *hardware* em Verilog (ModelSim) até chegar à compilação em QUARTUS II e FPGA.

6.1 MÁQUINA OPERANDO COMO GERADOR

Esta seção apresentará a implementação da máquina síncrona a imã permanente com imãs interiores. Como demonstrado no Capítulo 3, é uma máquina mais complexa que a SMPM. Entretanto, o equacionamento montado para a SMPM fornece a estrutura base para a implementação da IPM, o equacionamento da corrente sofre poucas alterações e ficam como em (6.13) e (6.14), o conjugado foi aplicado de acordo com a equação (3.52), o equacionamento do movimento permanece inalterado. Os termos referentes a fcem (6.15 e 6.16) foram reestruturados, assim como a inserção do perfil das indutâncias (6.17 a 6.19). Os demais termos presentes no equacionamento da IPM foram representados como constantes seguindo a representação do modelo dinâmico da seção 3.2.4.

$$I_{sd} = \frac{I_{sd}(l_d(\theta_r) - R_s h) + h(V_{sd} - \omega_r I_{sq} l_{ac}(\theta_r) + E_d)}{l_d(\theta_r)} \tag{6.13}$$

$$I_{sq} = \frac{I_{sq}\big(l_q(\theta_r) - R_s h\big) + h(V_{sq} - \omega_r I_{sd} l_{ac}(\theta_r) + E_q)}{l_q(\theta_r)} \tag{6.14}$$

$$E_d = \omega_r(\lambda_{da}\sin(6\theta_r) + \lambda_{db}\sin(12\theta_r)) \tag{6.15}$$

$$E_q = \omega_r\big(\lambda_{pm} + \lambda_{qa}\cos(6\theta_r) + \lambda_{qb}\cos(12\theta_r)\big) \tag{6.16}$$

$$l_d(\theta_r) = L_d + l_{d1}\cos(6\theta_r) \tag{6.17}$$

$$l_q(\theta_r) = L_q + l_{q1}\cos(6\theta_r) \tag{6.18}$$

$$l_{ac}(\theta_r) = l_{ac0} + l_{ac1}\cos 6\theta_r \tag{6.19}$$

Os parâmetros utilizados para este motor estão representados na Tabela 17 de acordo com o trabalho de (Costa 2013), onde pode ser encontrado maiores informações sobre a caracterização da máquina.

Tabela 17. Parâmetros para a máquina IPM.

Potência Nominal	0,25 KW
Corrente Nominal	2,5 A
Velocidade Nominal	900 RPM
Par de Pólos	4
L_d	9,55 mH
L_q	13,22 mH
l_{d1}	0,1 mH
l_{q1}	0,3 mH
l_{ac1}	4,8 mH
l_{dh}	0,55 mH
l_{qh}	2,0 mH
Resistência do estator (r_s)	1,39 Ω
λ_{pm}	177,4 mWb
λ_{da}	30,5 mWb
λ_{db}	1,6 mWb
λ_{qa}	39 mWb
λ_{qb}	12,3 mWb
Momento de Inércia (J)	0,00776 Kgfm²
Coeficiente de Atrito (f_ω)	0,00853 Kgfm²/s

Para a representação das variáveis em ponto fixo foi adotado o mesmo procedimento da máquina SMPM. A representação base do sistema é de Q-27, que oferece uma precisão de $2^{-27} = 7{,}45 \times 10^{-9}$, e o número de algarismos significativos do sistema serem considerados são de 7 casas decimais.

A Tabela 18 mostra os valores das constantes da Tabela 17 e sua aproximação em ponto fixo. A Tabela 19 mostra a relação do erro absoluto com o numero de casas decimais corretas e o erro relativo com o número de algarismos significativos corretos. Nota-se que o número de algarismos significativos buscado para o valor aproximado foi atingido ($n \geq 7$). Assim como no caso da SMPM os registradores em Verilog foram dimensionados para 45 bits com sinal.

Tabela 18. Valores das constantes da máquina IPM e se valor aproximado em ponto fixo.

Constante	x	$\bar{x}$	Representação
L_d (H)	$9{,}55 \times 10^{-3}$	0,0095499977	Q-29
L_q (H)	$13{,}22 \times 10^{-3}$	0,013219999847	Q-32
l_{d1} (H)	$0{,}1 \times 10^{-3}$	$1{,}0000000474 \times 10^{-4}$	Q-34
l_{q1} (H)	$0{,}3 \times 10^{-3}$	$3{,}0000001424 \times 10^{-4}$	Q-34
l_{ac1} (H)	$4{,}8 \times 10^{-3}$	0,004800000227	Q-30
l_{dh} (H)	$0{,}55 \times 10^{-3}$	$5{,}4999999701 \times 10^{-4}$	Q-29
l_{qh} (H)	$2{,}0 \times 10^{-3}$	$2{,}00000009499 \times 10^{-3}$	Q-32
r_s (Ω)	1,39	1,390000000596	Q-29
λ_{pm} (Wb)	$177{,}4 \times 10^{-3}$	0,177400000393	Q-29
λ_{da} (Wb)	$30{,}5 \times 10^{-3}$	0,030500000342	Q-29
λ_{db} (Wb)	$1{,}6 \times 10^{-3}$	0,001600000075	Q-30
λ_{qa} (Wb)	39×10^{-3}	0,039000000804	Q-29
λ_{qb} (Wb)	$12{,}3 \times 10^{-3}$	0,012299999594	Q-29
J (Kgfm²)	0,00776	0,007759999949	Q-32
f_ω (Kgfm²/s)	0,00853	0,008529999991	Q-32

Tabela 19. Analise do erro absoluto e erro relativo para a validação da representação das constantes da máquina IPM.

Constante	$\Delta_{\bar{x}}$	k	$r_{\bar{x}}$	n
L_d (H)	$3{,}900 \times 10^{-10}$	9	$4{,}088 \times 10^{-8}$	7
L_q (H)	$1{,}520 \times 10^{-10}$	9	$1{,}150 \times 10^{-8}$	7
l_{d1} (H)	$4{,}749 \times 10^{-12}$	11	$4{,}749 \times 10^{-8}$	7
l_{q1} (H)	$1{,}424 \times 10^{-11}$	10	$4{,}749 \times 10^{-8}$	7
l_{ac1} (H)	$2{,}279 \times 10^{-10}$	9	$4{,}749 \times 10^{-8}$	7
l_{dh} (H)	$2{,}980 \times 10^{-12}$	11	$5{,}418 \times 10^{-9}$	7
l_{qh} (H)	$9{,}499 \times 10^{-11}$	9	$4{,}749 \times 10^{-8}$	7
r_s (Ω)	$5{,}960 \times 10^{-10}$	8	$1{,}528 \times 10^{-9}$	8
λ_{pm} (Wb)	$3{,}933 \times 10^{-10}$	9	$2{,}217 \times 10^{-9}$	8
λ_{da} (Wb)	$3{,}427 \times 10^{-10}$	9	$1{,}123 \times 10^{-8}$	7
λ_{db} (Wb)	$7{,}599 \times 10^{-11}$	9	$4{,}749 \times 10^{-8}$	7
λ_{qa} (Wb)	$8{,}046 \times 10^{-10}$	9	$2{,}063 \times 10^{-8}$	7
λ_{qb} (Wb)	$4{,}053 \times 10^{-10}$	9	$3{,}295 \times 10^{-8}$	7
J (Kgfm²)	$5{,}051 \times 10^{-11}$	9	$6{,}509 \times 10^{-9}$	7
f_ω (Kgfm²/s)	$8{,}121 \times 10^{-12}$	10	$9{,}520 \times 10^{-10}$	8

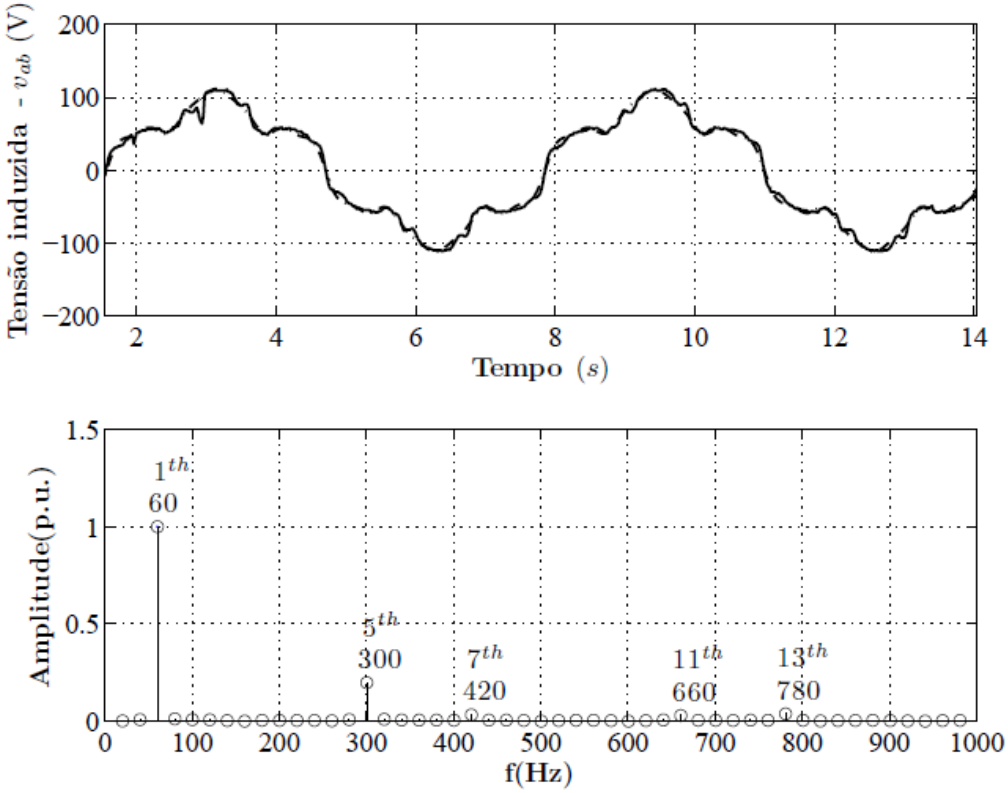


Figura 36. Tensão de linha v_{ab} experimentalmente obtido (linha sólida) e uma versão construída (linha pontilhada) obtidas através das frequências relevantes (abaixo).

A Figura 36 mostra o perfil da tensão induzida obtida experimentalmente e comparada com uma versão construída a partir das amplitudes das componentes harmônicas. Estas especificações serviram como base para a concepção dos parâmetros de fluxo da máquina (Tabela 17) do modelo obtido por (Costa 2013). No caso, a representação das harmônicas do sinal foi aproximada até a décima terceira harmônica, desconsiderando as harmônicas subsequentes. A Figura 37 apresenta a representação harmônica real do sinal medido até o octogésimo harmônico, onde se percebe uma maior quantidade de componentes harmônicos.

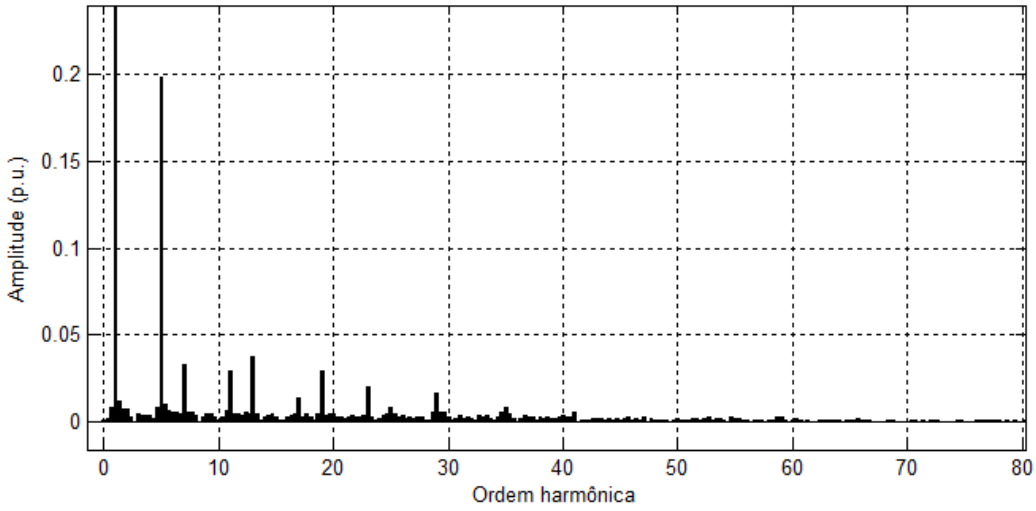


Figura 37. Representação harmônica ampliada para a tensão induzida medida.

Por este motivo, erro médio quadrático percentual entre a tensão induzida experimental e a versão construída é de 26,14%. Este valor poderia ser reduzido se fosse considerado um número maior de componentes harmônicos para representar os fluxos.

Para efeitos de validação do modelo da fcem, os resultados obtidos pelo simulador em FPGA foram comparados com o resultado obtido pela simulação de (Costa 2013), uma vez que, o objetivo é representar este modelo no FPGA. No modelo de (Costa 2013) o método de discretização utilizado é de Runge-Kutta. No caso da análise do perfil das correntes, os valores simulados no FPGA também serão comparados com o resultado experimental.

Como as equações de indutância e fcem exigem valores de seno e cosseno que operam a $6\theta_r$ e $12\theta_r$, foi necessário incluir uma nova função dentro do módulo da máquina em Verilog. O módulo "Funções" apenas retorna um valor de seno e cosseno se o angulo de entrada estiver entre 0 e 2π. Como os ângulos chegam a um valor de até doze vezes, foi implementada a função "corrige_angulo" que tem como objetivo adequar o valor de entrada para uma representação equivalente entre 0 e 2π.

A Tabela 20 apresenta o consumo de recursos do FPGA para simular a fcem da IPM, enquanto que a Tabela 21 demonstra o consumo da máquina como gerador incluindo o equacionamento das correntes. Segundo o resultado da análise de tempo, simular a fcem da máquina exige 91,7ns (correspondente a um *clock* de 10,9MHz) para cada passo de cálculo. O equacionamento completo da máquina como gerador, incluindo o cálculo da fcem, necessida de 0,46µs para ser calculado (equivalente a 2,13MHz).

Tabela 20. Consumo de recursos do FPGA Cyclone IV para simular a fcem do IPM.

Descrição	Necessário	Disponível	(%)
Elementos lógicos (LEs)	4618	114480	4
Bits de memória	66240	3981312	2
Multiplicadores 9bits	110	532	21

Tabela 21. Consumo de recursos do FPGA Cyclone IV para simular a IPM como gerador.

