MARCELO FAVORETTO CASTOLDI

Protótipo Virtual da Estratégia DTC Aplicada a Motores de Indução
Usando Linguagem VHDL

Dissertação apresentada à Escola
de Engenharia de São Carlos da
Universidade de São Paulo, para
obtenção do Título de Mestre em
Engenharia Elétrica.

Área de Concentração: Sistemas
Dinâmicos

Orientador: Prof. Dr. Manoel Luís
de Aguiar

São Carlos
2006

AUTORIZO A REPRODUÇÃO E DIVULGAÇÃO TOTAL OU PARCIAL DESTE TRABALHO, POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO, PARA FINS DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

Ficha catalográfica preparada pela Seção de Tratamento
da Informação do Serviço de Biblioteca – EESC/USP

C354p Castoldi, Marcelo Favoretto
Protótipo virtual da estratégia DTC aplicada a motores
de indução usando linguagem VHDL / Marcelo Favoretto
Castoldi ; orientador Manoel Luís de Aguiar. -- São
Carlos, 2006.

Dissertação (Mestrado-Programa de Pós-Graduação em
Engenharia Elétrica. Área de Concentração: Sistemas
Dinâmicos) -- Escola de Engenharia de São Carlos da
Universidade de São Paulo, 2006.

1. Motores de indução. 2. Controle DTC. 3. FPGA.
4. Linguagem VHDL. I. Título.

FOLHA DE JULGAMENTO

Candidato: Engenheiro **MARCELO FAVORETTO CASTOLDI**

Dissertação defendida e julgada em 23-11-2006 perante a Comissão Julgadora:

Prof. Dr. **MANOEL LUÍS DE AGUIAR (Orientador)**
(Escola de Engenharia de São Carlos/USP)

Prof. Dr. **AZAURI ALBANO DE OLIVEIRA JUNIOR**
(Escola de Engenharia de São Carlos/USP)

Prof. Associado **EDUARDO MARQUES**
(Instituto de Ciências Matemáticas e de Computação/USP)

Prof. Associado **GERALDO ROBERTO MARTINS DA COSTA**
Vice-Coordenador em Exercício do Programa de Pós-Graduação em
em Engenharia Elétrica

Prof^a. Titular **MARIA DO CARMO CALIJURI**
Presidente da Comissão de Pós-Graduação da EESC

Dedico esta obra à minha mãe
pelo amor e apoio sempre
presentes.

AGRADECIMENTOS

Agradeço primeiramente a Deus por ter me concedido saúde e principalmente força nos momentos mais difíceis desta etapa da minha vida.

Agradeço muito ao professor Dr. Manoel Luís de Aguiar pela extraordinária orientação, sempre pronto a buscar soluções nos momentos mais difíceis deste trabalho. Agradeço também à sua grande amizade durante todo este processo, sempre mostrando o melhor caminho a ser seguido.

À minha família que manteve apoio irrestrito tanto na parte afetiva quanto, em alguns momentos, na parte financeira.

À minha namorada principalmente pela paciência nos momentos de descontrole, sempre pronta a me acalmar e me confortar. Agradeço a todos da sua família também pelo suporte incondicional.

A todos os amigos e professores do LACEP que sempre se prontificaram a ajudar nos instantes de necessidade.

A todos meus amigos, principalmente ao Rodrigo, que me ajudou muito na resolução de problemas técnicos ocorridos durante este trabalho.

Enfim, agradeço a todos que de uma forma ou outra contribuíram para a conclusão deste trabalho.

RESUMO

CASTOLDI, M. F. **Protótipo Virtual da Estratégia DTC Aplicada a Motores de Indução Usando Linguagem VHDL**. 2006. 110p. Dissertação (Mestrado) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2006.

Este trabalho desenvolve a simulação do controle de velocidade de motor de indução com a técnica de Controle Direto de Torque (*Direct Torque Control* - DTC), sendo executada em um dispositivo lógico programável tipo FPGA (*Field Programmable Gate Array*). A simulação é realizada usando-se dois programas: O MATLAB/Simulink e o ModelSim, sendo que estes dois programas trabalham em modo de co-simulação provida pelo *toolbox* Link-for-ModelSim do Simulink. Enquanto a dinâmica do motor e do inversor é executada no MATLAB, o algoritmo de controle da estratégia DTC é executada no ModelSim. O algoritmo de acionamento DTC é escrito em linguagem de descrição de *hardware* VHDL (*Very High Speed Integrated Circuit Hardware Description Language* – VHSIC HDL) e utiliza a aritmética de ponto flutuante. Os resultados das simulações são apresentados e analisados no final deste trabalho.

Palavras – chave: Motor de Indução; Controle DTC; FPGA; Linguagem VHDL.

ABSTRACT

CASTOLDI, M. F. **Virtual Prototype of the DTC Strategy Applied to Induction Motors Using VHDL Code**. 2006. 110p. M.Sc. Dissertation – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2006.

This work presents a simulation of induction motor speed control using the technique of direct torque control (DTC), performed in a reprogrammable device type FPGA. The simulation is performed using two programs: MATLAB/Simulink and ModelSim, where these two programs work in a co-simulation mode, provide by Link for ModelSim *toolbox* from Simulink. While the motor and inverter dynamics is performed in MATLAB, the control algorithm of DTC technique runs in the ModelSim program. The algorithm of DTC drive is written in hardware description language (VHDL) and use the float point arithmetic. The simulation results are presented and analyzed in the end of this work.

Keywords: Induction Motor; DTC; FPGA; VHDL Code.

LISTA DE FIGURAS

Figura 2.1 - Representação dos sistemas de coordenadas trifásicas e ortogonal.....	26
Figura 2.2 - Plano complexo e referências arbitrárias.	28
Figura 2.3 - Inversor de tensão.	30
Figura 2.4 - Esquema simplificado do inversor de tensão.....	31
Figura 2.5 - Representação dos vetores espaciais de tensão gerados pelo inversor.	32
Figura 2.6 - Vetores dos fluxos de estator e de rotor no plano $\alpha\beta$	33
Figura 2.7 - Variação do fluxo de estator.	33
Figura 2.8 - Alteração do fluxo concatenado produzido por um vetor espaço de tensão.....	34
Figura 2.9 - Vetores espaço de tensão divididos em setores.	36
Figura 2.10 - Esquema do DTC.....	37
Figura 2.11 - Algoritmo de obtenção do setor do fluxo de estator (comparação).....	39
Figura 3.1 - Veículo elétrico de alta velocidade.....	51
Figura 4.1 - Arquitetura interna de uma FPGA.....	57
Figura 4.2 - Sistema reconfigurável híbrido.....	58
Figura 5.1 - Co-simulação Simulink - ModelSim.	64
Figura 5.2 - Bloco utilizado para co-simulação MATLAB - ModelSim.....	64
Figura 5.3 - Ambiente de simulação dos blocos do DTC.....	65
Figura 5.4 - Número de precisão simples <i>IEEE 754 (single format)</i>	67
Figura 5.5 - Simulação da estratégia DTC em malha aberta.	68
Figura 5.6 - Simulação da estratégia DTC com controle de velocidade.	69
Figura 6.1 - Fluxo de estator estimado pelo programa VHDL (Caso 1).	71
Figura 6.2 – Fluxo de estator medido no motor (Caso 1).....	72

Figura 6.3 - Torque eletromagnético estimado pelo programa VHDL (Caso 1).	72
Figura 6.4 - Torque eletromagnético medido no motor (Caso 1).	73
Figura 6.5 - Componente real da corrente de estator (Caso 1).	74
Figura 6.6 - Componente imaginária da corrente de estator (Caso 1).	74
Figura 6.7 - Componente real do fluxo de estator (Caso 1).	75
Figura 6.8 - Componente imaginária do fluxo de estator (Caso 1).	75
Figura 6.9 - Correntes trifásicas do estator (Caso 1).	76
Figura 6.10 - Velocidade do motor (Caso 1).	76
Figura 6.11 - Velocidade do motor (Caso 2, Simulação 1).	78
Figura 6.12 - Torque eletromagnético estimado pelo programa VHDL (Caso 2, Simulação 1).	78
Figura 6.13 - Torque eletromagnético medido no motor (Caso 2, Simulação 1).	79
Figura 6.14 - Fluxo de estator estimado pelo programa VHDL (Caso 2, Simulação 1).	80
Figura 6.15 - Fluxo de estator medido no motor (Caso 2, Simulação 1).	80
Figura 6.16 - Componente real da corrente de estator (Caso 2, Simulação 1).	81
Figura 6.17 - Componente imaginária da corrente de estator (Caso 2, Simulação 1).	81
Figura 6.18 - Componente real do fluxo de estator (Caso 2, Simulação 1).	82
Figura 6.19 - Componente imaginária do fluxo de estator (Caso 2, Simulação 1).	82
Figura 6.20 - Correntes trifásicas do estator (Caso 2, Simulação 1).	83
Figura 6.21 - Velocidade do motor (Caso 2, Simulação 2).	84
Figura 6.22 - Torque eletromagnético estimado pelo programa VHDL (Caso 2, Simulação 2).	85
Figura 6.23 - Torque eletromagnético medido no motor (Caso 2, Simulação 2).	85
Figura 6.24 - Fluxo de estator estimado pelo programa VHDL (Caso 2, Simulação 2).	86
Figura 6.25 - Fluxo de estator medido no motor (Caso 2, Simulação 2).	86

Figura 6.26 - Componente real da corrente de estator (Caso 2, Simulação 2).....	87
Figura 6.27 - Componente imaginária da corrente de estator (Caso 2, Simulação 2).....	87
Figura 6.28 - Componente real do fluxo de estator (Caso 2, Simulação 2).....	88
Figura 6.29 - Componente imaginária do fluxo de estator (Caso 2, Simulação 2).	88
Figura 6.30 - Correntes trifásicas do estator (Caso 2, Simulação 2).	89
Figura A.1 - Representação para números em ponto flutuante.	100
Figura A.2 - Comparação entre arredondamento e truncamento.....	101
Figura A.3 - Formato de ponto flutuante.....	103
Figura A.4 - Tipos de notação em ponto flutuante definidos pelo padrão <i>IEEE-754</i>	105
Figura B.1 - Conexão Simulink/ModelSim.....	107
Figura B.2 - Configuração da janela <i>Ports</i>	109
Figura B.3 - Configuração da janela <i>Clocks</i>	110
Figura B.4 - Configuração da janela <i>Timescales</i>	110
Figura B.5 - Configuração da janela <i>Connection</i>	111

LISTA DE TABELAS

Tabela 2.1- Definição dos setores espaciais.....	38
Tabela 2.2 - Vetor de tensão a ser implementado pelo inversor.	40
Tabela 5.1 - Parâmetros do motor simulado.	66
Tabela A.1 - Padrão de Ponto Flutuante <i>IEEE-754</i>	104

LISTA DE SIGLAS

DTC	<i>Direct Torque Control</i>
FPGA	<i>Field Programmable Gate Array</i>
VHSIC	<i>Very High Speed Integrated Circuit</i>
VHDL	<i>VHSIC Hardware Description Language</i>
ASIC	<i>Application-Specific Integrated Circuit</i>
CC	Corrente Contínua
CA	Corrente Alternada
MIT	Motor de Indução Trifásico
IGBT	<i>Insulated Gate Bipolar Transistor</i>
FOC	<i>Field Oriented Control</i>
DSC	<i>Direct Self Control</i>
DC	<i>Direct Current</i>
SVM	<i>Space Vector Modulation</i>
EKF	<i>Extended Kalman Filter</i>
DSVM	<i>Discrete Space Vector Modulation</i>
EMF	<i>Electro-Magnetic Field</i>
DSP	<i>Digital Signal Processor</i>
NPC	<i>Neutral Point Clamped</i>
AMS	<i>Analog Mixed Simulation</i>
A/D	<i>Analog to Digital</i>
ABB	Asea Brown Boveri
IGCT	<i>Insulated Gate Commutated Thyristor</i>
CLB	<i>Configurable Logic Block</i>
IOB	<i>Input Output Block</i>
LUT	<i>Lookup Table</i>
GPP	<i>General Purpose Processor</i>
ULA	Unidade Lógica Aritmética
IEEE	<i>Institute of Electrical and Electronics Engineers</i>
MOSFET	<i>Metal-Oxide-Semiconductor Field Effect Transistors</i>
FPU	<i>Float Point Unit</i>
ANSI	<i>American National Standards Institute</i>

EDA	<i>Electronic Design Automation</i>
TCP/IP	<i>Transmission Control Protocol/ Internet Protocol</i>

LISTA DE SÍMBOLOS

v_a, v_b, v_c	tensão das fases a, b e c respectivamente
$\vec{v}$	vetor espacial de tensão
v_α	componente real do vetor tensão
v_β	componente imaginária do vetor tensão
α	operador de deslocamento espacial (120°)
α^2	operador de deslocamento espacial (240°)
i_a, i_b, i_c	corrente das fases a, b e c respectivamente
$\vec{i}$	vetor espacial de corrente
i_α	componente real do vetor corrente
i_β	componente imaginária do vetor corrente
ψ_a, ψ_b, ψ_c	fluxo das fases a, b e c respectivamente
$\vec{\psi}$	vetor espacial de fluxo
ψ_α	componente real do vetor fluxo
ψ_β	componente imaginária do vetor fluxo
α'	ângulo entre o referencial de estator e o vetor tensão de estator
β'	ângulo entre o referencial de rotor e o vetor tensão de rotor
λ	ângulo entre o referencial de estator e o referencial genérico
θ	ângulo entre os referenciais de estator e rotor
ω_s	velocidade síncrona
ω_{mec}	Velocidade mecânica do rotor
ω_λ	velocidade angular do referencial genérico
λ_0	posição inicial do referencial genérico
$\vec{v}_{s\alpha\beta}$	vetor espacial da tensão de estator ($\vec{v}_s$)
R_s	resistência do enrolamento de uma fase de estator
$\vec{i}_{s\alpha\beta}$	vetor espacial da corrente de estator ($\vec{i}_s$)
$\vec{\psi}_{s\alpha\beta}$	vetor espacial do fluxo concatenado do estator ($\vec{\psi}_s$)

$\vec{v}_{r\alpha\beta}$	vetor espacial da tensão de rotor ($\vec{v}_r$)
R_r	resistência de uma fase do rotor (enrolamento equivalente) referida ao estator
$\vec{i}_{r\alpha\beta}$	vetor espacial da corrente de rotor ($\vec{i}_r$)
$\vec{\psi}_{r\alpha\beta}$	vetor espacial do fluxo concatenado do rotor ($\vec{\psi}_r$)
z_p	número de pares de pólos do motor
L_s	indutância própria do estator
L_H	indutância mútua entre o estator e o rotor
L_r	indutância própria do rotor
m_d	torque eletromagnético
E_{DC}	tensão do barramento CC do inversor
$Ch1-Ch6$	chaves do inversor
Ch_a, Ch_b, Ch_c	chaves das fases a, b e c do inversor, respectivamente
$\vec{v}_i$	vetor espacial de tensão gerado pela combinação das chaves do inversor
$\vec{v}_0, \vec{v}_7$	vetores espaciais de tensão nulos (comandados pelo inversor)
$\vec{v}_1$ a $\vec{v}_6$	vetores espaciais de tensão não nulos (comandados pelo inversor)
L'_s	indutância transiente de estator
σ	constante de dispersão resultante
γ	ângulo entre o vetor fluxo de estator e o vetor fluxo de rotor
$\delta_{\psi s}$	ângulo entre o referencial estacionário e o fluxo de estator
τ_r	constante de tempo do rotor
τ_s	constante de tempo do estator
Δt	intervalo de tempo
$\Delta\gamma$	variação do ângulo entre os vetores fluxo de estator e fluxo de rotor
$\Delta torque$	variação do torque eletromagnético
N	setor espacial no qual uma grandeza se encontra
δ	ângulo de uma grandeza referenciada ao estator
ψ_{sref}	amplitude do fluxo de referência do estator
m_{dref}	torque de referência

$\Delta\psi_s$	largura da banda de histerese do fluxo do estator
Δm_d	largura da banda de histerese de torque eletromagnético
Γ	saída da histerese de torque do controle DTC
Φ	saída da histerese de fluxo do controle DTC
ξ	amplitude de erro
m_w	tamanho da mantissa (bits) de um número ponto flutuante
e_w	tamanho do expoente (bits) de um número ponto flutuante
f	número em ponto flutuante
S	bit do sinal de um número em ponto flutuante
M	mantissa fracionária de um número em ponto flutuante
E	expoente de um número em ponto flutuante
E_{bias}	excesso do expoente de um número em ponto flutuante

SUMÁRIO

Capítulo 1 - Introdução	20
Capítulo 2 - Motor de Indução e o Acionamento com Controle de Torque Direto – DTC	23
2.1 - Introdução	23
2.2 - O Motor de Indução Trifásico	24
2.3 - Acionamento com Inversor de Tensão	30
2.4 - Fundamentos e Relações Gerais do DTC	32
2.5 - Configurações do DTC	36
2.5.1 - Obtenção do Setor Espacial do Fluxo de Estator.....	37
2.5.2 - Controladores de Histerese de Torque e de Fluxo.....	39
2.5.3 - Obtenção dos Vetores de Tensão.....	40
2.5.4 - Características Gerais, Vantagens e Desvantagens da Técnica DTC	40
Capítulo 3 - Estado da Arte: o Uso e Realização do DTC no Acionamento de Motores de Indução.....	42
3.1 - Introdução	42
3.2 - Propostas Alternativas na metodologia DTC	43
3.3 - Alternativas de Implementação da Técnica DTC.....	46
3.4 - Desenvolvimentos Comerciais e Aplicações Industriais.....	49
Capítulo 4 - Hardware Reconfigurável	53
4.1 - Introdução	53
4.2 - Arquiteturas Reconfiguráveis	55
4.3 - Dispositivos Reconfiguráveis	56
4.4 - Programação de Dispositivos Reconfiguráveis	58
Capítulo 5 - Elaboração da Plataforma Virtual de Ensaios DTC.....	62
5.1 - Introdução	62
5.2 - Parâmetros da Plataforma Virtual.....	63
5.3 - Simulações.....	67
5.3.1 - Caso 1: Malha Aberta de Velocidade	67

5.3.2 - Caso 2: Malha Fechada de Velocidade	68
Capítulo 6 - Resultados das Simulações.....	70
6.1 - Introdução	70
6.2 - Resultado das Simulações	71
6.2.1 - Caso 1: Malha Aberta de Velocidade.....	71
6.2.2 - Caso 2: Malha Fechada de Velocidade	77
Capítulo 7 - Conclusões e Sugestões para Trabalhos Futuros.....	90
7.1 - Conclusões	90
7.2 - Sugestões para Trabalhos Futuros.....	92
Capítulo 8 - Referências Bibliográficas.....	93
Apêndice A - Aritmética de Ponto Flutuante	99
A.1 - Introdução.....	99
A.2 - Análise da Aritmética de Ponto Flutuante.....	99
A.3 - Padrão IEEE-754.....	102
Apêndice B - Co-Simulação Simulink/ModelSim	106
B.1 - Conexão Simulink/Modelsim.....	107
B.2 - Modos de Comunicação	107
B.3 - Conectando Simulink e ModelSim.....	108

Capítulo 1

Introdução

Em anos recentes, as aplicações comerciais de motores de indução com campo orientado têm aumentado muito, incluindo máquinas da indústria metalúrgica, máquinas da indústria de papel, máquinas de ferramentas, e assim por diante [1]. Em paralelo, estudos estão sendo realizados para encontrar soluções para o controle de motores de indução com dois objetivos principais: i) um controle preciso e rápido do fluxo e do torque do motor como em acionamentos com orientação de campo, e ii) redução da complexidade dos algoritmos envolvidos na técnica de orientação de campo.