Descrição	Necessário	Disponível	(%)
Elementos lógicos (LEs)	11547	114480	10
Bits de memória	66240	3981312	2
Multiplicadores 9bits	205	532	39

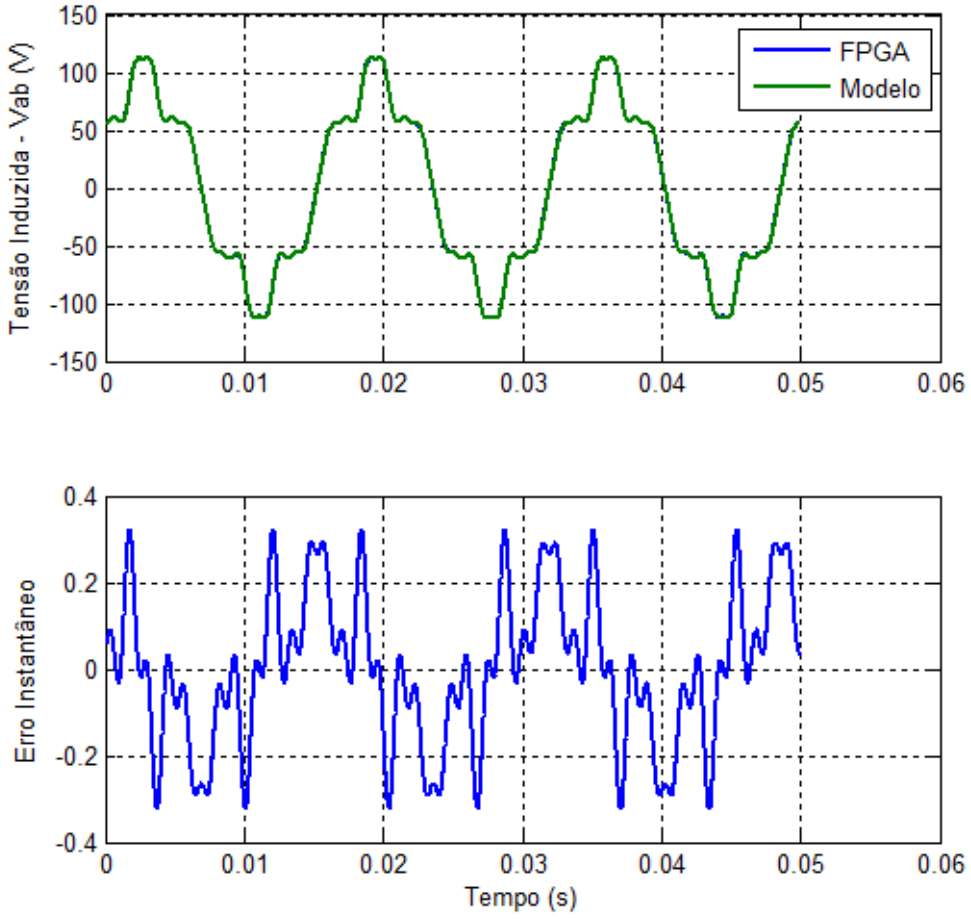


Figura 38. Tensão induzida de linha V_{ab}: (a) FPGA - linha azul, modelo Runge-Kutta-Fehlberg - linha verde; (b) Erro instantâneo entre a tensão induzida calculada pelo FPGA e pelo modelo Runge-Kutta-Fehlberg.

A Figura 38 mostra a comparação da tensão induzida obtida pelo FPGA e o modelo de (Costa 2013). Na simulação, a máquina é acionada como gerador a uma velocidade de 900

RPM. O erro médio quadrático foi calculado com N_a = 500 e apresenta um valor de 0,0303 V²/amostra, que representa 0,026% da amplitude da força eletromotriz simulada.

Para a verificação das correntes de fase foram utilizados coletados da máquina real. A máquina a IPM foi acionada por uma máquina auxiliar a 900 rpm e conectada a um banco resistivo (33,33Ω/fase). As correntes de fase i_a, i_b, i_c medidas são mostradas na Figura 39.

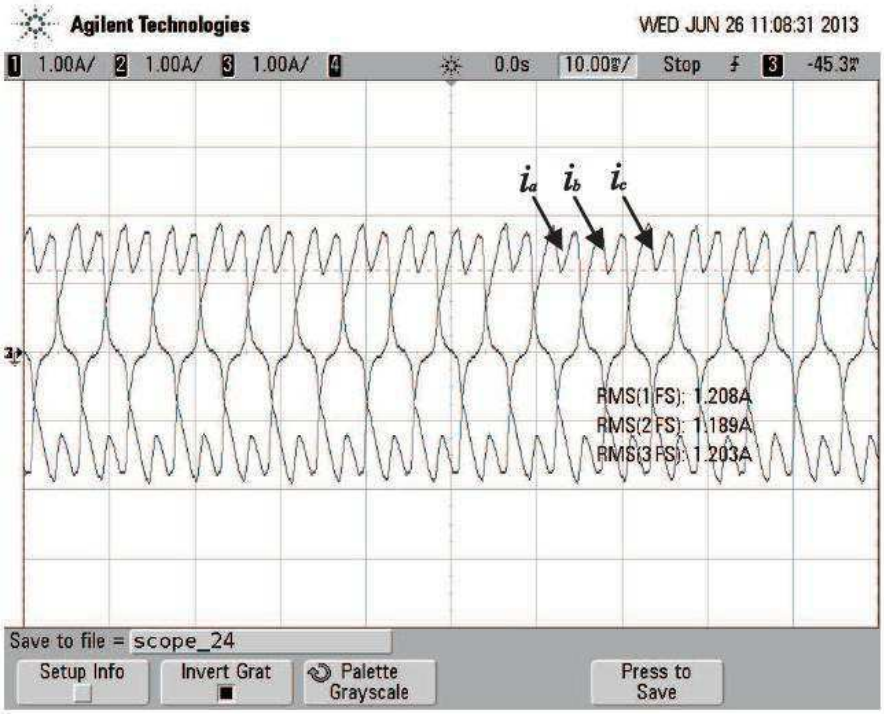


Figura 39. Correntes de fase i_a, i_b, i_c registradas no osciloscópio, com a máquina funcionando como gerador, e em seus terminais está conectado um banco resistivo (de 33,33Ω/150W por fase).

Em posse destes resultados, foram executadas as simulações com as mesmas características no sistema HIL e pelo modelo de Costa para a comparação dos resultados, mostrados na Figura 40 para a corrente de fase i_a.

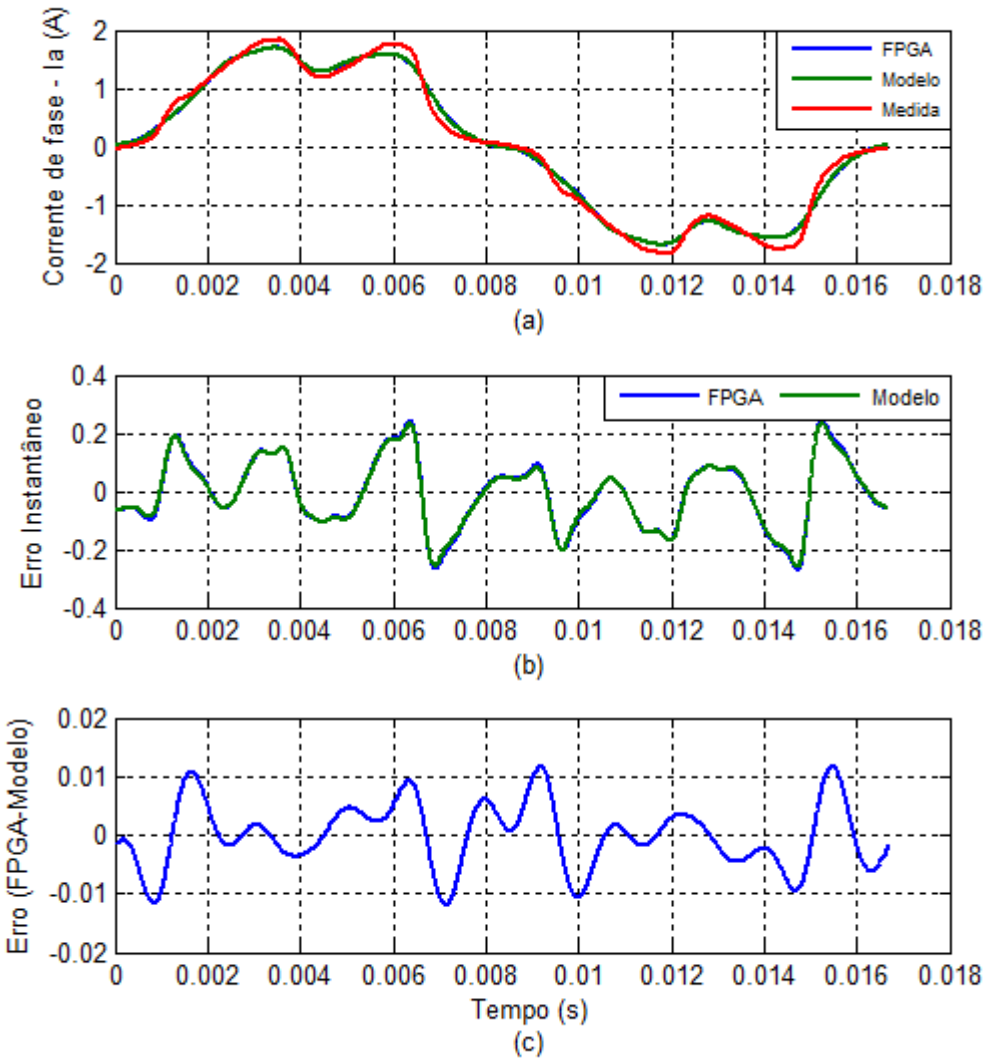


Figura 40. Corrente de fase i_a (a) obtida pelo FPGA, Medida e do Modelo; (b) Erro instantâneo do FPGA e Modelo em relação à experimental; (c) Erro instantâneo entre o FPGA e o Modelo.

O critério de avaliação adotado foi pelo erro médio quadrático relacionando os valores obtidos na equação (6.10). Para os resultados da Figura 40, utilizou-se um número de amostras N_a = 1670. Os erros médios quadráticos das correntes entre o FPGA e o Modelo apresentaram o mesmo valor para as três fases e foram de $e_{mqi_a} = e_{mqi_b} = e_{mqi_c}$ =30,57µ A²/Amostras que correspondem a 0,00167% das amplitudes das correntes de fase. A Tabela 22 mostra a comparação entre o sistema HiL e os valores medidos, apresentando o erro médio quadrático e a porcentagem em relação a cada corrente de fase.

Tabela 22. Valores do critério de validação para os erros médios quadráticos entre as correntes de fase medidas e simuladas no sistema HIL.

Correntes $i_{a,b,c}$	**e_{mq}(A²/Amostras)**	**Erro(%)**
Corrente i_a	0,0147	0,80
Corrente i_b	0,0124	0,68
Corrente i_c	0,0132	0,72

Em face dos resultados obtidos, pode-se dizer que o modelo simulado no FPGA é equivalente ao modelo obtido por (Costa 2013) e representa satisfatoriamente o comportamento dinâmico da máquina experimental.

6.2 MÁQUINA OPERANDO COMO MOTOR

Após validado o modelo da máquina IPM, a mesma foi adaptada como motor e incorporada ao sistema HiL. A Tabela 23 apresenta o consumo de recursos necessários do FPGA para implementar a máquina IPM como motor. Segundo a ferramenta de análise de tempo do Quartus II, o tempo necessário para calcular o equacionamento da máquina como motor é de 0,907µs (aprox. 1,1MHz). Ou seja, já se encontra próximo do limite para o passo de cálculo desejado.

Tabela 23. Consumo de recursos do FPGA Cyclone IV para simular a IPM como Motor

Descrição	**Necessário**	**Disponível**	**(%)**
Elementos lógicos (LEs)	18838	114480	16
Bits de memória	99360	3981312	2
Multiplicadores 9bits	376	532	71

As figuras 41-44 mostram a resposta dinâmica do sistema de acionamento do motor. O IPM parte sem carga a uma velocidade de referencia de 900 rpm. No período de tempo compreendido entre 0,85s e 1s é aplicada uma carga mecânica de 1 N.m. Os resultados do sistema HIL são comparados com o simulador do modelo de Costa (Modelo).

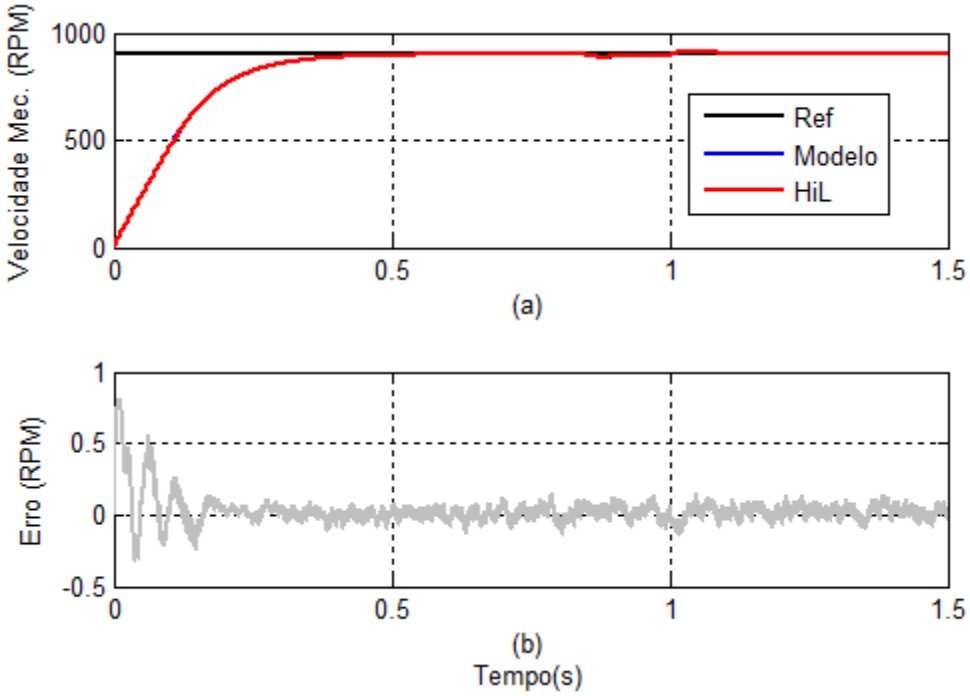


Figura 41. Comparação de Velocidade do motor IPM: (a) Velocidades do Modelo e HIL a uma Referencia de 900 rpm; (b) Erro Instantâneo entre o Sistema HIL e o Modelo.