A rapidez de resposta às solicitações exigidas do torque eletromagnético associada com as reduzidas oscilações do fluxo de estator são as principais vantagens dos acionamentos que utilizam a técnica de controle direto de torque (*Direct Torque Control* - DTC). Além disso, a escolha otimizada dos chaveamentos com a utilização desta técnica representa redução na frequência de chaveamento do inversor e, por conseguinte, redução das perdas [2].

O presente trabalho trata-se de simulações da estratégia DTC, sendo esta escrita em linguagem de descrição de hardware (*VHSIC Hardware Description Language* –VHDL) usando aritmética de ponto flutuante. Foram utilizados dois programas para executar as simulações: MATLAB/Simulink e o ModelSim. Eles trabalharam em modo de co-simulação, sendo que a parte de potência do sistema foi simulada no MATLAB/Simulink, e o código com a estratégia de controle DTC foi simulado no ambiente do ModelSim.

Foi realizada uma simulação para verificar a funcionabilidade do código e posteriormente mais duas simulações foram executadas com o intuito de simular um controle de velocidade de um motor de indução trifásico, usando a técnica DTC. Os resultados foram analisados e discutidos na dissertação.

O trabalho mostra uma nova forma de simulação de processos dinâmicos que contenham partes de algoritmos que são escritos em linguagem de descrição de hardware. Geralmente tais códigos são implementados em dispositivos lógicos programáveis, como por exemplo, ASICs (*Application-Specific Integrated Circuit*) e principalmente FPGAs. Tal metodologia permite verificar o comportamento destes códigos antes de suas implementações, reduzindo riscos de mudanças significativas quando implementados.

O trabalho é organizado como descrito nos parágrafos seguintes:

No Capítulo 2, primeiramente, são apresentadas as equações dos modelos do motor de indução empregados neste trabalho. Posteriormente, foi feita uma análise sobre inversores de tensão, tendo em vista principalmente seus estados de chaveamento. Na sequência, fundamentos teóricos básicos sobre o DTC foram apresentados e analisados. A configuração do DTC, assim como cada um de seus componentes, foram analisados no término deste capítulo.

O Capítulo 3, trata de uma revisão bibliográfica de muitos trabalhos realizados que utilizam a técnica de controle direto de torque. Primeiramente foram discutidas metodologias e formas de implementação desta técnica de controle. Uma pesquisa extensa sobre os módulos comerciais que trabalham realizando a técnica DTC é exibida a seguir.

O Capítulo 4 foi dedicado exclusivamente para o tema da computação reconfigurável. Primeiramente, foram introduzidos os conceitos de arquiteturas e dispositivos reconfiguráveis e, no término do capítulo, a programação de dispositivos reconfiguráveis foi analisada.

O Capítulo 5 apresenta como foi projetada a plataforma virtual de ensaios DTC proposta no trabalho. Primeiramente, foram apresentados os parâmetros construtivos de tal plataforma, explicando passo a passo todas as configurações da mesma. Posteriormente, foram descritas as simulações que validam a plataforma.

No Capítulo 6 são apresentados e analisados os resultados das simulações descritas no capítulo anterior.

O Capítulo 7 exhibe as conclusões gerais sobre o trabalho além de sugestões para futuros trabalhos.

No Apêndice A, o conceito de computação em ponto flutuante foi abordada, introduzindo conceitos e exemplos. A norma internacional de padronização de ponto flutuante também foi analisada nesta parte do trabalho.

O Apêndice B apresenta, de maneira resumida, como conectar os programas em questão para fazer a co-simulação.

Capítulo 2

Motor de Indução e o Acionamento com Controle de Torque Direto – DTC

2.1 Introdução

O estudo da técnica DTC aplicada a um motor de indução, analisando suas características, estrutura, vantagens e desvantagens é feito depois de se conhecer os modelos matemáticos envolvidos nesse sistema. Dessa forma são apresentados a seguir os modelos matemáticos empregados nesse trabalho, bem como as transformações de coordenadas pertinentes. Os modelos aqui apresentados não serão desenvolvidos passo a passo por serem muito conhecidos e facilmente encontrado em publicações da área de acionamentos de motores como, por exemplo, em [3], [4] e [5]. Porém, as simplificações envolvidas nestes modelos serão desenvolvidas e analisadas. Serão apresentados os modelos reduzidos empregados na estimação do fluxo do estator e do torque eletromagnético, realizados durante a execução do algoritmo de controle DTC proposto no trabalho.

O modelo do inversor de tensão também será apresentado neste capítulo, assim como as características e configurações da técnica DTC. No item em que são tratadas as principais características do DTC, são expostas equações e figuras que facilitam a compreensão da técnica. No tópico em que é tratada a configuração da técnica, cada bloco componente do

DTC é detalhadamente explicado e analisado, sendo tiradas conclusões sobre as principais vantagens e desvantagens deste acionamento no final do capítulo.

2.2 O Motor de Indução Trifásico

O motor de indução é o tipo de motor elétrico mais utilizado e difundido nas mais variadas aplicações em todo mundo. Em princípio, este fato se deve por adotar um sistema de distribuição de energia elétrica em corrente alternada. Com o tempo, o motor de indução provou ser ainda um tipo de motor muito robusto em termos de operação e construção e proporcionando consideráveis vantagens econômicas no tocante à aquisição e manutenção. Estas vantagens foram ainda reforçadas com a introdução dos motores de indução com rotor em gaiola, possibilitando a sua construção com custo ainda mais baixo [6].

Apesar de todas estas características, os motores de indução não eram utilizados em aplicações com velocidade variável devido à necessidade de estruturas muito complexas e desempenho inferior aos servosistemas CC.

Entretanto, o interesse sempre foi mantido, de tal forma que muitos esforços foram dedicados no sentido do desenvolvimento dos servosistemas CA. Os resultados iniciais surgiram em termos de novas técnicas de modelagem com o propósito de se obter um conjunto de equações dinâmicas mais simples e voltadas para aplicações de controle [6]. Porém, a necessidade de um grande esforço computacional além de conversores de potência ainda inexistentes ou de desempenho insatisfatório, eram barreiras para sua implementação.

O surgimento de microprocessadores de alto desempenho e baixo custo além do grande avanço tecnológico na produção de componentes semicondutores (eletrônica de potência) foram determinantes para o uso de servosistemas CA.

Atualmente, visto que o desenvolvimento e implantação de sistemas de automação já são normalmente desenvolvidos com base em sistemas digitais, o custo global do acionamento e controle de servosistemas CC e CA se tornaram concorrentes. Do ponto de

vista de desempenho, com auxílio de técnicas de modelagem e acionamento especiais, os servosistemas CA também se igualam aos servosistemas CC. Visto que o custo dos motores de indução é bem inferior, os servosistemas CA passam a ser mais interessantes atualmente [7].

Os motores de indução trifásicos convencionais podem conter dois enrolamentos trifásicos, sendo um na estrutura fixa denominada estator e outro na estrutura girante denominada rotor. Uma forma muito comum é quando o enrolamento de rotor é substituído por sistema de barras paralelas e ligeiramente inclinadas e curto-circuitadas em seus extremos formando um tipo de gaiola de esquilo e denominado rotor em gaiola. Há também os motores com rotor bobinado que dispõem as extremidades do enrolamento trifásico possibilitando uma atuação ou mesmo medições de grandezas elétricas do rotor. Os motores com rotor bobinado possuem custos mais elevados.

Considerando-se as formas construtivas descritas, é possível se estabelecer os elementos fundamentais para definição do circuito elétrico equivalente do motor. Em primeiro lugar, o enrolamento trifásico de estator e eventualmente de rotor, é constituído por 3 bobinas de várias espiras, às quais são distribuídas espacialmente na estrutura circular do motor. A cada uma destas bobinas é atribuído um eixo e que se encontram defasados de 120° elétricos entre si. Desta forma, tanto na estrutura de rotor como de estator, definem-se 3 indutâncias e 3 resistências referentes ao enrolamento trifásico, sendo cada uma delas definidas como grandezas por fase. Devido à critérios de construção simétrica do motor, as indutâncias são admitidas iguais entre si, assim como as resistências.

Tensões alternadas trifásicas alimentam o motor por meio do estator. As correntes de estator no enrolamento trifásico geram o campo girante que induz correntes no rotor, e a interação das correntes do estator e do rotor produzem o torque que faz girar o motor.

Uma vez que o Motor de Indução Trifásico – MIT, é constituído por três enrolamentos independentes defasados espacialmente na estrutura de estator e excitados por tensões/correntes eletricamente defasadas no tempo, uma visão espacial do comportamento dinâmico e em regime e mesmo da contribuição total de cada componente de fase (estator/rotor) pode resultar extremamente complexa.

Com intuito de se obter uma simplificação de modelo do MIT e uma redução do número de variáveis das expressões matemáticas que descrevem o comportamento dinâmico do mesmo, projeta-se o sistema de coordenadas trifásicas em um plano coordenado ortogonal [8].

Nesta metodologia, substitui-se o sistema trifásico (de 3 eixos com defasagem de 120° entre si) por um sistema ortogonal. O MIT será visto então como sendo constituído por apenas dois enrolamentos defasados espacialmente de 90° no estator e rotor. A Figura 2.1 representa a disposição dos sistemas trifásico e o ortogonal. Com base na Figura 2.1, o novo sistema de eixos recebe denominação $\alpha\beta$ e consequentemente índices α e β .

Figura 2.1 - Representação dos sistemas de coordenadas trifásicas e ortogonal.

A modelagem dinâmica de máquinas, parte do princípio de que as grandezas das máquinas elétricas são consideradas periódicas, as quais podem ser resolvidas como séries de

harmônicas. Assim, as ondas espaciais são representadas por fasores espaciais. O modelo ortogonal do MIT resulta em consideráveis simplificações, porém uma visualização espacial das componentes das grandezas elétricas ainda não é garantida.