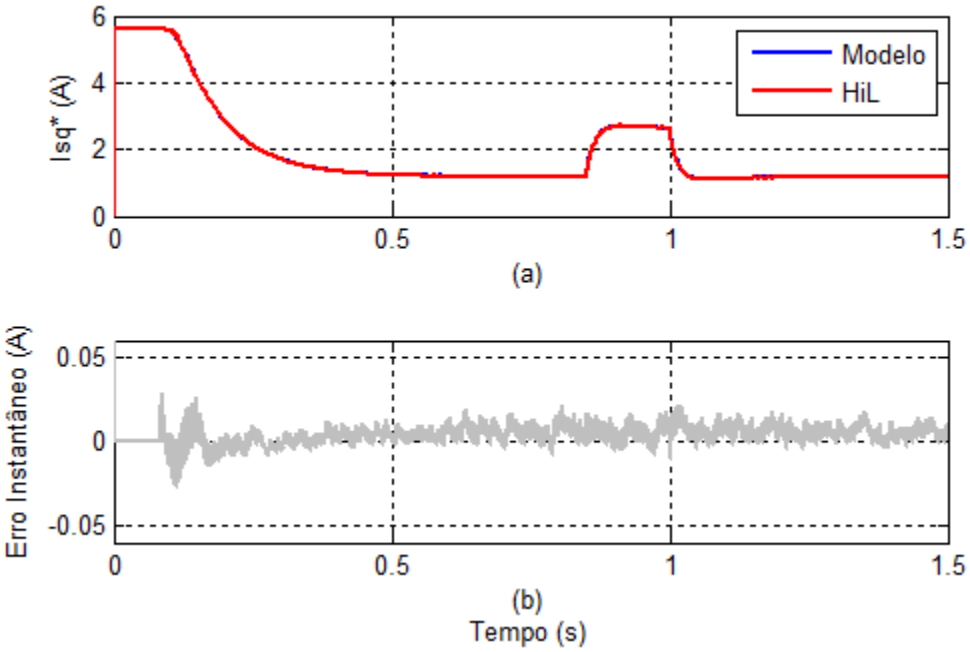


Figura 42. Comparação da Corrente I_{sq} de referência dos controladores para o IPM: (a) Controlador do modelo - linha azul, Controlador da plataforma HIL (DSP) - linha vermelha; (b) erro instantâneo entre os controladores do modelo e da plataforma HIL.

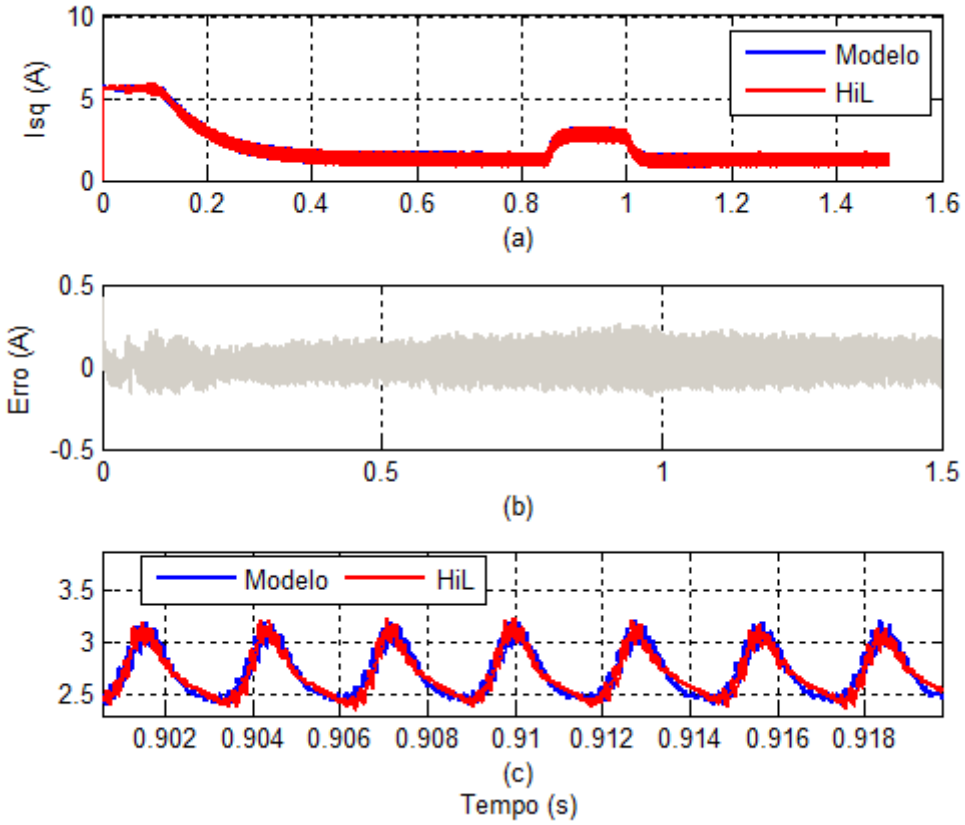


Figura 43. Comparação da Corrente I_{sq} do IPM: (a) Cuvas de corrente; (b) Erro instantâneo entre o sistema HIL e o modelo; (c) Ampliação das correntes I_{sq}.

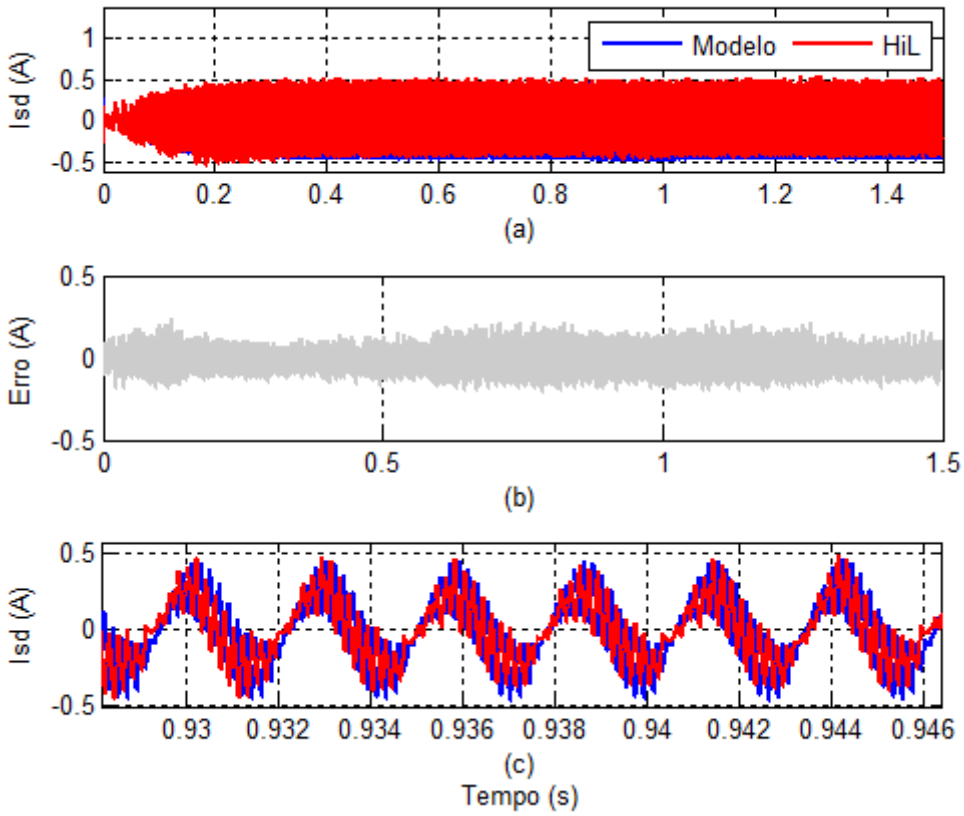


Figura 44. Comparação da Corrente I_{sd} do IPM: (a) Cuvas de corrente; (b) Erro instantâneo entre o sistema HIL e o modelo; (c) Ampliação das correntes I_{sd}.

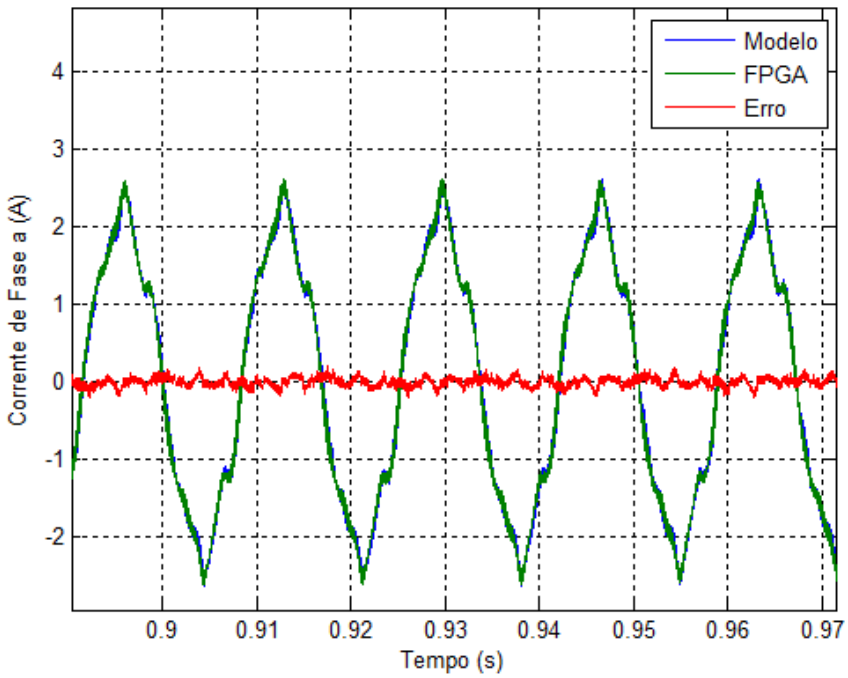


Figura 45. Ampliação da corrente de fase Ia com carga de 1 N.m.

A Figura 45 apresenta a corrente de fase i_a com carga aplicada de 1 N.m. Com um número de amostras N_a= 800, o erro médio quadrático foi de 0,0030 A²/Amostra, que representa 0,113% da amplitude fundamental.

Para garantir-se que os pequenos erros numéricos percebidos não representam um problema, uma nova simulação foi realizada. Com o passar do tempo, o acumulo de erros em FPGA poderia fazer com que a máquina simulada não representasse mais o modelo original. A diferença de representação ocasiona em discrepância nos sinais de referência esperados pelo controlador, acumulando erros de tal forma, que ocasionaria a instabilidade e provável "colapso" no controle da máquina simulada.

A simulação consiste em acionar a máquina a 900 rpm e aplicar uma carga mecânica de 1N.m. Após entrar em regime a corrente de referência I_{sq}^* deve permanecer com o mesmo comportamento durante toda a simulação. A corrente de referencia I_{sq}^* do controlador do DSP é monitorada durante 24 horas para verificar se este padrão se confirma. O resultado é mostrado na Figura 46.

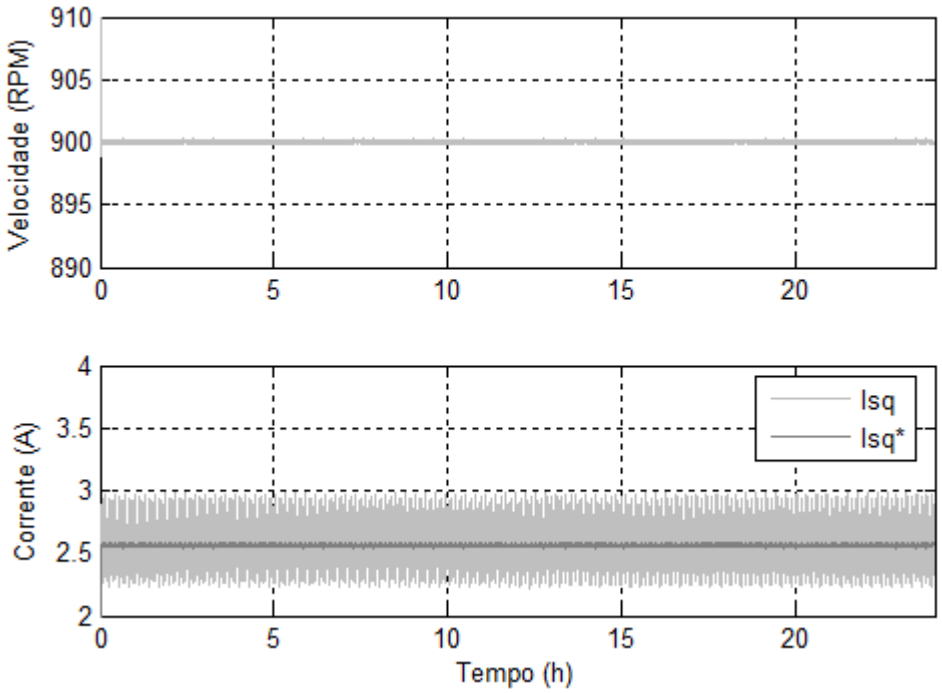


Figura 46. Simulação do IPM durante 24 horas: (a) Velocidade mecânica da máquina; (b) corrente I_{sq} - cinza claro, corrente I_{sq}^* - cinza escuro.

Como se pode perceber pelo resultado a velocidade manteve-se constante, assim como a corrente i_{sq} e i_{sq}^* mantiveram o mesmo padrão de comportamento durante todo o tempo. Os valores das grandezas se mantiveram dentro do resultado esperado.

6.3 CONCLUSÕES

Este capítulo apresentou a implementação da máquina IPM na plataforma HiL. Trata-se de uma replicação do método utilizado para o SMPM. Os critérios adotados para a aproximação da máquina em ponto fixo foram demonstrados, onde se buscou o melhor dimensionamento das constantes para se trabalhar com uma representação em Q-27. A máquina foi simulada operando como gerador e como motor.

O modelo da máquina discretizado pelo método de Euler, simulado na plataforma HiL, foi comparado com resultados experimentais e com o modelo de (Costa 2013) que é baseado em uma aproximação por Runge-Kutta de quarta ordem.

Na simulação como gerador, as correntes de fase obtidas por FPGA apresentam um erro médio quadrático em torno de 0,7% em relação ao resultado experimental e um erro ainda menor quando comparado ao obtido por (Costa 2013) de apenas 0,00167%.

Na simulação como motor, o modelo do sistema HIL representa satisfatoriamente o modelo original obtido por (Costa 2013). Sendo executado em plataformas distintas o erro médio quadrático da corrente de fase i_a foi de apenas 0,113%.

Com base nos resultados obtidos com a plataforma de simulação em FPGA pode-se concluir que o sistema representa adequadamente o modelo da máquina a imã IPM.

Capítulo 7

Conclusões e Trabalhos Futuros

Neste trabalho foi apresentado o desenvolvimento de uma plataforma de simulação em tempo real baseado em *Hardware in-the-loop*. No sistema, a planta é simulada em FPGA e controlada a partir de um dispositivo DSP. A interface entre os dois dispositivos é feita através de entradas e saídas digitais.

Do ponto de vista de utilização, o usuário opera com o simulador HIL do mesmo modo como o faria em um sistema real. Visto que, em ambos os casos, o usuário está trabalhando diretamente com o controlador. Seja o DSP conectado a um sistema real, com um conversor real, motor real ou conectado a um sistema simulado; a interação entre o projetista e o *hardware* de controle é a mesma.

Na sua concepção, a plataforma foi idealizada para se trabalhar com máquinas elétricas. Na sua arquitetura está prevista a emulação de sete conversores A/D e a utilização de três sinais PWM interligados a um módulo conversor de potência ideal, que comanda o módulo da planta simulada. Porém, partindo desta configuração, pode-se adaptar a plataforma para outras necessidades. Há uma reserva de pinos na interface entre os dispositivos que permite adequar a plataforma para alguma demanda extra. Além do que, a emulação de conversores A/D também pode ser aumentada.

Na dissertação, foram implementados dois modelos de máquinas síncronas a imã permanente, o que não restringe a aplicação apenas para estas máquinas. Uma máquina de indução, por exemplo, também poderia ser utilizada.

O modelo matemático das máquinas síncronas a imã permanente foi apresentado, onde foi utilizada a representação *dq*. A simplificação oferecida neste referencial demanda menor esforço computacional frente ao estacionário.

A máquina SMPM possui um comportamento mais linear, sem apresentar muitas saliências, as grandezas no referencial estacionário possuem um comportamento senoidal. Enquanto que, o modelo da máquina IPM ocorre o contrário, as saliências são bem

significativas, distorcendo o que seria um comportamento senoidal da máquina. Estas características foram representadas nos modelos simulados. Entretanto, saliências provocadas por efeitos de saturação das máquinas não foram considerados.

A discretização dos modelos matemáticos, também pensando na simplificação e menor esforço computacional, foram feitos pelo método de Euller. Com um passo de cálculo de 1μs o método apresenta resultados satisfatórios.

Os critérios para a conversão do modelo discreto para a aritmética em ponto fixo foram discutidos. A representação em ponto fixo foi de Q-27, o que permitiu resultados satisfatórios. Os erros frente aos modelos de referência, com aritmética de ponto flutuante de precisão dupla, foram pequenos, com valores abaixo de 0,15% para ambos os motores.

TRABALHOS FUTUROS

Neste trabalho foi desenvolvido uma plataforma de simulação em tempo real e apresentada uma metodologia para a implementação de duas máquinas síncronas a imã permanente. São sugeridos trabalhos futuros:

- Melhorar o modelo do conversor de potência, substituindo o modelo ideal das chaves para um comportamento real, incorporando efeito de tempo-morto das chaves do conversor;
- Investigar a influencia da saturação na máquina a imã com imãs montados no interior do rotor (IPM) e assim representar esses efeitos no modelo simulado em FPGA.
- Padronizar o procedimento de implementação da simulação em tempo-real a outros tipos de motores elétricos, como por exemplo, o motor de indução trifasico.

Referências Bibliográficas

Abourida, Simon, Jean Bélanger, e Christian Dufour. “Real-Time HIL Simulation of a Complete PMSM Drive at 10us Time Step.” *IEEE Power Electronics and Applications.* 2005. 9 pp.- P.9 .

Albuquerque, André Ribeiro Linz. “Aplicações de Hardware-in-the-loop no desenvolvimento de uma mão robótica.” Universidade de São Paulo, São Carlos, 2007, 172.

Altera. *Quartus® II Introduction for Verilog Users.* Altera® Corporation, 2010.

Altera® Corporation. *QUARTUS® II Introduction for Verilog Users.* 2010.

Bachir, Tarek Ould, e Jean-Pierre David. “FPGA-Based Real-Time Simulation of State-Sace Models Using Floating-Point Cores.” *EPE-PEMC 2010.* IEEE, 2010. 26-31.

Bahri, I., M-W. Naouar, E. Monmasson, I. Slama-Belkhodja, e L. Charaabi. “Design of an FPGA-Based Real-Time Simulator for Electrical System.” *13 Internationa Power Electronics and Motion Control Conference.* IEEE, 2008.

Baratto, Giovani. “Solução de Equações Diferenciais Ordinárias Usando Métodos Numéricos.” UFSM - Universidade Federal de Santa Maria, 2007, 14.

Bouscayrol, A. “Different types of Hardware-in-the-Loop simulation for electric drives.” *IEEE ISIE* , June 30-July 2 2008: 2146-2151.

Carpentier, Michel P. J. “Análise Numérica.” Departamento de Matemática do Instituto Superior Técnico U. T. L., 1993, 228.

Charaabi, L, E Monmasson, e I Slama-Belkhodja. “FPGA-based Real-Time emulation of Induction motor using fixed point representation.” *IECON 2008*, 2008: 2393-2398.

Colli, Vincenzo Delli, Roberto Di Stefano, e Fabrizio Marignetti. “A System-on-Chip Sensorless Control for a Permanent-Magnet Sycnchronous Motor.” *IEEE Transactions on Industrial Electronics.* IEEE, 2010.

Colli, Vincenzo Delli, Roberto Di Stefano, Fabrizio Marignetti, e Maurizio Scarano. “Hardware in the Loop Simulation of a FPGA-based Speed and Position Observer for non-Salient Permanent Magnet Synchronous Motors.” *IECON.* IEEE, 2007. 992-997.

Costa, Marcos Aurélio Araújo. *Caracterização e Controle de um Motor Síncrono a Imã Interiores.* Campina Grande-PB: Dissertação (Mestrado) - Universidade Federal de Campina Grande, 2013.

Da Silva, Italo Roger Ferreira Moreno Pinheiro. "Emulação em FPGA de chaves IGBTs com características reais." Universidade Federal de Campina Grande, Campina Grande, 2012, 23.

Darba, A, F. De Belie, T Vyncke, e J Melkebeek. "FPGA-Based Real-Time Simulation of Sensorless Control of PMSM Drive at Standstill." *International Symposium on Power Electronics, Electrical Drives, Automation and Motion*, 2012: 1063-1068.

Dufour, Christian, Guillaume Dumur, Jean -Nicolas Paquin, e Jean Bélanger. "A multi-core pc-based simulator for the hardware-in-the-loop testing of modern train and ship traction systems." *EPE-PEMC 2008.* 2008. 1475 - 1480.

Dufour, Christian, Jean Bélanger, Simon Abourida, e Vincent Lapointe. "Real-time simulation of finite-element analysis permanent magnet synchronous machine drives on a FPGA card." *IEEE Power Electronics and Applications*, 2007: 1-10.

Dufour, Christian, Simon Abourida, Jean Bélanger, e Vincent Lapointe. "Real-Time Simulation of Permanent Magnet Motor Drive on FPGA Chip for High-Bandwidth Controller Test and Validation." *IEEE ISIE 2006*, 2006: 4581-4586.

Dufour, Christian, Vincent Lapointe, Jean Bélanger, e Simon Abourida. "Hardware-int-the-Loop Closed-Loop Experiments with an FPGA-based Permanent Magnet Synchronous Motor Drive System and a Rapidly Prototyped Controller." *ISIE 2008* (IEEE), 2008: 2152-2158.

Duman, Erkan, Hayrettin Can, e Erhan Akin. "Real Time FPGA Implementation of Induction Machine Model - A Novel Approach." IEEE, 2007.

Fernandes, E. M. *Estimador de posição rotórica para motor síncrono a ímã permanente (Tese).* Campina Grande - PB: Universidade Federal de Campina Grande, 2011.

Fernandes, Eisenhawer de M., Denis Ricardo Huller, Alexandre Cunha Oliveira, e Antonio M. N. Lima. "Estimação de Posição Rotórica para motores síncronos a imã permanente baseado em sistema de simulação em tempo real." *XIX Congresso Brasileiro de Automática, CBA 2012*. Campina Grande: CBA, 2012. 1798-1805.

Fernandes, Eisenhawer de Moura. “Estimação de posição e velocidade de uma máquina síncrona a ímã permanente.” Universidade Federal de Campina Grande, Campina Grande, PB, 2006, 145.

Ferreira, Thullyo Dennier Castro Reis. “Uma Arquitetura Reed-Solomon baseada em FPGA/Soft-core.” Unifei, Itajubá-MG, 2011, 61.

Gebregergis, A, e P Pillay. “Implementation of fuel cell emulation on DSP and dSPACE controllers in the design of power electronic converters.” 46, n. 1 (2010): 285-294.

Huller, Denis Ricardo, Eisenhawer de M. Fernandes, e Alexandre Cunha Oliveira. “Speed sensorless control of PMSM motors implemented in real-time simulator.” *Industry Applications (INDUSCON), 2012 10th IEEE/IAS International Conference on.* Fortaleza: IEEE, 2012. 1-7.

Kuon, R. E. R. J., e I E Tessier. “FPGA ARCHITECTURE: Survey and Challenges.” *Foundations and Trends in Electronic Design Automation* 2 (2008): 135-253.

Le-Huy, Philippe, Sylvain Guérette, Louis Dessaint, e Hoang Le-Huy. “Dual-Step Real-Time Simulation of Power Electronic Converters Using an FPGA.” *IEEE ISIE 2006.* Montréal, Québec, Canada: IEEE, 2006. 1548-1553.

Lobão, Diomar Cesar. *Introdução aos métodos numéricos.* http://www.professores.uff.br/diomar_cesar_lobao/material/Metodos_Numericos/UFF_Metodos_Numericos.pdf.

Lucía, O, O Jiménez, L A Barragán, I Urriza, J M Burdío, e D Navarro. “Real-time FPGA-based Hardware-in-the-Loop Development Test-Bench for Multiple Output Power Converters.” IEEE, 2010.

Lucía, Óscar, Isidro Urriza, Luis Barragán, e Denis Navarro. “Real-Time FPGA-Based Hardware-in-the-Loop Simulation Test Bench Applied to Multiple-Output Power Converters.” *IEEE Transactions on Industry Applications.* IEEE, 2011.

Ma, Zhixun, e Ralph Kennel. “System-on-Chip Sensorless Control of PMSM Combining Signal Injection and Flux Observer.” *IEEE 7th International Power Electronics and Motion Control conference*, June 2-5 2012: 1201-1205.

Matar, Mahmoud, e Reza Iravani. “FPGA Implementation of the Power Electronic Converter Model for Real-Time Simulation of Electromagnetic Transients.” *IEEE Transaction on Power Delivery*, April de 2010.

Matar, Mahmoud, Mohamed Abdel-Rahman, e Abdel-Mohaimen Soliman. “FPGA - Based Real-Time Digital Simulation.” *International Conference on Power System Transients.* Montreal, 2005.

Menghal, P. M., e A. Jaya Laxmi. “Real Time Control of Electrical Machine: A Review.” *IEEE*, 2010: 6.

Morgado, Luísa. http://home.utad.pt/~luisam/Teoria%20Erros.pdf (acesso em 07 de 04 de 2013).

Myaing, Aung, e Venkata Dinavahi. “FPGA-Based Real-Time Emulation of Power Electronic Systems With Detailed Representation of Device Characteristics.” *IEEE Transactions on Industrial Eletronics.* IEEE, 2011.

Palnitkar, Samir. *Verilog HDL: A Guide to Digital Design and Sythesis.* Second Edition. Pretince Hall, 2003.

Pellini, Eduardo Lorenzetti. “Um arcabouço para aplicações em tempo real em sistemas de potência.” Escola Politécnica da Universidade de São Paulo, São Paulo, 2010, 184.

Rodrigues, Rosália, e Antonio Pereira. *Capítulo I - Representação de números e erros.* http://www2.mat.ua.pt/rosalia/cadeiras/AN/TPcap1.pdf (acesso em 07 de 04 de 2013).

Soares, Pedro Manuel Oliveira dos Reis. “Discretização de Controladores Contínuos (Dissertação).” Faculdade de Engenharia da Universidade do Porto, Porto, 1996, 115.

Song, Ke, Liu Weituo, e Guangzhao Luo. “Permanent Magnet Synchronous Motor Field Oriented Control and HIL Simulation.” *IEEE Vehicle Power and Propulsion Conference (VPPC).* 2008. pp. 6.

Tursini, M, R Petrella, e A Scafati. “Speed and Position Estimation for PM Synchronous Motor with Back-EMF Observer.” *IEEE Conference of Industrial Electronics Society.* 2005. 2083-2090.

Vahid, Frank. *Sistemas digitais: projeto, otimização e HDLs.* Porto Alegre: Artmed, 2008.

Wang, Guoqiang, Trung N. Tran, e Hugo Andrade. “A Graphical Programming and Design Environment for FPGA-based Hardware.” IEEE, 2010. 337-340.

Washington, Chris, e Jordan Doman. “Creating Next Generation HIL Simulators with FPGA Technology.” *IEEE*, 2010: 6.

APÊNDICES

APÊNDICE A

Sintonia dos Controladores das Máquinas

Cálculo do controlador de corrente

O controlador de corrente implementado nos testes é um controlador PI síncrono no referencial síncrono do rotor. O diagrama contendo o sistema elétrico formado pelos controladores de corrente e o modelo da máquina pode ser visto na Figura 47. Percebe-se também a existência de um termo de acoplamento entre as malhas d e q, representados por $\omega_r l_{sq}$ e $\omega_r l_{sd}$.

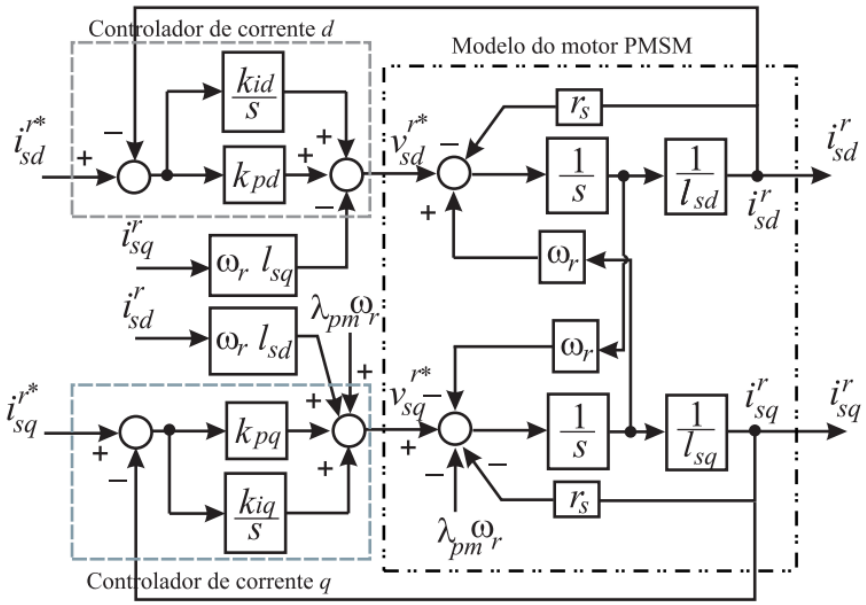


Figura 47. Diagrama de controle de corrente do motor a ímã permanente.

Deste modo, a estrutura do controlador PI padrão é modificada para desacoplar o acoplamento existente entre as malhas d e q. Os termos de desacoplamento possibilitam

melhorar o desempenho deste controlador de corrente independente da frequência síncrona ω_r. O equacionamento para a obtensão dos cálculos dos ganhos é apresentado como em (E. d. Fernandes 2006). Será descrito o procedimento para o cálculo dos ganhos do controlador PI de eixo direto, para o controlador de eixo *q* é adotado um procedimento similar.