Tendo em vista a geração de conjugado magnético como análise do MIT, o qual é produzido por uma relação de fluxos e correntes do motor, é desejável que cada grandeza seja representada de forma a englobar as contribuições de cada fase, fazendo-se assim necessário a utilização de fasores de espaço. O uso da notação de fasor de espaço resulta da imposição de que o sistema ortogonal seja um plano complexo e todas as grandezas representadas neste plano, serão descritas por uma parte real e uma parte imaginária. Desta forma, impõe-se que todas as grandezas elétricas sejam representadas como entidades complexas [9].

Assim, as tensões, correntes e fluxos do motor podem ser descritos no modelo ortogonal complexo ($\alpha\beta$) a partir de seu modelo trifásico, como visto a seguir:

$$\vec{v} = v_\alpha + jv_\beta = \frac{2}{3} \begin{bmatrix} 1 & \alpha & \alpha^2 \end{bmatrix} \begin{bmatrix} v_a \\ v_b \\ v_c \end{bmatrix} \quad (2.1)$$

$$\vec{i} = i_\alpha + ji_\beta = \frac{2}{3} \begin{bmatrix} 1 & \alpha & \alpha^2 \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} \quad (2.2)$$

$$\vec{\psi} = \psi_\alpha + j\psi_\beta = \frac{2}{3} \begin{bmatrix} 1 & \alpha & \alpha^2 \end{bmatrix} \begin{bmatrix} \psi_a \\ \psi_b \\ \psi_c \end{bmatrix} \quad (2.3)$$

com

$$\alpha = e^{j120^\circ} = \cos(120^\circ) + j\sin(120^\circ) = -\frac{1}{2} + j\frac{\sqrt{3}}{2} \quad (2.4)$$

Nas definições 2.1 a 2.3, assume-se que a fase “a” do sistema trifásico coincide com o eixo real do plano ou sistema complexo. Os termos α e α^2 indicam a direção das componentes nas fases “b” e “c” respectivamente, ou seja, são operadores de deslocamento espacial de

modo que α desloca ou opera um deslocamento espacial de 120° e α^2 opera um deslocamento espacial de 240° .

Geralmente, grandezas envolvidas nas expressões fundamentais de tensão do motor estão parte relacionadas ao referencial de estator e parte ao referencial de rotor. Para se manipular tais equações, é necessário se obter a representação em um referencial único e comum para as grandezas de estator e rotor. Além disso, a representação com referencial único, é necessária para que se possa obter estratégias de controle adequadas [6].

A Figura 2.2 mostra o plano complexo com possíveis referenciais únicos que podem ser adotados. O vetor de espaço de tensões de estator tem um deslocamento angular α' com relação ao referencial de estator, enquanto que o vetor de espaço de tensões de rotor um deslocamento β' com relação ao referencial de rotor [5].

Figura 2.2 - Plano complexo e referências arbitrárias.

Considerando-se o referencial genérico λ - com deslocamento angular λ com relação ao estator - o vetor de tensões de estator tem um deslocamento angular $(\alpha' - \lambda)$ com relação ao referencial λ . No caso do vetor de tensões de rotor, o deslocamento angular é $(\beta' - \lambda + \theta)$. O referencial genérico pode ser assumido como sendo qualquer daqueles definidos anteriormente. Da mesma forma a velocidade angular do referencial λ será dada por:

$$\omega_\lambda = \frac{d}{dt} \lambda \quad (2.5)$$

com

$$\lambda = \omega_\lambda t + \lambda_0 \quad (2.6)$$

As equações do motor no referencial genérico λ , podem ser vistas a seguir:

$$\vec{v}_{s\alpha\beta} = R_s \vec{i}_{s\alpha\beta} + \frac{d}{dt} \vec{\psi}_{s\alpha\beta} + j \omega_\lambda \vec{\psi}_{s\alpha\beta} \quad (2.7)$$

$$\vec{v}_{r\alpha\beta} = R_r \vec{i}_{r\alpha\beta} + \frac{d}{dt} \vec{\psi}_{r\alpha\beta} + j(\omega_\lambda - z_p \omega_{mec}) \vec{\psi}_{r\alpha\beta} \quad (2.8)$$

$$\vec{\psi}_{s\alpha\beta} = L_s \vec{i}_{s\alpha\beta} + L_H \vec{i}_{r\alpha\beta} \quad (2.9)$$

$$\vec{\psi}_{r\alpha\beta} = L_H \vec{i}_{s\alpha\beta} + L_r \vec{i}_{r\alpha\beta} \quad (2.10)$$

$$m_d = \frac{3}{2} z_p \operatorname{Im} \{ \vec{i}_{s\alpha\beta} \vec{\psi}_{s\alpha\beta}^* \} \quad (2.11)$$

Como a técnica de controle DTC controla grandezas do estator (correntes, tensões e fluxo), para facilitar o equacionamento do motor, adota-se o referencial estacionário ($\omega_\lambda = 0$) [4] [5]. Assim, as equações que regem o motor neste referencial podem ser vistas na seqüência:

$$\vec{v}_{s\alpha\beta} = R_s \vec{i}_{s\alpha\beta} + \frac{d}{dt} \vec{\psi}_{s\alpha\beta} \quad (2.12)$$

$$\vec{v}_{r\alpha\beta} = R_r \vec{i}_{r\alpha\beta} + \frac{d}{dt} \vec{\psi}_{r\alpha\beta} - j z_p \omega_{mec} \vec{\psi}_{r\alpha\beta} \quad (2.13)$$

$$\vec{\psi}_{s\alpha\beta} = L_s \vec{i}_{s\alpha\beta} + L_H \vec{i}_{r\alpha\beta} \quad (2.14)$$

$$\vec{\psi}_{r\alpha\beta} = L_H \vec{i}_{s\alpha\beta} + L_r \vec{i}_{r\alpha\beta} \quad (2.15)$$

$$m_d = \frac{3}{2} z_p (\psi_{s\alpha} i_{s\beta} - \psi_{s\beta} i_{s\alpha}) \quad (2.16)$$

Estas equações são fundamentais para o DTC, que parte do princípio de controle de fluxo de estator e torque eletromagnético do motor. No decorrer do trabalho, as equações do motor serão citadas e analisadas oportunamente.

2.3 Acionamento com Inversor de Tensão

A técnica DTC se resume numa escolha adequada e otimizada dos estados de chaveamento do inversor de tensão que alimenta o MIT. Desta forma o estudo das formas de acionamento dos inversores de tensão é um item importante para este estudo.

Os inversores tipicamente empregados nesta técnica de controle são implementados com IGBT (*Insulated Gate Bipolar Transistor*) e diodos, como mostra a Figura 2.3. Isto se deve basicamente à faixa de frequência de chaveamento suportada por estes componentes.

Figura 2.3 - Inversor de tensão.

O nível de tensão E_{DC} é obtido pela retificação da tensão da rede. Os transistores (*Ch1-Ch6*) são comandados por um bloco de controle, que no caso é justamente o procedimento designado DTC, fazendo com que o inversor gere a tensão requerida em cada instante de chaveamento [10]. Os transistores de um mesmo braço, por exemplo *Ch1* e *Ch4*

são acionados alternadamente de forma a evitar um curto-circuito e podem ser representados como chaves ideais. Assim, pode-se simplificar a representação do inversor tal como mostra a Figura 2.4.

Figura 2.4 - Esquema simplificado do inversor de tensão.

A combinação de três das chaves com dois estados (aberta e fechada) permite a obtenção de oito combinações gerando oito vetores de tensão. Efetuando-se alguns cálculos verifica-se facilmente o vetor espacial de tensão para cada combinação das chaves, de acordo com a equação (2.17) a seguir:

$$\vec{v}_i = \frac{2}{3} (Ch_a + Ch_b \alpha + Ch_c \alpha^2) E_{DC} \quad (2.17)$$

onde Ch_a , Ch_b e Ch_c possuem valores “1” (chave fechada) ou “0” (chave aberta) dependendo do acionamento ou não das chaves.

A representação espacial dos oito vetores de espaço segundo a equação (2.17) pode ser observada na Figura 2.5. Os vetores nulos $\vec{v}_0$ e $\vec{v}_7$ correspondem à situações em que os transistores Ch_1 , Ch_3 e Ch_5 (Figura 2.3) estão simultaneamente cortados ($\vec{v}_0$) ou conduzindo ($\vec{v}_7$). Os vetores espaciais $\vec{v}_1$ a $\vec{v}_6$ possuem a mesma amplitude, dada por dois terços da tensão do barramento CC do inversor.

Figura 2.5 - Representação dos vetores espaciais de tensão gerados pelo inversor.

2.4 Fundamentos e Relações Gerais do DTC

Em uma máquina de indução trifásica, a produção de torque eletromagnético pode ser expresso através da equação (2.18), na qual é proporcional ao produto vetorial do vetor fluxo do estator $\vec{\psi}_{s\alpha\beta}$ pelo vetor do fluxo de rotor referido ao estator $\vec{\psi}_{r\alpha\beta}$ [2].

$$m_d = -\frac{3}{2} z_p \frac{L_H}{L'_s L_r} (\vec{\psi}_{s\alpha\beta} \times \vec{\psi}_{r\alpha\beta}) \quad (2.18)$$

onde, $L'_s = \sigma L_s$ e $\sigma = 1 - \frac{L_H^2}{L_s L_r}$

Tendo em vista a Figura 2.6, a equação (2.18) pode ser desenvolvida como segue:

$$m_d = \frac{3}{2} z_p \frac{L_H}{L'_s L_r} |\vec{\psi}_{r\alpha\beta}| |\vec{\psi}_{s\alpha\beta}| \sin(\gamma) \quad (2.19)$$

na qual, γ é considerado o ângulo entre os vetores dos fluxos do estator e do rotor.