A Figura 48 ilustra o sistema formado pelo controlador PI de eixo *d* e a malha de eixo *d* da máquina. A função de transferência de malha aberta (FTMA) é dada por:

$$G_{od}(s) = \frac{\frac{k_{pd}}{l_{sd}}\left(s+\frac{k_{id}}{k_{pd}}\right)}{s\left(s+\frac{r_s}{L_d}\right)} \tag{A.1}$$

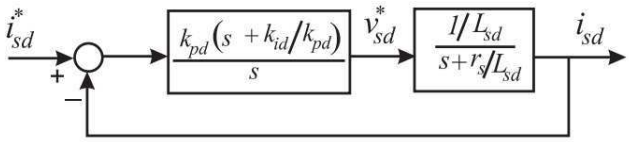


Figura 48. Controlador de corrente PI síncrono para a malha de eixo d.

Cancelando-se o pólo do sistema elétrico da malha *d* com o zero do controlador PI, tem-se:

$$\frac{k_{id}}{k_{pd}} = \frac{r_s}{l_{sd}} \tag{A.2}$$

Portanto, a função de transferência de malha fechada (FTMF) é dada por:

$$G_{fd} = \frac{\frac{k_{pd}}{l_{sd}}}{\left(s+\frac{k_{pd}}{l_{sd}}\right)} = \frac{1}{s\left(\frac{l_{sd}}{k_{pd}}\right)+1} \tag{A.3}$$

A partir da equação (A.3), o ganho k_{pd} é definido em função da frequência de corte ou largura de faixa desejada para a função de transferência de malha fechada (f_d), ou seja:

$$k_{pd} = 2\pi f_d L_{sd} \tag{A.4}$$

Utilizando o mesmo procedimento para determinação dos ganhos do controlador PI síncrono de eixo q, pode-se sumarizar como foram determinados os ganhos dos controladores de corrente de eixo d e q.

- eixo d: $k_{id} = 2\pi f_d L_{sd}$ e $k_{pd} = \frac{L_{sd}}{r_s} k_{id}$;
- eixo q: $k_{iq} = 2\pi f_q L_{sq}$ e $k_{pq} = \frac{L_{sq}}{r_s} k_{iq}$;

onde f_d e f_q são as larguras de faixa desejadas das malhas fechadas do eixo d e q, respectivamente.

Cálculo para o controlador de velocidade

O cálculo para o controlador de velocidade também pode ser encontrado em (E. d. Fernandes 2006), mas será descrito devido a importância na utilização no trabalho. O diagrama de blocos da Figura 49 representa o controlador PI de velocidade e a malha mecânica da máquina.

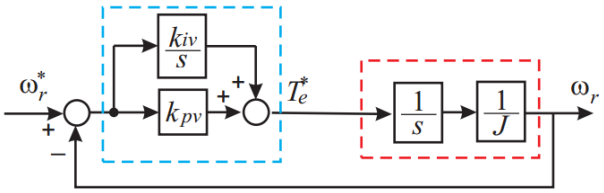


Figura 49. Controlador de velocidade e planta mecânica da máquina PMSM.

A função de transferência de malha fechada para a malha de velocidade é expressa por:

$$G_{f\omega}(s) = \frac{k_{pv}(s+\frac{k_{iv}}{k_{pv}})}{Js^2+k_{pv}s+k_{iv}} \qquad \text{(A.5)}$$

O modelo mecânico da máquina apresenta um polo na origem (s = 0). Dessa forma, não se pode utilizar a técnica de cancelamento de polo da planta com o zero do controlador de velocidade. Portanto, foi estabelecido que os polos da função de transferência de malha fechada sejam polos reais e idênticos. Assim, a discriminante do polinômio $p(s) = Js^2 +$

$k_{pv}s + k_{iv}$ deve ser nulo. Dessa forma, obtém-se como relação entre os ganhos do controlador:

$$\frac{k_{iv}}{k_{pv}^2} = \frac{1}{4J} \tag{A.6}$$

Uma vez garantida esta condição, os polos do sistema em malha fechada são dados por:

$$s_1 = s_2 = \frac{k_{pv}}{2J} \tag{A.7}$$

Da relação acima, pode-se determinar o ganho proporcional do controlador com base na frequência f_v, que determina a largura de faixa do sistema em malha fechada.

$$k_{pv} = 4\pi J f_v \tag{A.8}$$

APÊNDICE B

Códigos Fonte Implementados em FPGA

B.1 Módulo de nível superior

```
// --------------------------------------------------------------------
// de2_115_golden_top.v
// --------------------------------------------------------------------

module DE2_115_GOLDEN_TOP(

      //////// CLOCK //////////
      CLOCK_50,
   CLOCK2_50,
   CLOCK3_50,
      ENETCLK_25,

      //////// Sma //////////
      SMA_CLKIN,
      SMA_CLKOUT,

      //////// LED //////////
      LEDG,
      LEDR,

      //////// KEY //////////
      KEY,

      //////// SW //////////
      SW,

      //////// SEG7 //////////
      HEX0,
      HEX1,
      HEX2,
      HEX3,
      HEX4,
      HEX5,
```

```
HEX6,
HEX7,

//////// LCD //////////
LCD_BLON,
LCD_DATA,
LCD_EN,
LCD_ON,
LCD_RS,
LCD_RW,

//////// RS232 //////////
UART_CTS,
UART_RTS,
UART_RXD,
UART_TXD,

//////// PS2 //////////
PS2_CLK,
PS2_DAT,
PS2_CLK2,
PS2_DAT2,

//////// SDCARD //////////
SD_CLK,
SD_CMD,
SD_DAT,
SD_WP_N,

//////// VGA //////////
VGA_B,
VGA_BLANK_N,
VGA_CLK,
VGA_G,
VGA_HS,
VGA_R,
VGA_SYNC_N,
VGA_VS,

//////// Audio //////////
AUD_ADCDAT,
AUD_ADCLRCK,
AUD_BCLK,
AUD_DACDAT,
AUD_DACLRCK,
AUD_XCK,

//////// I2C for EEPROM //////////
EEP_I2C_SCLK,
```

```
EEP_I2C_SDAT,

//////// I2C for Audio and Tv-Decode //////////
I2C_SCLK,
I2C_SDAT,

//////// Ethernet 0 //////////
ENET0_GTX_CLK,
ENET0_INT_N,
ENET0_MDC,
ENET0_MDIO,
ENET0_RST_N,
ENET0_RX_CLK,
ENET0_RX_COL,
ENET0_RX_CRS,
ENET0_RX_DATA,
ENET0_RX_DV,
ENET0_RX_ER,
ENET0_TX_CLK,
ENET0_TX_DATA,
ENET0_TX_EN,
ENET0_TX_ER,
ENET0_LINK100,

//////// Ethernet 1 //////////
ENET1_GTX_CLK,
ENET1_INT_N,
ENET1_MDC,
ENET1_MDIO,
ENET1_RST_N,
ENET1_RX_CLK,
ENET1_RX_COL,
ENET1_RX_CRS,
ENET1_RX_DATA,
ENET1_RX_DV,
ENET1_RX_ER,
ENET1_TX_CLK,
ENET1_TX_DATA,
ENET1_TX_EN,
ENET1_TX_ER,
ENET1_LINK100,

//////// TV Decoder //////////
TD_CLK27,
TD_DATA,
TD_HS,
TD_RESET_N,
TD_VS,
```

```
/////// USB OTG controller
OTG_DATA,
OTG_ADDR,
OTG_CS_N,
OTG_WR_N,
OTG_RD_N,
OTG_INT,
OTG_RST_N,
OTG_DREQ,
OTG_DACK_N,
OTG_FSPEED,
OTG_LSPEED,
 //////// IR Receiver //////////
 IRDA_RXD,

 //////// SDRAM //////////
 DRAM_ADDR,
 DRAM_BA,
 DRAM_CAS_N,
 DRAM_CKE,
 DRAM_CLK,
 DRAM_CS_N,
 DRAM_DQ,
 DRAM_DQM,
 DRAM_RAS_N,
 DRAM_WE_N,

 //////// SRAM //////////
 SRAM_ADDR,
 SRAM_CE_N,
 SRAM_DQ,
 SRAM_LB_N,
 SRAM_OE_N,
 SRAM_UB_N,
 SRAM_WE_N,

 //////// Flash //////////
 FL_ADDR,
 FL_CE_N,
 FL_DQ,
 FL_OE_N,
 FL_RST_N,
 FL_RY,
 FL_WE_N,
 FL_WP_N,

 //////// GPIO //////////
 GPIO,
```

```
     //////// HSMC (LVDS) //////////
//   HSMC_CLKIN_N1,
//   HSMC_CLKIN_N2,
     HSMC_CLKIN_P1,
     HSMC_CLKIN_P2,
     HSMC_CLKIN0,
//   HSMC_CLKOUT_N1,
//   HSMC_CLKOUT_N2,
     HSMC_CLKOUT_P1,
     HSMC_CLKOUT_P2,
     HSMC_CLKOUT0,
     HSMC_D,
//   HSMC_RX_D_N,
     HSMC_RX_D_P,
//   HSMC_TX_D_N,
     HSMC_TX_D_P,
    //////// EXTEND IO //////////
    EX_IO
);

//=======================================================
//  PARAMETER declarations
//=======================================================


//=======================================================
//  PORT declarations
//=======================================================

////////////// CLOCK //////////
input                       CLOCK_50;
input                       CLOCK2_50;
input                       CLOCK3_50;
input                       ENETCLK_25;

////////////// Sma //////////
input                       SMA_CLKIN;
output                           SMA_CLKOUT;

////////////// LED //////////
output               [8:0]      LEDG;
output              [17:0]      LEDR;

////////////// KEY //////////
input           [3:0]      KEY;

////////////// SW //////////
input          [17:0]      SW;
```

```
////////////// SEG7 /////////
output              [6:0]      HEX0;
output              [6:0]      HEX1;
output              [6:0]      HEX2;
output              [6:0]      HEX3;
output              [6:0]      HEX4;
output              [6:0]      HEX5;
output              [6:0]      HEX6;
output              [6:0]      HEX7;

////////////// LCD /////////
output                         LCD_BLON;
inout          [7:0]      LCD_DATA;
output                         LCD_EN;
output                         LCD_ON;
output                         LCD_RS;
output                         LCD_RW;

////////////// RS232 /////////
output                         UART_CTS;
input                     UART_RTS;
input                     UART_RXD;
output                         UART_TXD;

////////////// PS2 /////////
inout                     PS2_CLK;
inout                     PS2_DAT;
inout                     PS2_CLK2;
inout                     PS2_DAT2;

////////////// SDCARD /////////
output                         SD_CLK;
inout                     SD_CMD;
inout          [3:0]      SD_DAT;
input                     SD_WP_N;

////////////// VGA /////////
output              [7:0]      VGA_B;
output                         VGA_BLANK_N;
output                         VGA_CLK;
output              [7:0]      VGA_G;
output                         VGA_HS;
output              [7:0]      VGA_R;
output                         VGA_SYNC_N;
output                         VGA_VS;

////////////// Audio /////////
input                     AUD_ADCDAT;
inout                     AUD_ADCLRCK;
```

```
inout                       AUD_BCLK;
output                           AUD_DACDAT;
inout                       AUD_DACLRCK;
output                           AUD_XCK;

////////////// I2C for EEPROM //////////
output                           EEP_I2C_SCLK;
inout                       EEP_I2C_SDAT;

////////////// I2C for Audio and Tv-Decode //////////
output                           I2C_SCLK;
inout                       I2C_SDAT;

////////////// Ethernet 0 //////////
output                           ENET0_GTX_CLK;
input                       ENET0_INT_N;
output                           ENET0_MDC;
inout                       ENET0_MDIO;
output                           ENET0_RST_N;
input                       ENET0_RX_CLK;
input                       ENET0_RX_COL;
input                       ENET0_RX_CRS;
input           [3:0]       ENET0_RX_DATA;
input                       ENET0_RX_DV;
input                       ENET0_RX_ER;
input                       ENET0_TX_CLK;
output              [3:0]       ENET0_TX_DATA;
output                           ENET0_TX_EN;
output                           ENET0_TX_ER;
input                       ENET0_LINK100;

////////////// Ethernet 1 //////////
output                           ENET1_GTX_CLK;
input                       ENET1_INT_N;
output                           ENET1_MDC;
inout                       ENET1_MDIO;
output                           ENET1_RST_N;
input                       ENET1_RX_CLK;
input                       ENET1_RX_COL;
input                       ENET1_RX_CRS;
input           [3:0]       ENET1_RX_DATA;
input                       ENET1_RX_DV;
input                       ENET1_RX_ER;
input                       ENET1_TX_CLK;
output              [3:0]       ENET1_TX_DATA;
output                           ENET1_TX_EN;
output                           ENET1_TX_ER;
input                       ENET1_LINK100;
```

```
////////////// TV Decoder 1 //////////
input                       TD_CLK27;
input           [7:0]       TD_DATA;
input                       TD_HS;
output                           TD_RESET_N;
input                       TD_VS;


////////////// USB OTG controller //////////
inout             [15:0]     OTG_DATA;
output            [1:0]      OTG_ADDR;
output                       OTG_CS_N;
output                       OTG_WR_N;
output                       OTG_RD_N;
input             [1:0]      OTG_INT;
output                       OTG_RST_N;
input             [1:0]      OTG_DREQ;
output            [1:0]      OTG_DACK_N;
inout                        OTG_FSPEED;
inout                        OTG_LSPEED;

////////////// IR Receiver //////////
input                       IRDA_RXD;

////////////// SDRAM //////////
output               [12:0]      DRAM_ADDR;
output                [1:0]      DRAM_BA;
output                           DRAM_CAS_N;
output                           DRAM_CKE;
output                           DRAM_CLK;
output                           DRAM_CS_N;
inout                 [31:0]     DRAM_DQ;
output                [3:0]      DRAM_DQM;
output                           DRAM_RAS_N;
output                           DRAM_WE_N;

////////////// SRAM //////////
output               [19:0]      SRAM_ADDR;
output                           SRAM_CE_N;
inout                [15:0]      SRAM_DQ;
output                           SRAM_LB_N;
output                           SRAM_OE_N;
output                           SRAM_UB_N;
output                           SRAM_WE_N;

////////////// Flash //////////
output               [22:0]      FL_ADDR;
output                           FL_CE_N;
inout                 [7:0]      FL_DQ;
```

```
output                          FL_OE_N;
output                          FL_RST_N;
input                           FL_RY;
output                          FL_WE_N;
output                          FL_WP_N;

////////////// GPIO /////////
inout          [35:0]      GPIO;

////////////// HSMC (LVDS) /////////

//input                         HSMC_CLKIN_N1;
//input                         HSMC_CLKIN_N2;
input                           HSMC_CLKIN_P1;
input                           HSMC_CLKIN_P2;
input                           HSMC_CLKIN0;
//output                        HSMC_CLKOUT_N1;
//output                        HSMC_CLKOUT_N2;
output                          HSMC_CLKOUT_P1;
output                          HSMC_CLKOUT_P2;
output                          HSMC_CLKOUT0;
inout                [3:0]      HSMC_D;
//input              [16:0]      HSMC_RX_D_N;
input              [16:0]      HSMC_RX_D_P;
//output             [16:0]      HSMC_TX_D_N;
output             [16:0]      HSMC_TX_D_P;

//////// EXTEND IO /////////
inout          [6:0]       EX_IO;

wire clockN,clockN1,clockN2,clockN3; //base do clock 1us
reg [2:0] contpwm1; //para a sincronizacao do divisor de frequencia
wire [2:0] mQ123; //vias dos sinais pwm
wire [63:0] mVsd,mVsq,mIsd,mIsq,mtt,mWm,mTeta;
reg [15:0] mCarga,mCC=150;
//------INTERFACE IHM-------------------------
wire [15:0] visualizador;
reg [15:0] temp_visualizador,temp_prog;
wire teclas,en,clock_display;
assign teclas = !KEY[3] | !KEY[2] | KEY[1] | KEY[0];
assign visualizador = temp_visualizador;
assign LEDG[0] = mQ123[0];
assign LEDG[2] = mQ123[1];
assign LEDG[4] = mQ123[2];
espera(teclas,en,CLOCK_50);
reg [4:0] c_funcao;
reg [1:0] funcao;   /* funcao 0 - ver velocidade
                                        funcao 1 - programar
Barramento CC
```

```
                                                funcao 2 - programar Carga*/

assign HEX5 = 7'b0001110;
assign HEX6 = 7'hff;
assign HEX7 = 7'hff;
SEG7_LUT_4 SE74(HEX0,HEX1,HEX2,HEX3,visualizador,CLOCK_50);
SEG7_LUT   hx4(.oSEG(HEX4),.iDIG(funcao));
always @ (posedge en)
begin
#100000
if(!KEY[1])temp_prog = temp_prog + 1;
if(!KEY[0])temp_prog = temp_prog - 1;
if(!KEY[3])
 begin
  c_funcao = c_funcao + 1;
  if(c_funcao == 5)
  begin
  funcao = funcao + 1'b1;
  c_funcao = 0;
  end
  case(funcao)
  7'd1:temp_prog = mCC;
  7'd2:temp_prog = mCarga;
  endcase
  end
end
      always @ (posedge clock_display)
      begin
            case(funcao)
            7'd0:temp_visualizador = mWm >> 21;
            7'd1:begin
            mCC = temp_prog;
            temp_visualizador = mCC;
            end
            7'd2:begin
            mCarga = temp_prog;
            temp_visualizador = mCarga;
            end
            7'd3:temp_visualizador = 0;
            endcase
      end
//------END INTERFACE IHM----------------------

div_freq
co1(.clk_in(CLOCK_50),.clk_out1(clockN),.contpwm1(contpwm1),.clk_out
3(clock_display),.clk_out0(clockN1),.clk_out2(clockN2),.clk_out4(clo
ckN3));
```

```
filtro_pwm
fp(.iQ123(GPIO[28:26]),.oQ123(mQ123),.clock50(CLOCK_50),.clock1(cloc
kN));
enviaGPIO0_DSP
egpiodsp(mTeta,mIsd,mIsq,mWm,mtt,mVsd,mVsq,GPIO,clockN);
motor
mot(.clock(clockN),.angulo(mTeta),.oVsd(mVsd),.oVsq(mVsq),.oIsd(mIsd
),.oIsq(mIsq),.oCe(mtt),.velocidade(mWm),.en(SW[17]),.q123(mQ123),.s
em_pwm(SW[16]),.hab_carga(SW[0]),.iVcc(mCC),.iCarga(mCarga),.clk1(cl
ockN1),.clk2(clockN2),.clk3(clockN3));

//sincronizacao
always @ (posedge GPIO[26])//mQ123[0]
begin
contpwm1 = contpwm1+1;
end

endmodule
```