A constante de tempo elétrica do rotor de um motor de indução gaiola de esquilo

$\left(\tau_r = \frac{L_r}{R_r} \right)$ é normalmente elevada quando comparada com a constante de tempo elétrica do

estator $\left(\tau_s = \frac{L_s}{R_s} \right)$ [4]. Ou seja, o módulo do fluxo do rotor muda muito lentamente

comparado com a mudança no módulo do fluxo do estator. Considerando constante o módulo do fluxo estator, o módulo do fluxo do rotor é praticamente constante. Assim, se os módulos dos fluxos de estator e rotor são assumidos constantes, de acordo com a equação (2.19), o torque eletromagnético pode ser modificado alterando o ângulo γ na direção requerida. A variação do ângulo γ pode ser alterada facilmente pelo chaveamento apropriado do inversor de tensão, resultando em uma alteração do vetor de tensão. Esta é a essência do controle direto de torque.

Figura 2.6 - Vetores dos fluxos de estator e de rotor no plano $\alpha\beta$.

A situação mencionada anteriormente pode ser compreendida observando-se a Figura 2.7 a seguir [5].

Figura 2.7 - Variação do fluxo de estator.

O vetor espaço de tensão $\vec{v}_1$ é aplicado por Δt segundos, acarretando uma variação angular no fluxo de estator equivalente ao $\Delta\gamma$.

Com base na Figura 2.7, poderia-se então usar um dos seis possíveis vetores de espaço de tensão relativos ao inversor de tensão e operar uma das seguintes variações do fluxo de estator, tal como ilustrado pela Figura 2.8 [5].

Figura 2.8 - Alteração do fluxo concatenado produzido por um vetor espaço de tensão.

Tendo em vista a equação de torque dada por (2.19) e a Figura 2.7, verifica-se que uma variação na tensão do estator, causa uma variação no fluxo de estator, com o fluxo do rotor permanecendo constante. Assim, segundo a Figura 2.7, produziu-se um decréscimo $\Delta torque$ de torque, já que o ângulo γ foi reduzido. Assim, diz-se que através da tensão atuou-se diretamente no fluxo e no torque instantaneamente, desde que as variações sejam suficientemente pequenas e em reduzido período de tempo.

Portanto, para aumentar o torque deve-se mover o vetor fluxo pra frente do movimento e para reduzir o torque avança-se contra ou mantém-se parado o vetor fluxo de estator em relação ao movimento. Assim, as condições básicas para o controle DTC são:

- o torque gerado é proporcional ao seno do ângulo entre os vetores fluxo de estator e rotor;
- reações do fluxo de rotor devido a mudanças no fluxo de estator são muito mais lentas (comparadas ao estator).

Desde que os vetores de espaço de tensão disponíveis pela topologia do inversor da Figura 2.5 são constantes e defasados de 60° entre si, eles determinam setores de atuação em que podem produzir efeito significativo na atuação sobre o torque. Uma atuação significativa, segundo a equação (2.19) é uma significativa contribuição do termo γ e, segundo a Figura 2.8, alguns vetores contribuem mais e outros menos. É exatamente esta seleção adequada que produzirá um chaveamento otimizado do inversor em função de que como se deseja atuar sobre o torque. Esta contribuição significativa é estabelecida pelas posições fixas de cada vetor de espaço de tensão em função de setores angulares em que eles atuam. Uma das grandes vantagens da estratégia DTC é a necessidade de saber apenas o setor angular em que o fluxo se encontra e não sua posição exata. Assim, o plano dos vetores de espaço deve ser dividido em setores que provêm contribuições significativas no torque, tal como esclarecido acima. A distribuição destes setores é tal como mostra a Figura 2.9.

A partir da Figura 2.9, é possível estabelecer o setor de uma determinada grandeza conhecendo-se o ângulo espacial desta grandeza tal como (2.20):

$$(2N - 3)\frac{\pi}{6} < \delta(N) < (2N - 1)\frac{\pi}{6} \quad (2.20)$$

sendo N o setor no qual se encontra a grandeza.

A expressão (2.20) será necessária para se obter o setor do fluxo de estator e então escolher qual será o próximo vetor de tensão a ser chaveado no motor.

Figura 2.9 - Vetores espaço de tensão divididos em setores.

2.5 Configurações do DTC

A Figura 2.10 apresenta um diagrama de blocos resumido da técnica DTC. O principal objetivo da técnica é o controle do torque e do fluxo do estator, realizado através de controladores histerese, o que assegura uma rápida resposta de torque [10].

As correntes medidas na entrada do motor, são entradas do bloco “modelo do motor”, que estima o torque e o fluxo de estator naquele instante, além do setor espacial onde o fluxo de estator se encontra. O torque e o fluxo de estator estimados são comparados às suas respectivas referências. Os erros de tais comparações são entradas dos comparadores de histerese (normalmente de dois níveis para o fluxo e de três níveis para o torque) que verificam se o torque e o fluxo estão dentro de seus limites determinados. As saídas destes comparadores, assim como o setor espacial do fluxo de estator são entradas da “Tabela de Vetores”, que determina o chaveamento adequado do inversor para manter o fluxo de estator e o torque próximos a seus valores de referência.

Figura 2.10 - Esquema do DTC.

Na instrumentação estão indicadas somente a medição das correntes de forma a se construir o vetor de corrente de estator. Como o sistema é simétrico, é possível medir a corrente de somente duas das fases, sendo a terceira encontrada facilmente através das medições das duas primeiras. As tensões podem ser medidas ou podem ser reconstruídas a partir do estado de chaveamento comandado no inversor. Esta última opção requer mais esforço computacional e deve ser um compromisso de projeto.

A referência de fluxo geralmente é escolhida com o valor do fluxo nominal do motor (calculado através dos parâmetros construtivos do motor), enquanto que a referência do torque pode ser um comando direto ou gerada através de uma malha de velocidade e/ou posição.

Cada bloco que compõe a estratégia de Controle DTC será explicado a seguir.

2.5.1 Obtenção do Setor Espacial do Fluxo de Estator

Tendo como base a equação (2.12), o fluxo de estator do motor pode ser estimado de acordo com a equação (2.21) a seguir:

$$\vec{\psi}_{s\alpha\beta} = \int (\vec{v}_{s\alpha\beta} - R_s \vec{i}_{s\alpha\beta}) dt + \vec{\psi}_{s\alpha\beta}(0) \quad (2.21)$$

O setor espacial N é obtido a partir do ângulo ($\delta_{\psi s}$) entre o referencial estacionário e o fluxo de estator, tal como na equação (2.22) e tendo em vista a equação (2.20) [2]:

$$\delta_{\psi s} = \text{tg}^{-1} \left(\frac{\psi_{s\beta}}{\psi_{s\alpha}} \right) \quad (2.22)$$

Outra maneira de obter o setor em que o fluxo se encontra é utilizar senos e cossenos dados pela equação (2.23) a seguir [10]:

$$\cos(\delta_{\psi s}) = \frac{\psi_{s\alpha}}{|\vec{\psi}_{s\alpha\beta}|} \quad (2.23a)$$

$$\text{sen}(\delta_{\psi s}) = \frac{\psi_{s\beta}}{|\vec{\psi}_{s\alpha\beta}|} \quad (2.23b)$$

A Tabela 2.1 mostra os valores dos senos e cossenos correspondentes a cada setor espacial.

Tabela 2.1- Definição dos setores espaciais utilizando senos e cossenos.

Setor	$\cos(\delta_{\psi s})$	$\text{sen}(\delta_{\psi s})$
1	$\frac{\sqrt{3}}{2} < \cos(\delta_{\psi s})$	-
2	$0 \leq \cos(\delta_{\psi s}) \leq \frac{\sqrt{3}}{2}$	$0 \leq \text{sen}(\delta_{\psi s})$
3	$-\frac{\sqrt{3}}{2} \leq \cos(\delta_{\psi s}) \leq 0$	$0 \leq \text{sen}(\delta_{\psi s})$
4	$\cos(\delta_{\psi s}) < -\frac{\sqrt{3}}{2}$	-
5	$-\frac{\sqrt{3}}{2} \leq \cos(\delta_{\psi s}) \leq 0$	$\text{sen}(\delta_{\psi s}) < 0$
6	$0 \leq \cos(\delta_{\psi s}) \leq \frac{\sqrt{3}}{2}$	$\text{sen}(\delta_{\psi s}) < 0$

Uma terceira alternativa para obter o setor espacial do vetor fluxo de estator pode ser observado no algoritmo da Figura 2.11 [2]. Este apresenta vantagens sobre os demais por evitar a divisão e operações trigonométricas inversas, sendo que na Figura,

$$\vec{\psi}_{s\beta}^* = |\vec{\psi}_{s\alpha\beta}| \sin(30^\circ).$$

Figura 2.11 - Algoritmo de obtenção do setor do fluxo de estator (comparação).

2.5.2 Controladores de Histerese de Torque e de Fluxo

Para selecionar os vetores de chaveamento do inversor deve-se avaliar a atuação dos controladores de histerese e prover as variáveis Φ e Γ . A variável de fluxo é avaliada em função de acréscimo/decrécimo de magnitude. O torque por sua vez deve ser avaliado em função de aumento e diminuição e também da manutenção de sua amplitude. Desta forma o estado das variáveis Φ e Γ são obtidas como sendo:

$$\Phi = \begin{cases} 0 & \text{se } |\vec{\psi}_{s\alpha\beta}| < (\psi_{sref} - \Delta\psi_s) \\ 1 & \text{se } |\vec{\psi}_{s\alpha\beta}| > (\psi_{sref} + \Delta\psi_s) \end{cases} \quad (2.24)$$

$$\Gamma = \begin{cases} 1 & \text{se } m_d \leq (m_{dref} - \Delta m_d) \\ 0 & \text{se } m_d < (m_{dref} + \Delta m_d) \text{ e } m_d > (m_{dref} - \Delta m_d) \\ -1 & \text{se } m_d \geq (m_{dref} + \Delta m_d) \end{cases} \quad (2.25)$$

2.5.3 Obtenção dos Vetores de Tensão

As formulações dos vetores de tensão em função das variáveis Φ e Γ seguem o princípio ilustrado na Figura 2.8. Estas formulações podem resultar em equações de verificação e ou simples tabelas de consulta. Como forma de minimizar o esforço computacional, a geração de tabelas pode ser mais eficiente. A Tabela 2.2 a seguir mostra o vetor de tensão a ser aplicado no inversor tendo em vista o estado das saídas dos controladores de histerese e o setor espacial do fluxo

Tabela 2.2 - Vetor de tensão a ser implementado pelo inversor.