B.2 Interface com o Controlador

```
// ---------------------------------------------------------------------
// enviaGPIO0_DSP.v
// ---------------------------------------------------------------------

//TENSOES DE BASE = 300 V
module
enviaGPIO0_DSP(iTeta,iIsd,iIsq,iWm,iCe,iVsd,iVsq,ioGPIO_0,clock);
input signed[63:0] iTeta,iIsd,iIsq,iWm,iCe,iVsd,iVsq;
inout [35:0] ioGPIO_0;
input clock;
wire requisita;
wire sinal;
wire signed[15:0] oDado;
reg [3:0] posicao;
reg signed [15:0] bufferSaida;
reg start;
reg signed [63:0] eBase,ed,eq,iBase,id,iq,ce,wBase,wm;

assign oDado = bufferSaida;
assign sinal = start&clock;
//--------PASSAGEM DOS VALORES PARA OS PINOS DA GPIO0
assign requisita = ioGPIO_0[31];
assign ioGPIO_0[15:3] = oDado[12:0];
assign ioGPIO_0[20] = oDado[13];
assign ioGPIO_0[22] = oDado[14];
```

```
assign ioGPIO_0[24] = oDado[15];
assign ioGPIO_0[34] = sinal;
//-------------------------------------------------
always @ (posedge clock)
begin
eBase = 6991;//(1/300)<<21; valor de base 300V
ed = iVsd*eBase;
eq = iVsq*eBase;
iBase =  83886;//(1/25)<<21; valor de base 25 A
id = iIsd*iBase;
iq = iIsq*iBase;
ce = iCe*iBase;
wBase = 2097; //(1/1000)<<21; valor base 1000
wm = iWm*wBase;
if(requisita&&!start)
begin
posicao = 4'b0000;
start = 1'b1;
end

if(start)
begin
posicao = posicao + 1'b1;
case(posicao)
4'b0001:begin
bufferSaida = iTeta >> 16;
end

4'b0010:begin
bufferSaida = id >> 26; //16bits pu
end

4'b0011:begin
bufferSaida = iq >> 26;//16bits pu
end

4'b0100:begin
bufferSaida = wm >> 26; //iWm valor real
end

4'b0101:begin
bufferSaida = ce >> 26;
end

4'b0110:begin
bufferSaida = ed >> 26;
end

4'b0111:begin
```

```
bufferSaida = eq >> 26;
end

4'b1000:begin
bufferSaida = 0;
start = 1'b0;
end
endcase
end
end

endmodule
```

B.3 Segurador PWM

```
// -------------------------------------------------------------------
// filtro_pwm.v
// -------------------------------------------------------------------

module filtro_pwm(iQ123,oQ123,clock50,clock1);
input [2:0] iQ123;
input clock50,clock1;
output [2:0] oQ123;
reg [2:0] rQ123;
reg [2:0] contClock1,contClock1_ant;
reg [7:0] Q1,Q2,Q3;

assign oQ123 = rQ123;

always @ (posedge clock50)
begin
if(iQ123[0]==1)Q1 = Q1 + 1;
if(iQ123[1]==1)Q2 = Q2 + 1;
if(iQ123[2]==1)Q3 = Q3 + 1;

      if(contClock1!=contClock1_ant)
      begin
            if(Q1>25)rQ123[0] = 1;
               else rQ123[0] = 0;
            if(Q2>25)rQ123[1] = 1;
               else rQ123[1] = 0;
            if(Q3>25)rQ123[2] = 1;
               else rQ123[2] = 0;
            Q1=0;
            Q2=0;
            Q3=0;
            contClock1_ant = contClock1;
```

```
          end
end

always @ (posedge clock1)
begin
contClock1 = contClock1 + 1;
end
endmodule
```

B.4 Simulação da Planta

```
// ------------------------------------------------------------------
// motor.v
// ------------------------------------------------------------------

module
motor(angulo,velocidade,oVsd,oVsq,oIsd,oIsq,oCe,clock,en,q123,iVcc,i
Carga,sem_pwm,hab_carga);
input clock,en,sem_pwm,hab_carga;
input [15:0] iVcc,iCarga;
input [2:0] q123;
output signed [63:0] oVsd,oVsq,velocidade,oIsd,oIsq,oCe;
output [30:0] angulo;
//NESTA VERSAO O h ESTÁ EM 32 BITS
//PARA GARANTIR A ROBUSTEZ EM ALTERACOES AS VARIAVEIS FORAM
DIMENSIONADAS NO PADRAO LONG 64bits
parameter bv = 27; //quantidade de bits para a parte fracionaria
parameter bh = 32; //quantidade de bits para o passo de cálculo
integer Q1 = bh;//qh
integer Q2 = bv;
integer Q3 = bh-bv;
integer Q4 = bv+bh-32;
integer Q5 = bv - 21;
//CONSTANTES
//DA MAQUINA
reg signed [63:0] flm = 18253611;       //0.068*Q28;
reg signed [63:0] p = 4;                //4 pares de polos
reg signed [63:0] rs =  100193533;      //0.7465*Qv
reg signed [63:0] rcarga= 0;
reg signed [63:0] rpmTorad = 219613;    //Q21*0.104719755120
reg signed [63:0] pi=1048576;           //Q21*pi
reg signed [63:0] Q21 =  2097152;
reg signed [63:0] hf =  4295;           //1e-6*Q32
reg signed [63:0] ld=9792525;           //0.00228*Q32
reg signed [63:0] lq=10909216;          //0.00254*Q32
reg signed [63:0] jj=15118284;              //0.22e-3*Q36
reg signed [63:0] cm=0;                    //Conj. mecanico
reg signed [63:0] ff=0;                     //Coef. At.
```

```
//VARIAVEIS GERAIS
reg signed [63:0] vsd=0,vsq=0,vsda=0,vsqa=0;
reg signed [155:0] temp_ome = 0;
reg signed [97:0] temp_ome1=0;
reg signed [63:0] isd=0;
reg signed [63:0] isq=0;
reg signed [63:0] isd_a=0;
reg signed [63:0] isq_a=0;
reg signed [63:0] ed=0;
reg signed [63:0] eq=0;
reg signed [63:0] wm = 0;
reg signed [63:0] wr = 0;
reg [30:0] ome=0;    //angulo
//VARIAVEIS DOS CALCULOS DA FCEM
reg signed [127:0] eq1_1;
//VARIAVEIS DOS CALCULOS CORRENTES
reg signed [127:0] isd1p1 = 0;
reg signed [63:0] isd1p = 0;
reg signed [63:0] isd1 = 0;
reg signed [147:0] isd2p1a1 = 0;
reg signed [127:0] isd2p1a = 0;
reg signed [63:0] isd2p = 0;
reg signed [127:0] isd2p1 = 0;
reg signed [127:0] isq22=0;
reg signed [63:0] isd2 = 0;
reg signed [63:0] isd3 = 0;
reg signed [63:0] ld32 = 0;

reg signed [127:0] isq1p1 = 0;
reg signed [63:0] isq1p = 0;
reg signed [63:0] isq1 = 0;
reg signed [147:0] isq2p1a1 = 0;
reg signed [127:0] isq2p1a = 0;
reg signed [63:0] isq2p = 0;
reg signed [127:0] isq2s = 0;
reg signed [63:0]  isq2 = 0;
reg signed [63:0]  isq3 = 0;
reg signed [63:0]  lq32 = 0;
//VARIAVEIS DO CONJUGADO E MOVIMENTO
reg signed [63:0] cel=0;
reg signed [63:0] cel1=0,cel1p1=0,cel1p=0,cel2=0,cel3=0;
reg signed [127:0] cel11=0,cel11p=0,cel2p=0,acel=0;
reg signed [63:0] wmff=0,wmp1=0,wmp2=0,wmt=0;
reg signed [127:0] wmp3=0;
reg signed [63:0] wma=0;
//CONVERSOR
reg q1;
reg q2;
reg q3;
```

```
reg signed[63:0] V10;           //Tensao de polo V1
reg signed[63:0] V20;           //Tensao de polo V2
reg signed[63:0] V30;           //Tensao de polo V3
reg signed[127:0] V1n;
reg signed[79:0] vai;           //Tensao de saida Va
reg signed[79:0] vbi;           //Tensao de saida Vb
reg signed[79:0] vci;           //Tensao de saida Vc
reg signed[63:0] Ev12=150;
reg signed [79:0] vsalfac1;
reg signed [127:0] vsalfac2,vsbetac1,vsbetac2,vsalfac3;
reg signed [127:0] vsd1,vsd2,vsq1,vsq2;
reg signed[63:0] tpw=42950;//100e-6
reg signed[31:0] t;
reg signed[22:0] cos2,sen2;
wire [20:0] angome;
reg signed[79:0] valfa,vbeta,valfamed,vbetamed,valfam,vbetam;
reg signed [79:0] vsd1med,vsd2med,vsq1med,vsq2med,vsdmed,vsqmed;
//VARIAVEIS DIVERSAS
reg signed [63:0] Q25=33554432;
reg signed [63:0] rq23=53510;  //sqrt(2/3)*2^16
reg signed [63:0] v12=32768;   //(1/2)*2^16
reg signed [63:0] rq32=56756;  //sqrt(3)/2 *2^16
reg signed [63:0] v13=21845;   //(1/3)*2^16;   //(1/3)*2^21
reg signed [63:0] rq3223=46341;//16

assign angome = ome >> 10;

assign angulo = ome;
assign oVsd = vsdmed;//>>Q5;
assign oVsq = vsqmed;//>>Q5;
assign velocidade = wm>>Q5;
assign oIsd = isd>>Q5;
assign oIsq = isq>>Q5;
assign oCe = cel >>Q5;

coss cosn2(.cos(cos2),.angulo(angome),.clock(clock));
seno senn2(.sen(sen2),.angulo(angome),.clock(clock));

initial

begin
isd =0;
isq = 0;
isd_a=0;
isq_a=0;

end

//LOOP PRINCIPAL
```

```
always @ (negedge clock)
begin
if (en==1)
begin

  //*****************************************************
  //Sistema de Potencia
  //*****************************************************
  // Conversor ideal

      q1 = q123[0];
      q2 = q123[1];
      q3 = q123[2];
      Ev12 = iVcc;
       if(q1==1'b1)V10 = Ev12;
                else V10= -Ev12;   //(((((2*q1))-1)*((Ev12)));
        if(q2==1'b1)V20 = Ev12;       //(((((2*q2))-1)*((Ev12)));
                else V20 = -Ev12;
       if(q3==1'b1)V30 = Ev12;       //(((((2*q3))-1)*((Ev12)));
                else V30 = -Ev12;


                V1n = ((-(V10 + V20 + V30))*v13)>>16;
     // Tensao aplicada ao motor
                vai = (V10 + V1n);
                vbi = (V20 + V1n);
                vci = (V30 + V1n);

       //Transformacao ABC para Alfa Beta
                    vsalfac2 = vbi+vci;
                    vsalfac1 = (vsalfac2*v12)>>16;
                    vsalfac3 = vai - vsalfac1;
                    valfa = (rq23*vsalfac3);
                    vsbetac1 = vbi-vci;
                    vbeta = (vsbetac1*rq3223);

       //Transformacao Alfa Beta para dq
                    vsd1 = (valfa*cos2)>>16;
                    vsd2 = (vbeta*sen2)>>16;
           vsda = vsd1+vsd2;
                    vsq1 = (valfa*sen2)>>16;
                    vsq2 = (vbeta*cos2)>>16;
           vsqa = ((-vsq1) + vsq2);
           vsd = vsda << 6; //Adequacao para Q27
           vsq = vsqa << 6; //Adequacao para Q27

//if(sem_pwm==1)
//begin
// vsd = 0;
```

```
// vsq = iVcc << bv;
//end
cm = 0;//reseta conjugado mecanico
if(hab_carga==1)cm =  iCarga*1342208; //adiciona conjugado mecanico


      isd_a=isd;
      isq_a=isq;
      wma = wm;

//*******************************************************
//Sistema Elétrico
//*******************************************************

  //Equacionamento da fcem
      ed = 0;
      eq = (wr*flm)>> 28;

  //Equacionamento da corrente Isd
  //isd = (isd_a*(ld-(rs+rcarga)*h)+h*(ed + lq*isq_a*wr ))/ld;
      isd1p1 = rs;
      isd1p = (isd1p1*hf)>> Q2;   //Q27
      isd1 = (ld-isd1p);          //Q27
      isd2p1a = (lq*isq_a) >> Q2;
      isd2p = (isd2p1a*wr) >> Q1; //Q27
      isd2p1 = (vsd+ed + isd2p);
      isd2 = (hf*isd2p1);         //Q54
      isd3 = (isd_a*isd1);        //Q54
      isd = (isd3+isd2)/ld;       //Q54-q27=q27

  //Equacionamento da corrente Isq
  //isq = (isq_a*(lq-(rs+rcarga)*h)+h*(-eq - ld*isd_a*wr ))/lq;
      isq1p1 = rs;
      isq1p = (isq1p1*hf) >> Q2;
      isq1 = (lq-isq1p);
      isq2p1a = (ld*isd_a) >> Q2;
      isq2p = (isq2p1a*wr) >> Q1;
      isq22 =vsq -eq - isq2p;
      isq2 = (hf*isq22);
      isq3 = (isq_a*isq1);
      isq = (isq3+isq2)/lq;


//*******************************************************
//Sistema mecânico
//*******************************************************

      //Equacionamento do conjugado
      //cel = p*(flm*isq+(ld-lq)*isd*isq);
```

```
  cel11 = isd*isq;
  cel1 = (cel11)>>Q2;
  cel1p1 = ld-lq;
  cel11p = cel1*cel1p1;
  cel1p = (cel11p)>>Q2;
  cel2p = flm*isq;
  cel2 = (cel2p)>>23;
  cel3 = cel1p + cel2;
  acel = (p*cel3);
  cel = acel >> 5;


//Equacionamento mecanico de velocidade
   //wm = wma + h*(cel-ff*wm-cm)/jj;
  wmp1 = cel - wmff - cm;
  wmp2 = (wmp1*hf)<<4;
  wmp3 = wmp2/jj;
  wmt = (wmp3);
  wm = wma + wmt;

       wr = p*wm;

//Calculo do angulo
   //ome = ome + wr*h;
  temp_ome = (wr*hf);
  temp_ome1 = (temp_ome*333772)>>49;
  ome = ome + temp_ome1;//angulo gira em 31 bits

 /*
 //CALCULO DAS TENSOES MEDIAS
       t = t + hf;
       valfam = valfam + valfa;
       vbetam = vbetam + vbeta;
       if(t>=tpw)
            begin
                 valfamed = (valfam*655)>>16; //655 = 1/100 Q16
                 vbetamed = (vbetam*655)>>16;

                 vsd1med = (valfamed*cos2)>>16;
                 vsd2med = (vbetamed*sen2)>>16;
        vsdmed = vsd1med+vsd2med;
                 vsq1med = (valfamed*sen2)>>16;
                 vsq2med = (vbetamed*cos2)>>16;
        vsqmed = ((-vsq1med) + vsq2med);

                 valfam = 0;
                 vbetam = 0;
                 t = 0;
            end
```

```
          */
end
end

endmodule
```