Φ	Γ	Setor 1	Setor 2	Setor 3	Setor 4	Setor 5	Setor 6
0	1	$\vec{v}_2$	$\vec{v}_3$	$\vec{v}_4$	$\vec{v}_5$	$\vec{v}_6$	$\vec{v}_1$
	0	$\vec{v}_7$	$\vec{v}_0$	$\vec{v}_7$	$\vec{v}_0$	$\vec{v}_7$	$\vec{v}_0$
	-1	$\vec{v}_6$	$\vec{v}_1$	$\vec{v}_2$	$\vec{v}_3$	$\vec{v}_4$	$\vec{v}_5$
1	1	$\vec{v}_3$	$\vec{v}_4$	$\vec{v}_5$	$\vec{v}_6$	$\vec{v}_1$	$\vec{v}_2$
	0	$\vec{v}_0$	$\vec{v}_7$	$\vec{v}_0$	$\vec{v}_7$	$\vec{v}_0$	$\vec{v}_7$
	-1	$\vec{v}_5$	$\vec{v}_6$	$\vec{v}_1$	$\vec{v}_2$	$\vec{v}_3$	$\vec{v}_4$

2.5.4 Características Gerais, Vantagens e Desvantagens da Técnica DTC

A seguir é feito um resumo das principais características do controle direto de torque, assim como das suas vantagens e desvantagens em acionamentos de motores.

Características Gerais

- Atuação direta e independente sobre o torque e fluxo com seleção otimizada de chaveamento;

- Atuação indireta sobre tensão e corrente de estator;
- Fluxos e correntes resultarão quase senoidais;
- Alto desempenho dinâmico;
- Frequência de chaveamento no inversor é dependente das bandas de histerese nos controladores.

Principais Vantagens

- Ausência de módulos de geração de modulação;
- Ausência de circuitos e estratégias de desacoplamento;
- Necessita poucos controladores;
- Necessita apenas do setor e não da posição do fluxo;
- Mínimo tempo de resposta.

Principais Desvantagens

- Pode apresentar problemas de partida e operação em baixa velocidade;
- Requer estimação de fluxo (tal como no controle de campo orientado - *Field Oriented Control* - FOC);
- Frequência de chaveamento no inversor é variável;
- Pode apresentar elevados *ripples* de torque.

Capítulo 3

Estado da Arte: o Uso e Realização do DTC no Acionamento de Motores de Indução

3.1 Introdução

As técnicas de controle de torque direto começaram a ser introduzidas no início da década de 80, a partir dos trabalhos de Takahashi [11] e Depenbrock [12] aplicando princípios de orientação de campo para o estator e modulação espacial [13]. A técnica introduzida por Takahashi [11] em 1985, controle direto de torque (DTC), é a mais conhecida desse conjunto de técnicas.

O DTC baseia-se na implementação de uma tabela lógica que calcula o chaveamento a ser aplicado pelo inversor de tensão em função dos parâmetros erro de torque, erro de fluxo do estator e setor espacial onde o fluxo do estator se encontra. A tabela lógica é aplicada de forma a realizar o controle direto do torque e do fluxo do estator. A cada instante de amostragem, um novo vetor de chaveamento é escolhido.

Uma rápida resposta de torque, além de uma excelente regulação da velocidade em malha fechada são obtidos com a técnica DTC. Trabalhos recentes apresentam a possibilidade de aplicação dessa técnica em sistemas de transporte, tais como acionamento de motores de tração para veículos elétricos além de outros dispositivos industriais [14] [15] [16] [17].

A estratégia DTC se mostra inadequada para o controle de motores de grande porte devido à alta frequência de chaveamento do inversor que controla o motor de indução [14]. Um outro representante de técnicas vetoriais de controle de torque, o DSC (*Direct Self Control*), não mostra esta dificuldade.

Para diminuir a frequência de chaveamento em acionamentos elétricos de alta potência, o DSC foi introduzido em 1989 por Depenbrock [12]. A estrutura deste método de controle é similar ao proposto por Takahashi, porém, o fluxo de estator é controlado de forma a descrever um hexágono regular enquanto que o controle de torque é realizado de maneira semelhante ao DTC. O DSC pode ser considerado um caso particular do DTC-Takahashi.

Diversos tipos e técnicas de controle direto de torque foram propostos para melhorar o desempenho do DTC introduzido por Takahashi [11]. Alguns serão explicados e analisados a seguir. Posteriormente, será introduzido um estudo sobre os módulos DTC fabricados comercialmente, e suas aplicações.

3.2 Propostas Alternativas na metodologia DTC

A técnica de controle DTC, por ser simples e eficaz, vem sendo muito aplicada e desenvolvida não só para motores de indução, como também para outros tipos motores, como por exemplo, os *brushless DC* [18]. Porém, neste trabalho, uma atenção especial será dada às técnicas de controle de torque que utilizam o motor de indução.

Em [19], o autor apresenta um novo controle de torque e fluxo baseado em modulação por vetores espaciais (*Space Vector Modulation-SVM*) para acionamentos *sensorless* em motores de indução. A modulação por vetores espaciais possibilita uma gama maior de vetores de tensão que podem ser aplicados no motor, melhorando a regulação de torque e do fluxo do mesmo. Um outro acionamento *sensorless* para motores de indução, utilizando filtro de Kalman extendido (*Extended Kalman Filter - EKF*), foi feito em [20]. A grande vantagem do EKF é seu bom desempenho em estimar os parâmetros elétricos do motor em condições de

baixa velocidade. Todos estes estudos tiveram resultados muito satisfatórios, mostrando o bom desempenho da técnica.

Uma alternativa bastante utilizada na elaboração do acionamento DTC são algoritmos que são escritos baseados em lógica *fuzzy*. Em [21] e [22] é melhorada a principal desvantagem do DTC, alto *ripple* de torque, por meio de inteligência artificial, descrevendo um controlador DTC lógico *fuzzy*. Um mapeamento de um acionamento híbrido FOC – DTC utilizando redes neurais é apresentado em [23]. Tal acionamento executa a técnica DTC durante a condição de transiente e a técnica FOC durante a operação em regime permanente, conseguindo assim desfrutar das principais vantagens de cada uma, já que o DTC possui uma resposta dinâmica mais rápida que o FOC, porém seu desempenho em regime é pior.

Muitos trabalhos acadêmicos são desenvolvidos com o intuito de aprimorar a técnica DTC, levando-se em conta a sua principal desvantagem, alto *ripple* de torque. Alguns destes trabalhos serão analisados a seguir:

- Marino [24] fez uma comparação entre o DTC convencional (Takahashi), o DTC que utiliza a técnica de modulação por vetores espaciais (SVM-DTC) e o DTC controlado por modulação por vetores espaciais discretos (*Discrete Space Vector Modulation* – DSVM-DTC), para verificar suas influências nas condições de operação do motor em regime permanente. Uma ênfase especial foi dada à distorção da corrente de estator e ao *ripple* de torque. O DSVM-DTC dividiu o período de amostragem em três intervalos, resultando em 19 possíveis vetores para cada instante de chaveamento. O SVM-DTC obteve melhor desempenho, porém requer um esquema de controle mais complexo e necessita da resistência do estator e da indutância de dispersão além de usar a predição dos campos elétrico e magnético do motor (*back Electromagnetic Field* – *back* EMF), tornando o esforço computacional muito intenso. O DSVM-DTC é uma boa solução entre o DTC clássico e o SVM-DTC, pois apresenta pequenos *ripples* de torque.

- Uma revisão de técnicas DTC usadas recentemente foi apresentada em [25]. A técnica DTC convencional (com tabela de chaveamento baseada em histereses), assim como o DSC e o DTC com frequência de chaveamento constante e que usa modulação de vetores espaciais foram analisadas. De acordo com os estudos é comprovado que o DSC tem melhor desempenho para acionamentos de alta potência e frequência de chaveamento baixa, sendo utilizado principalmente para tração e para veículos elétricos. Esquemas com frequência de chaveamento constante (DSVM-DTC) melhora consideravelmente o desempenho do acionamento em termos de redução de pulsações do torque e do fluxo, partida suave e operação em baixa velocidade. Consequentemente, DSVM-DTC é uma solução excelente para aplicações gerais do motor de indução.

- Segundo [1], em baixas velocidades, a estimação do fluxo de estator do motor é deteriorada devido ao efeito de um valor impreciso da resistência de estator. Um erro da resistência do estator causa um desvio do vetor espacial do fluxo da trajetória circular centrada na origem. Como consequência, a magnitude do fluxo de estator exibe indesejadas oscilações, produzindo flutuações de torque. Para obter um bom desempenho DTC em baixas velocidades, um novo estimador de fluxo foi proposto pelo autor. Tal estimador permite uma drástica redução da sensibilidade do esquema de controle para um erro da resistência de estator. A operação de tal estimador é baseada na ação de um filtro passa baixa da *back* EMF do rotor.

- Um acionamento DTC *sensorless* híbrido para aplicações industriais, domésticas e automotivas foi desenvolvido em [26], usando um processador digital de sinais (*Digital Signal Processor* - DSP) TMS320F243. Na condição de operação transitória, o acionamento funciona usando o DTC clássico. Em regime permanente, usando controladores lineares de fluxo e torque, o sistema de controle gera um vetor tensão de referência para o inversor alimentando o motor de indução. O vetor é gerado usando a técnica SVM. Em regime

permanente, o acionamento possibilita a operação com baixos *ripples* de torque. Um observador de estado sem adaptações de velocidade foi empregado, o que reduziu a exigência computacional. O princípio de operação híbrido e o observador de estado do motor resultaram em um alto desempenho do sistema.