B.5 Divisor de Frequência

```
// ------------------------------------------------------------------
// div_freq.v
// ------------------------------------------------------------------

//Divisor de frequencia
module div_freq(clk_in, clk_out0, clk_out1, clk_out2,
clk_out3,clk_out4,contpwm1);

input clk_in;
input [2:0] contpwm1;
output clk_out0;
output clk_out1;
output clk_out2;
output clk_out3;
output clk_out4;

reg [2:0] contpwm1_ant;
reg[24:0] div0_50;
reg[24:0] div1_50;
reg[24:0] div2_50;
reg[24:0] div3_50;

reg[24:0] acc0;
reg[24:0] acc1;
reg[24:0] acc2;
reg[24:0] acc3;

parameter f0 = 1000000;
parameter f1 = 1000000;          //Para os ciclos PWM a 1 MHz
parameter f2 = 1000000;
parameter f3 = 4;
//f: frequencia desejada = 100Hz
//acc_cont = 25000000/(frequencia desejada)

always@(posedge clk_in)
begin
     /***********CLOCK_0*************/
//   div0_50 = div0_50 + 1;
```

```
//    acc0 = 25000000/f0;
//    if(div0_50 >= acc0)
//    begin
//         div0_50 = 0;
//         clk_out0 = clk_out0^1;
//    end

      /***********CLOCK_1*************/
      div1_50 = div1_50 + 1;
      acc1 = 25000000/f1;
      if(div1_50 >= acc1)
      begin
           div1_50 = 0;
           clk_out1 = clk_out1^1;
      end//sincronizacao
      if (contpwm1!=contpwm1_ant)
      begin
      div1_50 = 0;
      clk_out1 = 1;
      contpwm1_ant = contpwm1;
      end
      clk_out0 = clk_out1;
      clk_out2 = clk_out1;
      clk_out4 = clk_out1;
      /***********CLOCK_2*************/
//    div2_50 = div2_50 + 1;
//    acc2  = 25000000/f2;
//    if(div2_50 >= acc2)
//    begin
//         div2_50 = 0;
//         clk_out2 = clk_out2^1;
//    end

      /***********CLOCK_3*************/
      div3_50 = div3_50 + 1;
      acc3  = 25000000/f3;
      if(div3_50 >= acc3)
      begin
           div3_50 = 0;
           clk_out3 = clk_out3^1;
      end
end

endmodule
```

APÊNDICE C

Metodologia para Redução do Tempo de Cálculo

O Capítulo 5 apresentou um método para a implementação de um modelo de máquina síncrona a ímã permanente simulado em FPGA utilizando, em Verilog, um único bloco *always* para a descrição das equações do sistema. O método é simples e de fácil implementação. Mas, como as equações são resolvidas de forma sequencial, o aumento do número de equações amplia linearmente o tempo necessário para o FPGA calcular o sistema.

Se o tempo para o FPGA calcular as equações for maior que o passo de cálculo, uma alternativa é dividir o equacionamento do sistema em blocos, e distribuir o equacionamento para serem executados em paralelo, conforme mostra a Figura 50. Os blocos representados pelos Sistemas de Potência, Sistema Elétrico e Sistema mecânico, são executados em paralelo. O equacionamento das correntes, na figura, também foi separado, um bloco para o eixo d e outro bloco para o eixo q. Com isto, divide-se as equações diminuindo o tempo total de cálculo.

Os blocos podem ser implementados de duas formas: cada um em um módulo Verilog, conectados apropriadamente pelas entradas e saídas; ou dentro de um mesmo módulo separados por blocos *always*. Neste trabalho, os blocos foram implementados dentro de um único módulo.

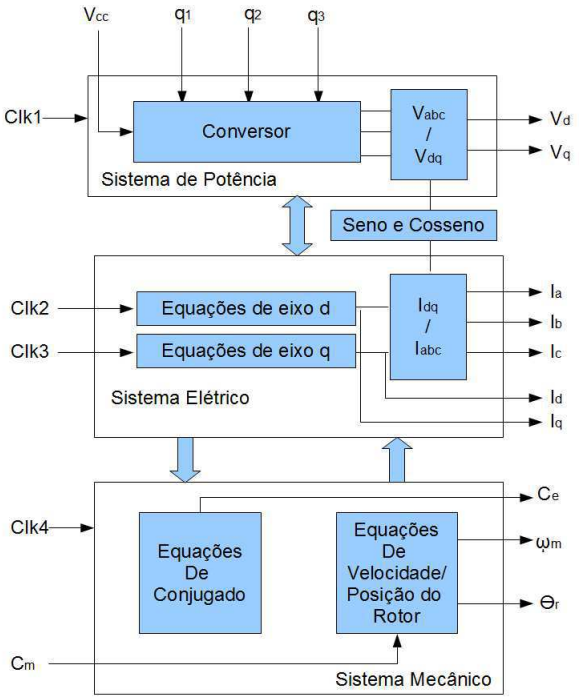


Figura 50. Diagrama de blocos para a simulação com tempo de cálculo reduzido.

Como cada bloco é executado de forma independente, é necessário que a comunicação entre os blocos esteja sincronizada por um sinal de *clock* de referência, que neste caso é o *clock* global sincronizado com o *hardware* dedicado de controle. Cada bloco tem variáveis de entrada e variáveis de saída. Os dados de entrada dos blocos estão armazenados em variáveis intermediárias, que são atualizadas a cada borda de descida do *clock*. Por exemplo, na borda de subida, inicia-se o processamento dos elementos, as equações presentes em cada bloco são calculadas. Na borda de descida, os valores computados são copiados para as variáveis intermediárias, atualizando assim, os dados de entrada de para o próximo ciclo de *clock*. A Figura 51 ilustra este processo.

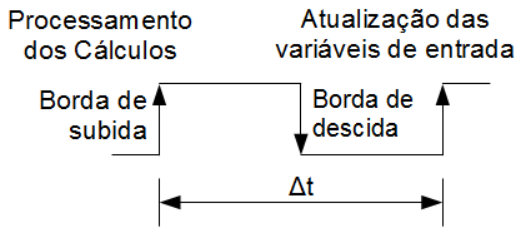


Figura 51. Esquema do gatilho de sincronização.

Para implementar o método a partir de um único bloco *always* (apêndice B.4), deve-se seguir os seguintes passos:

1. Dividir em blocos o equacionamento da máquina:

O equacionamento do sistema foi divido em quatro blocos *always* dentro de um único módulo Verilog:

a) Sistema de potência: a partir do estado das chaves do PWM são calculadas as tensões de fase e transformadas para o referencial dq.

b) Equações do eixo d da máquina: são calculadas a tensão induzida e a corrente de eixo d.

c) Equações do eixo q da máquina: são calculadas a tensão induzida e a corrente de eixo q.

d) Sistema mecânico: são calculados o torque eletromagnético, velocidade e posição do rotor da máquina.

2. Gerar os sinais de *clock* apropriados:

O módulo Verilog que contém o equacionamento deve ter em suas entradas pelo menos quatro sinais de *clock* distintos, um para cada bloco *always.* Estes sinais devem ser gerados por um módulo instanciado no módulo de nível superior. Como foi escolhido o mesmo passo de cálculo para os sistemas, a frequência e a fase dos quatro sinais de *clock* devem ser as mesmas.

Apesar de terem as mesmas características, não se deve gerar apenas um sinal de *clock* e atribuí-lo para as outras vias de *clock* dentro do módulo da simulação. Nas saídas do módulo divisor de frequência, devem estar presentes quatro vias distintas para acionar os blocos.

3. Identificar e implementar as variáveis intermediárias:

As variáveis correspondentes aos valores computados em cada bloco devem ser identificadas. Uma variável intermediária representa o valor de uma variável computada dentro dos outros blocos. Após identificadas, deve-se implementar o bloco *always* que atualiza a variável. Este bloco tem o mesmo sinal de *clock* da variável computada, porém, acionado pela borda de descida.

Seguindo estes passos, o modelo implementado no capítulo 5 foi adequado para realizar operações em paralelo. A utilização dos recursos do FPGA manteve a mesma porcentagem. Entretanto, a frequência máxima de *clock* subiu de 2,36 MHz para 8,25 MHz, o que permite simulações com passo de cálculo menores que 125ns.

Este método apresentado visa a simulação dos três sistemas (potência, elétrico e mecânico) sendo executados com o mesmo passo de cálculo. O código fonte da aplicação está demonstrada na seção B.1.