- Uma outra forma de se obter redução do *ripple* de torque é o uso de conversores multinível, tal como o trabalho [27]. Um algoritmo de controle para um acionamento DTC multinível foi introduzido. Esta técnica permite principalmente reduzir *ripples* de torque sem a necessidade de um incremento na frequência de chaveamento. Foi utilizado um conversor NPC (*neutral point clamped*) devido ao seu alto desempenho. Tal conversor permite a produção de 19 diferentes vetores de tensão, sendo caracterizados por 3 diferentes níveis de magnitude e 12 valores de fases diferentes. A técnica foi baseada no DTC de Takahashi (com algumas alterações) e utilizou comparadores de histerese de 2 e 5 níveis, para fluxo e torque respectivamente. As simulações mostraram que o acionamento DTC multinível obteve uma significativa melhora, principalmente em termos de redução de *ripple* de fluxo e torque.

- O emprego de motores não convencionais do tipo multi fases seria uma opção de obtenção de menores *ripples* de torque. Em [28], um controle direto de torque para um motor de indução de 5 fases foi apresentado. Pôde ser observada uma rápida dinâmica com baixo *ripple* no fluxo de estator e no torque do motor. O sistemas DTC de 5 fases possui 32 vetores espaciais de tensão, que propicia uma grande flexibilidade no chaveamento do inversor, tornando o controle de fluxo do estator e torque mais precisos se comparados ao DTC para motores de indução trifásicos.

3.3 Alternativas de Implementação da Técnica DTC

A técnica DTC era a princípio implementada com base em processadores digitais de sinais DSP's ou DSP's associadas a circuitos integrados para aplicações específicas (ASIC). Nestes casos a DSP processava os cálculos do modelo e os controladores de histereses

inerentes da técnica DTC. Os resultados destes cálculos serviam então como indexação de uma tabela otimizada de chaveamento armazenada em um ASIC [10].

Com o avanço da tecnologia, tornou-se possível tal implementação utilizando-se outros dispositivos, como por exemplo FPGA's, fazendo assim com que a técnica pudesse ser cada vez mais otimizada [29].

A seguir serão analisados e discutidos trabalhos que implementaram a técnica DTC em dispositivos diferentes, dando maior enfoque aos trabalhos que a implementaram utilizando FPGAs.

- Como mencionado anteriormente, uma prática usual era a implementação do DTC em dispositivos DSPs associados a ASICs. Por ser um dispositivo reprogramável, e não somente de aplicação específica, as FPGAs passaram a substituir os ASICs. Em [30], uma DSP TMS320C31 da empresa Texas foi utilizada para executar a modelagem do motor assim como os controladores de histerese da técnica DTC, e a tabela de chaveamento foi implementada em uma FPGA XC4005E da empresa Xilinx. Resultados experimentais comprovaram a excelente resposta de torque peculiar da técnica.

- Uma técnica para programar ASICs usando a linguagem VHDL aplicada para o controle de velocidade DTC de um motor de indução é apresentada em [31]. O programa é composto por transformadores de coordenadas, controladores de velocidade, estimadores de fluxo de estator e torque, controladores de fluxo de estator e torque, detector da posição do fluxo de estator, máquina de estados finitos e um vetor de chaveamento do inversor de tensão. Toda técnica é implementada apenas em uma FPGA (XC4052XL-PG411). As simulações, assim como os resultados experimentais mostraram boas respostas.

- Um algoritmo da estratégia DTC escrito em linguagem VHDL é apresentado em [10]. O algoritmo foi implementado tendo como base uma arquitetura com 5 blocos principais: modelo do motor, comparador de fluxo, comparador de torque, avaliação do setor do fluxo e

tabela de chaveamento. Cada ciclo de tempo foi particionado em 5 intervalos. O módulo do modelo do motor usou os intervalos de tempo 1 a 3. O módulo do comparador de fluxo, assim como o de avaliação do setor do fluxo e o comparador de torque foram processados em paralelo no quarto intervalo de tempo, enquanto que o último intervalo foi usado para computar a tabela de chaveamento. Neste trabalho foi usado um formato de ponto fixo complemento de dois com tamanho variável na representação numérica interna, para descrever todas as etapas da estratégia. O desempenho do código foi avaliado comparando os resultados da simulação VHDL com resultados de simulação com o aplicativo MATLAB. Um erro mínimo foi obtido entre a representação ponto fixo proposto e a representação usada no MATLAB.

- Um protótipo virtual aplicado para controle DTC de um MIT, usando VHDL-AMS (*Analogue Mixed Signal*) é apresentado em [32]. O autor mostra uma forma de como uma linguagem de alto nível (VHDL) pode ser usada para modelar um sistema incluindo vários domínios como eletrônica analógica e digital, eletrônica de potência, motor elétrico, controle, etc. Tal técnica permitiu simulações de sistemas com multi-domínios (analógico e digital) em um mesmo ambiente. A área digital foi caracterizada pela implementação do algoritmo de controle, enquanto que a área analógica foi composta por conversores de potência, sensores de corrente e tensão, filtros e conversores A/Ds. A descrição VHDL-AMS foi desenvolvida para cada componente (analógico/digital) do DTC citados anteriormente. Os componentes foram conectados no ambiente do *Simplorer Software*, e uma simulação da estratégia foi então realizada. Os resultados obtidos neste ambiente foram comparados a resultados da técnica implementada em MATLAB, observando-se bastante semelhança entre eles.

- Uma metodologia de desenvolvimento e implementação de controladores para acionamentos para velocidades variáveis usando o *software* Simulink e uma FPGA, utilizando o DTC como exemplo é apresentada em [29]. Um pacote da empresa Xilinx que permite a

simulação de funções da FPGA no ambiente Simulink foi utilizado. Tal produto é um Simulink *blockset*, chamado *System Generator*, que pode ser usado para ambos, implementação e simulação de FPGA. No entanto, o algoritmo de controle necessita ser construído apenas uma vez, ou seja, o mesmo algoritmo utilizado na simulação será utilizado na implementação física da FPGA. Com esta ferramenta, o algoritmo implementado na FPGA atuou exatamente como na simulação sem problemas de compatibilidade.

- Afim de comparar a implementação da estratégia DTC em DSP e em FPGA, o trabalho [33] analisa a implementação da estratégia nos dois dispositivos, primeiramente em uma DSP (ADMC401) e posteriormente em uma FPGA (APEX20KE200EFC484-2X). Ambos dispositivos apresentam a mesma área de silício ($1,5 \text{ cm}^2$). O chip DSP inclui conversores A/Ds necessários, enquanto que a FPGA não os possui, sendo assim necessário que tais conversores sejam adicionados à ela para obter as mesmas configurações analógicas/digitais de instrumentação. O algoritmo implementado na DSP foi escrito em linguagem C, enquanto que a linguagem VHDL foi utilizada para o caso da FPGA, porém ambos com o mesmo algoritmo. Os cálculos da estimação das grandezas do motor foi a parte que consumiu mais tempo na execução dos algoritmos de ambos os casos, principalmente por possuir operações como a extração de raiz quadrada. Os resultados das simulações e implementações mostraram que a execução da estratégia é melhor em FPGA do que em DSP, pois oferece melhor arquitetura teórica do ponto de vista da velocidade de execução do algoritmo.

3.4 Desenvolvimentos Comerciais e Aplicações Industriais

Inicialmente na elaboração desta pesquisa dedicou-se a uma análise das típicas aplicações industriais e investigação dos fabricantes que utilizam e comercializam produtos com a técnica de acionamento DTC.

Na busca por fabricantes, após intenso trabalho de investigação e busca por meios eletrônicos como a Internet, avaliou-se que apenas a empresa sueca Asea Brown-Boveri (ABB) comercializa produtos e equipamentos com a técnica DTC. Isto provém do fato de que esta empresa detém uma patente mundial da técnica DTC [34]. De fato, a procura e investimento tecnológico por um princípio de acionamento tal como o DTC foi iniciado e financiado pela ABB [35].

A ABB comercializa três tipos de módulos DTC com faixa de potência e tensão distintas, bem como tecnologias de semicondutores empregados. O ACS 600 e o ACS 800 possuem inversores que operam com IGBTs (*Insulated Gate Bipolar Transistor*), enquanto o ACS 6000 trabalha com IGCTs (*Insulated Gate Commutated Thyristor*). De acordo com [36], o ACS 600 trabalha em baixa tensão (380 a 690 V), sua potência varia de 2,2 a 3000 kW, sua faixa de frequência de conexão ao motor vai de 0 a 300 Hz e sua frequência de chaveamento do inversor IGBT possui alternativas de 4 (default), 8, 12 e 16 kHz, sendo utilizado principalmente em aplicações petrolíferas. O ACS 800 também trabalha em baixa tensão (380 a 500V), porém sua faixa de potência começa em 1,1 kW e chega a 3000 kW, com uma faixa de frequência de conexão ao motor variável entre 0 e 300 Hz e com as mesmas alternativas de chaveamento do ACS 600, sendo o inversor também composto com IGBTs. O mais potente dos módulos ABB é o ACS 6000, trabalhando em média tensão e com uma faixa de potência de 3 MW a 27 MW [37]. O ACS 6000 é um inversor de tensão que possui IGCTs operando como chaves semicondutores, podendo ser chaveado com uma frequência de até 2 kHz. Este pode ser utilizado em diversas aplicações como, por exemplo, guindastes, compressores, ventiladores e outras.

Como afirmado anteriormente na pesquisa, somente a ABB fabrica comercialmente módulos com a estratégia de acionamento DTC. Portanto, todas as aplicações citadas neste tópico são realizadas utilizando os módulos da empresa ABB, tendo a própria empresa como

fonte de informações. Devido à vasta gama de potência, os sistemas da referida empresa foram e são aplicados nas mais diferentes áreas da engenharia.

O DTC em um carro elétrico de super velocidade foi aplicado em [38] (Figura 3.1). O DTC provoca uma resposta instantânea de torque, fazendo com que o veículo acelere mais rapidamente.