C.1 Simulação da Planta em Paralelo

```
// ------------------------------------------------------------------
// motor.v
// ------------------------------------------------------------------


module
motor(angulo,velocidade,oVsd,oVsq,oIsd,oIsq,oCe,clock,en,q123,iVcc,i
Carga,sem_pwm,hab_carga,clk1,clk2,clk3);
input clock,en,sem_pwm,hab_carga;
input clk1,clk2,clk3;
input [15:0] iVcc,iCarga;
input [2:0] q123;
output signed [63:0] oVsd,oVsq,velocidade,oIsd,oIsq,oCe;
output [30:0] angulo;
//NESTA VERSAO O h ESTÁ EM 32 BITS
//PARA GARANTIR A ROBUSTEZ EM ALTERACOES AS VARIAVEIS FORAM
//DIMENSIONADAS NO PADRAO LONG 64bits
parameter bv = 27; //quantidade de bits para a parte fracionaria
parameter bh = 32; //quantidade de bits para o passo de cálculo
integer Q1 = bh;//qh
integer Q2 = bv;
integer Q3 = bh-bv;
integer Q4 = bv+bh-32;
integer Q5 = bv - 21;
//CONSTANTES
//DA MAQUINA
reg signed [63:0] flm = 18253611;       //0.068*Q28;
reg signed [63:0] p = 4;                //4 pares de polos
reg signed [63:0] rs =  100193533;      //0.7465*Qv
reg signed [63:0] rcarga= 0;
reg signed [63:0] rpmTorad = 219613;    //Q21*0.104719755120
reg signed [63:0] pi=1048576;           //Q21*pi
reg signed [63:0] Q21 =  2097152;
reg signed [63:0] hf =  4295;           //1e-6*Q32
reg signed [63:0] ld=9792525;           //0.00228*Q32
reg signed [63:0] lq=10909216;          //0.00254*Q32
reg signed [63:0] jj=15118284;              //0.22e-3*Q36
reg signed [63:0] cm=0;                                 //Conj.
mecanico
reg signed [63:0] ff=0;                                 //Coef. At.
//VARIAVEIS GERAIS
reg signed [63:0] vsd=0,vsq=0,vsda=0,vsqa=0;
```

```
reg signed [155:0] temp_ome = 0;
reg signed [97:0] temp_ome1=0;
reg signed [63:0] isd=0;
reg signed [63:0] isq=0;
reg signed [63:0] isd_a=0;
reg signed [63:0] isq_a=0;
reg signed [63:0] ed=0;
reg signed [63:0] eq=0;
reg signed [63:0] wm = 0;
reg signed [63:0] wr = 0;
reg [30:0] ome=0;//angulo
//VARIAVEIS DOS CALCULOS DA FCEM
reg signed [127:0] eq1_1;
//VARIAVEIS DOS CALCULOS CORRENTES
reg signed [127:0] isd1p1 = 0;
reg signed [63:0] isd1p = 0;
reg signed [63:0] isd1 = 0;
reg signed [147:0] isd2p1a1 = 0;
reg signed [127:0] isd2p1a = 0;
reg signed [63:0] isd2p = 0;
reg signed [127:0] isd2p1 = 0;
reg signed [127:0] isq22=0;
reg signed [63:0] isd2 = 0;
reg signed [63:0] isd3 = 0;
reg signed [63:0] ld32 = 0;

reg signed [127:0] isq1p1 = 0;
reg signed [63:0] isq1p = 0;
reg signed [63:0] isq1 = 0;
reg signed [147:0] isq2p1a1 = 0;
reg signed [127:0] isq2p1a = 0;
reg signed [63:0] isq2p = 0;
reg signed [127:0] isq2s = 0;
reg signed [63:0]  isq2 = 0;
reg signed [63:0]  isq3 = 0;
reg signed [63:0]  lq32 = 0;
//VARIAVEIS DO CONJUGADO E MOVIMENTO
reg signed [63:0] cel=0;
reg signed [63:0] cel1=0,cel1p1=0,cel1p=0,cel2=0,cel3=0;
reg signed [127:0] cel11=0,cel11p=0,cel2p=0,acel=0;
reg signed [63:0] wmff=0,wmp1=0,wmp2=0,wmt=0;
reg signed [127:0] wmp3=0;
reg signed [63:0] wma=0;
//CONVERSOR
reg q1;
reg q2;
reg q3;
reg signed[63:0] V10;           //Tensao de polo V1
reg signed[63:0] V20;           //Tensao de polo V2
```

```
reg signed[63:0] V30;           //Tensao de polo V3
reg signed[127:0] V1n;
reg signed[79:0] vai;           //Tensao de saida Va
reg signed[79:0] vbi;           //Tensao de saida Vb
reg signed[79:0] vci;           //Tensao de saida Vc
reg signed[63:0] Ev12=150;
reg signed [79:0] vsalfac1;
reg signed [127:0] vsalfac2,vsbetac1,vsbetac2,vsalfac3;
reg signed [127:0] vsd1,vsd2,vsq1,vsq2;
reg signed[63:0] tpw=42950;//100e-6
reg signed[31:0] t;
reg signed[22:0] cos2,sen2;
wire [20:0] angome;
reg signed[79:0] valfa,vbeta,valfamed,vbetamed,valfam,vbetam;
reg signed [79:0] vsd1med,vsd2med,vsq1med,vsq2med,vsdmed,vsqmed;
//OUTRAS VARIAVEIS
reg signed [63:0] Q25=33554432;
reg signed [63:0] rq23=53510; //sqrt(2/3)*2^16
reg signed [63:0] v12=32768;  //(1/2)*2^16
reg signed [63:0] rq32=56756; //sqrt(3)/2 *2^16
reg signed [63:0] v13=21845;   //(1/3)*2^16;   //(1/3)*2^21
reg signed [63:0] rq3223=46341;//16
reg signed [63:0] vsdh=0,vsqh=0,isdh=0,isqh=0,wrh=0;
assign angome = ome >> 10;

assign angulo = ome;
assign oVsd = vsdmed;//>>Q5;
assign oVsq = vsqmed;//>>Q5;
assign velocidade = wm>>Q5;
assign oIsd = isd>>Q5;
assign oIsq = isq>>Q5;
assign oCe = cel >>Q5;

coss cosn2(.cos(cos2),.angulo(angome),.clock(!clock));
seno senn2(.sen(sen2),.angulo(angome),.clock(!clock));

initial

begin
isd =0;
isq = 0;
isd_a=0;
isq_a=0;

end

//ATUALIZAÇÃO DAS VARIÁVEIS INTERMEDIÁRIAS
always @ (negedge clock)
begin
```

```
vsdh = vsd;
vsqh = vsq;
end
always @ (negedge clk1)
begin
isdh = isd;
end
always @ (negedge clk3)
begin
isqh = isq;
end
always @ (negedge clk2)
begin
wrh = wr;
end

//*******************************************************
//SISTEMA DE POTENCIA
//*******************************************************
  // CONVERSOR IDEAL
always @ (posedge clock)
begin
//if (en==1)
//begin

      q1 = q123[0];
      q2 = q123[1];
      q3 = q123[2];
      Ev12 = iVcc;
        if(q1==1'b1)V10 = Ev12;
                  else V10= -Ev12;   //(((((2*q1))-1)*((Ev12)));
     if(q2==1'b1)V20 = Ev12;//(((((2*q2))-1)*((Ev12)));
                  else V20 = -Ev12;
        if(q3==1'b1)V30 = Ev12;//(((((2*q3))-1)*((Ev12)));
                  else V30 = -Ev12;


                  V1n = ((-(V10 + V20 + V30))*v13)>>16;
     // Tensao aplicada ao motor
                  vai = (V10 + V1n);
                  vbi = (V20 + V1n);
                  vci = (V30 + V1n);

        //Transformacao ABC para Alfa Beta
                        vsalfac2 = vbi+vci;
                        vsalfac1 = (vsalfac2*v12)>>16;
                        vsalfac3 = vai - vsalfac1;
                        valfa = (rq23*vsalfac3);
                        vsbetac1 = vbi-vci;
```

```
                    vbeta = (vsbetac1*rq3223);

        //Transformacao Alfa Beta para dq
                    vsd1 = (valfa*cos2)>>16;
                    vsd2 = (vbeta*sen2)>>16;
            vsda = vsd1+vsd2;
                    vsq1 = (valfa*sen2)>>16;
                    vsq2 = (vbeta*cos2)>>16;
            vsqa = ((-vsq1) + vsq2);
            vsd = vsda << 6; //Adequacao para Q27
            vsq = vsqa << 6; //Adequacao para Q27


//if(sem_pwm==1)
//begin
// vsd = 0;
// vsq = iVcc << bv;
//end
//end
end //TERMINA O BLOCO DO CONVERSOR DE POTENCIA


//*******************************************************
//SISTEMA ELETRICO
//*******************************************************
  //EQUACIONAMENTO EIXO D
always @ (posedge clk1)
begin


      isd_a=isd;

      ed = 0;


 //Equacionamento da corrente Isd
  //isd = (isd_a*(ld-(rs+rcarga)*h)+h*(ed + lq*isq_a*wr ))/ld;
      isd1p1 = rs;
      isd1p = (isd1p1*hf)>>(Q2); //Q27
      isd1 = (ld-isd1p);   //Q27
      isd2p1a = (lq*isq_a)>>(Q2);
      isd2p = (isd2p1a*wr)>>Q1; //Q27
      isd2p1 = (vsd+ed + isd2p);
      isd2 = (hf*isd2p1);  //Q54
      isd3 = (isd_a*isd1); //Q54
      isd = (isd3+isd2)/ld;  //Q54-q27=q27
```

```
end
//EQUACIONAMENTO EIXO Q
always @ (posedge clk3)
begin
    isq_a=isq;
    eq = (wrh*flm)>> 28;

  //Equacionamento da corrente Isq
  //isq = (isq_a*(lq-(rs+rcarga)*h)+h*(-eq - ld*isd_a*wr ))/lq;
      isq1p1 = rs;
      isq1p = (isq1p1*hf)>>(Q2);
      isq1 = (lq-isq1p);
      isq2p1a = (ld*isd_a)>>(Q2);
      isq2p = (isq2p1a*wr)>>Q1;
      isq22 = vsq -eq - isq2p;
      isq2 = (hf*isq22);
      isq3 = (isq_a*isq1);
      isq = (isq3+isq2)/lq;

end
//*****************************************************
//Sistema mecânico
//*****************************************************
always @ (posedge clk2)
begin
        wma = wm;
       //Equacionamento do conjugado
       //cel = p*(flm*isq+(ld-lq)*isd*isq);
      cel11 = isd*isq;
      cel1 = (cel11)>>Q2;
      cel1p1 = ld-lq;
      cel11p = cel1*cel1p1;
      cel1p = (cel11p)>>Q2;
      cel2p = flm*isq;
      cel2 = (cel2p)>>23;
      cel3 = cel1p + cel2;
      acel = (p*cel3);
      cel = acel >> 5;

 cm = 0;//reseta conjugado mecanico
if(hab_carga==1)cm =  iCarga*1342208;//adiciona conjugado mecanico


    //Equacionamento mecanico de velocidade
       //wm = wma + h*(cel-ff*wm-cm)/jj;
      wmp1 = cel - wmff - cm;
      wmp2 = (wmp1*hf)<<4;
      wmp3 = wmp2/jj;
      wmt = (wmp3);
```

```
        wm = wma + wmt;

             wr = p*wm;

      //Calculo do angulo
         //ome = ome + wr*h;
        temp_ome = (wr*hf);
        temp_ome1 = (temp_ome*333772)>>49;
        ome = ome + temp_ome1;//angulo gira em 31 bits


       /*
       //CALCULO DAS TENSOES MEDIAS
             t = t + hf;
             valfam = valfam + valfa;
             vbetam = vbetam + vbeta;
             if(t>=tpw)
                  begin
                       valfamed = (valfam*655)>>16; //655 = 1/100 Q16
                       vbetamed = (vbetam*655)>>16;

                       vsd1med = (valfamed*cos2)>>16;
                       vsd2med = (vbetamed*sen2)>>16;
              vsdmed = vsd1med+vsd2med;
                       vsq1med = (valfamed*sen2)>>16;
                       vsq2med = (vbetamed*cos2)>>16;
              vsqmed = ((-vsq1med) + vsq2med);

                       valfam = 0;
                       vbetam = 0;
                       t = 0;
                  end
             */
end
endmodule
```

APÊNDICE D

Instruções para a Utilização da Plataforma

Esta seção descreve as instruções básicas para a utilização da plataforma. A FIGURA mostra a plataforma constituída pelo FPGA e DSP conectados via cabo flat de 40 vias. A Tabela 24 mostra a configuração do cabo que une os dispositivos.

Tabela 24. Pinagem do cabo de conexão entre FPGA e DSP.

	FPGA DE2		**DSP TMS320F28335**	
Descrição	GPIO1	PINO	Pino P8 (DSP)	Sinal Placa (DSP)
dado[0]	3	4	12	GPIO3
dado[1]	4	5	13	GPIO4
dado[2]	5	6	14	GPIO5
dado[3]	6	7	16	GPIO6
dado[4]	7	8	21	GPIO7
dado[5]	8	9	30	GPIO8
dado[6]	9	10	31	GPIO9
dado[7]	10	13	32	GPIO10
dado[8]	11	14	29	GPIO11
dado[9]	12	15	37	GPIO12
dado[10]	13	16	17	GPIO13
dado[11]	14	17	5	GPIO14
dado[12]	15	18	22	GPIO15
dado[13]	20	23	23	GPIO16
dado[14]	22	25	24	GPIO17
dado[15]	24	27	25	GPIO18
chave q1	26	31	9	GPIO0
chave q2	27	32	10	GPIO1
chave q3	28	33	11	GPIO2
pedido	31	36	18	ECAP1
GND		12,30	19,20,39,40	GND

Aproveitando os recursos disponíveis na placa DE2, como os displays, botões do tipo *push-button* e *toggle switches*, foi programado no FPGA uma interface homem máquina que

permite ao usuário interagir com a simulação. Nesta interface é possível visualizar a velocidade da máquina, programar a tensão no barramento CC do conversor e inserir uma carga mecânica no motor. A Tabela 25 mostra estas funções da interface.

Tabela 25. Interface Homem Máquina da Placa FPGA

Função (display)	Descrição
F0	Visualização da velocidade (rad/s)
F1	Visualização e programação do Vcc/2 (V)
F2	Visualização e programação da carga mecânica (N.m)

A Tabela 26 indica as funções das teclas na placa DE2. Através da tecla KEY[3], altera-se a visualização das funções. As teclas KEY[1] e KEY[0] permitem alterar valores nas funções. A SW[17] tem a função de habilitar a simulação, ou seja, no estado 0 a simulação fica em espera, no estado 1 a simulação irá responder aos estímulos externos. SW[0] insere a carga mecânica no eixo do motor, no estado 0 a carga é igual a zero, ja no estado 1, é aplicado ao motor a carga programada na função F2.

Tabela 26. Funções das teclas na placa DE2

Tecla	Descrição
KEY[3]	Altera função
KEY[1]	Aumenta Valor (para funções F1 e F2)
KEY[0]	Diminui Valor (para funções F1 e F2)
SW[17]	Habilita Simulação. 0 - em espera 1 - ativa
SW[16]	Modo: 0 - normal 1 - sem PWM
SW[0]	Habilita Carga: 0 - sem carga mecânica 1 - com carga mecânica

O SW[16] altera o modo de operação da simulação. No estado 0, a simulação está em operação normal. A Figura 52 mostra um diagrama do modo normal, neste modo de operação o motor é controlado pelo DSP através dos sinais PWM.

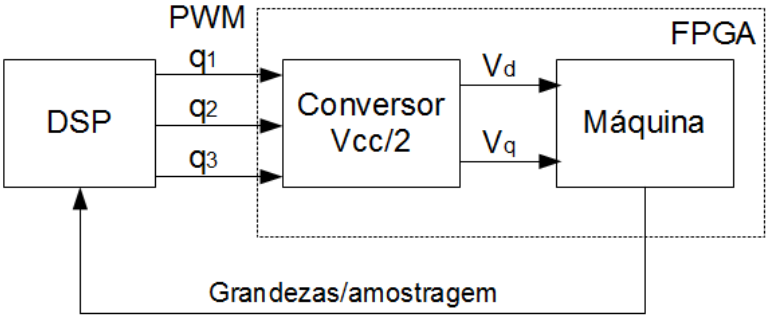


Figura 52. Simulação no modo normal.

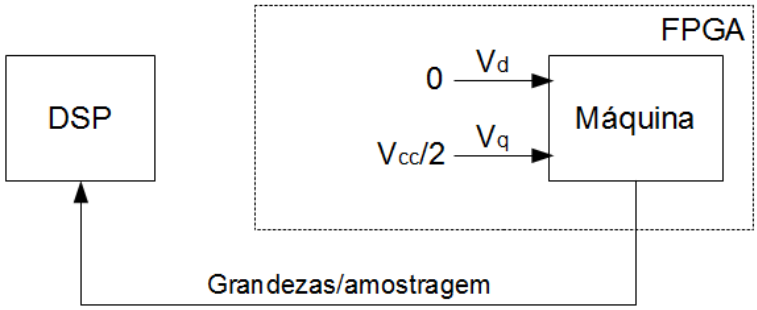


Figura 53. Simulação no modo sem PWM.

A Figura 53 apresenta o modo sem PWM. Neste modo de operação a máquina não é acionada pelo DSP. O DSP apenas faz a amostragem das grandezas para a visualização. O conversor é desabilitado e a tensão aplicada ao motor corresponde a zero no eixo d e Vcc/2 no eixo q. Este modo é util para realizar testes e analisar a resposta do modelo do motor.