Figura 3.1 - Veículo elétrico de alta velocidade.

Outra aplicação interessante do acionamento DTC é referente ao controle de máquinas de filmes plásticos [39]. O módulo é utilizado porque não precisa de um *encoder* de medição, além de conseguir um controle preciso do torque e de velocidade da máquina, o que são fatores fundamentais para a fabricação a que ela se dispõe. Outra aplicação semelhante dos módulos pode ser encontrada para controle de máquinas de papel [40]. Nesta aplicação é necessário um ótimo controle de torque e velocidade para que o papel não seja danificado.

O DTC pode ser empregado, com muita qualidade em sistemas de propulsão marinha [41]. Estes sistemas precisam de uma rápida resposta de torque, e como citado anteriormente, tal estratégia se encaixa perfeitamente neste caso.

Em ABB [42], verificam-se as vantagens de aplicação do DTC em geradores eólicos. O projeto da turbina é otimizado graças ao controle rápido do DTC. Assim, as oscilações no eixo diminuem, o stress mecânico é reduzido e a vida útil da turbina aumenta.

O DTC também é muito aplicado em moinhos [43]. Com um chaveamento otimizado, consegue-se um melhor controle da velocidade e do torque do motor, princípio básico do acionamento. Pelas mesmas características, a estratégia também é muito utilizada em aplicações petrolíferas [44].

Capítulo 4

Hardware Reconfigurável

4.1 Introdução

De acordo com [45], a computação está tendo um assombroso aumento em quase todas as atividades e áreas do conhecimento. Nos dias atuais, principalmente na engenharia, os problemas estão ficando cada vez maiores e mais complexos. Geralmente, as soluções para tais problemas requerem uma grande demanda de recursos computacionais para armazenamento e processamento dos dados. Outro agravante desses problemas é a necessidade de soluções em curtíssimos intervalos de tempo, ou até em tempo real. Grande parte desses problemas não consegue ter soluções satisfatórias (tempo de resposta e desempenho) quando estas são implementadas em *software* sequencial executado em *hardware* monoprocessador. Desta forma, estudos estão sendo realizados com o objetivo de melhorar o tempo de resposta, o desempenho geral e a qualidade dos resultados. A computação reconfigurável faz parte destes estudos e vem sendo aplicada cada vez mais para solucionar tais problemas complexos [46].

É importante tentar determinar os motivos que acarretaram o surgimento da computação reconfigurável. Analisando o uso da computação e os sistemas computacionais, verificamos que as soluções podem ser classificadas ou divididas em soluções implementadas em *Hardware* Fixo e em soluções implementadas em *Hardware* Programável através de

Software, sendo que esses tipos de implementações de soluções geralmente são denominados soluções em *hardware* e soluções em *software*. A computação reconfigurável surgiu para melhorar o desempenho, flexibilidade, custo, etc de cada um desses tipos de implementações de soluções.

As principais deficiências da implementação em *hardware* fixo são a falta de flexibilidade e a sua sub-utilização quando se trata de aplicações diversas [45]. Isso porque um dispositivo fixo não pode ser alterado após a sua fabricação e do mesmo não poder ser adaptado a certas situações que ocorrem durante a execução de diversas tarefas computacionais.

Já na implementação em *software*, a principal deficiência está relacionada com o desempenho, muitas vezes aquém do desejado na execução de aplicações ou tarefas computacionais. Apesar da grande flexibilidade, este tipo de implementação apresenta problemas como perdas de tempo ao executar alguns tipos de tarefas ou operações. Caso estas operações fossem executadas em *hardware* fixo, gastariam um menor tempo já que não necessitaria buscar instruções na memória, decodificá-las, buscar dados na memória e executar a operação requerida. Assim, quanto mais complexo é o sistema (maior número de operações), menor deve ser o desempenho em *hardware* programável tornando-se insatisfatório.

Para sanar tais problemas, surgiu a computação reconfigurável que é um tipo de implementação intermediária, entre as tradicionais soluções em *hardware* (*hardware* fixo) e *software* (*hardware* programável). Utilizando este tipo de implementação computacional pretende-se usar as principais vantagens dos dois métodos, ou seja, ter o desempenho das soluções implementadas em *hardware* fixo e a flexibilidade das soluções implementadas em *hardware* programável e *software* [47] [48] [49].

4.2 Arquiteturas Reconfiguráveis

As arquiteturas reconfiguráveis são as arquiteturas onde os blocos (módulos) lógicos construtivos básicos podem ser reconfigurados, na sua lógica ou funcionalidade interna, e os blocos de interconexão responsáveis pela interligação desses blocos lógicos construtivos e pela definição da estrutura da arquitetura, também podem ser reconfigurados. Estes blocos lógicos construtivos normalmente implementam ou são as unidades funcionais de processamento, armazenamento, comunicação ou entrada e saída de dados. As arquiteturas reconfiguráveis podem ser puramente reconfiguráveis ou então híbridas (mistas), com utilização dos modelos de *hardware* fixo e/ou *hardware* programável em conjunto com o modelo de *hardware* reconfigurável.

Outro ponto muito importante da computação reconfigurável é o paralelismo [50]. A arquitetura reconfigurável, ao invés de processar uma função através de um conjunto de instruções executadas sequencialmente ao longo do tempo, como em um processador, geralmente processam a função através de unidades funcionais configuradas no espaço (unidades lógicas mapeadas em diferentes blocos construtivos básicos dentro dos dispositivos reconfiguráveis, como os FPGAs). Tem-se, portanto, computação paralela envolvendo diferentes unidades funcionais que geram e consomem resultados intermediários. A diferença, portanto, está na computação temporal/sequencial das operações ou tarefas versus computação espacial/paralela das operações ou tarefas.

O espectro atual de sistemas computacionais reconfiguráveis e também das suas arquiteturas é muito amplo e diversificado, com destaque para os seguintes tipos: coprocessadores reconfiguráveis, processadores reconfiguráveis, computadores reconfiguráveis, sistemas embutidos reconfiguráveis e sistemas híbridos (compostos de módulos reconfiguráveis e módulos não reconfiguráveis, que podem ser implementados com *hardware* fixo ou *hardware* programável usando *software*). Uma análise da utilização e da

evolução desses sistemas nos indica uma tendência de crescimento na amplitude e na complexidade desse espectro de sistemas computacionais reconfiguráveis e de arquiteturas reconfiguráveis.

4.3 Dispositivos Reconfiguráveis

Os dispositivos que são utilizados para implementar sistemas computacionais reconfiguráveis são normalmente compostos por uma matriz de elementos reconfiguráveis. Cada elemento deve ser configurado anteriormente antes de sua utilização para realizar uma operação. Para tal, existem os *bits* de configuração (*bitstream*) dos dispositivos determinando a função que o mesmo irá desempenhar a partir do momento em que for configurado ou reconfigurado. No momento da configuração, são especificadas também as portas de entrada e de saída de cada elemento reconfigurável da matriz, e dessa forma, uma nova configuração é gerada, chamada configuração do roteamento dos dados. Esta configuração é de extrema importância para o funcionamento do dispositivo reconfigurável, pois quanto melhor o roteamento, melhor será a utilização da área do dispositivo reconfigurável, acarretando em um melhor desempenho do mesmo na execução de suas funções. O tempo de resposta do sistema a um sinal de entrada, está intimamente ligado à configuração do roteamento.

Um FPGA típico possui uma arquitetura interna composta de uma matriz de blocos lógicos configuráveis (CLB - *Configurable Logic Block*) cercados por uma rede de interconexão programável, formada de blocos de interconexão (Figura 4.1). Circundando todo o circuito, existem os blocos de entrada e saída (IOB - *Input Output Block*), que também são programáveis, e que servem como interface entre o mundo exterior e a lógica interna do dispositivo. A arquitetura de um CLB varia de família para família e de fabricante para fabricante, mas basicamente são compostos de pontos de entrada, que se conectam a blocos que implementam funções puramente combinacionais (LUT - *Lookup table*), multiplexadores que direcionam o fluxo dos sinais internamente ao CLB; e de registradores (tipicamente *flip-*

flops) que estão ligados às saídas e também podem realimentar as entradas dos geradores de funções combinacionais. Todos os elementos são configuráveis e propiciam uma grande flexibilidade para implementação de funções. A rede de interconexão programável é composta por diferentes tipos de segmentos de conexão, capazes de interligar a maioria das entradas e saídas dos CLBs entre si e aos IOBs. Isso tudo permite que circuitos complexos, máquinas de estado, e algoritmos sejam implementados nas FPGAs. Como podemos observar existem recursos disponíveis para a implementação de paralelismo na arquitetura interna dos dispositivos reconfiguráveis.

Figura 4.1 - Arquitetura interna de uma FPGA.

No que diz respeito à configuração ou reconfiguração das FPGAs, nota-se que existem algumas características peculiares, podendo a FPGA suportar uma reconfiguração estática ou dinâmica. Quando a FPGA deve ser programada antes de executar tal tarefa, e depois ela só consegue ser reconfigurada após parar de executar suas funções, diz-se que ela teve uma reconfiguração estática. A reconfiguração dinâmica é feita com a tarefa sendo executada pela FPGA.

Passada a etapa de configuração, os dispositivos reconfiguráveis estão prontos para serem usados. Normalmente, os sistemas reconfiguráveis híbridos usam um *hardware*

reconfigurável junto a um processador de propósito geral (GPP – *general purpose processor*) que o auxilia nos processos (Figura 4.2). O dispositivo reconfigurável, geralmente é uma FPGA. Este dispositivo serve para acelerar a execução de algoritmos, mapeando as partes que requerem uma demanda computacional grande para o substrato reconfigurável. Já o GPP, serve para executar o algoritmo que não pode ser eficientemente acelerado pelo dispositivo reconfigurável. Assim sendo, os sistemas reconfiguráveis híbridos utilizam, nesses casos, os dispositivos reconfiguráveis como sendo um coprocessador para o GPP.

Figura 4.2 - Sistema reconfigurável híbrido.

Outra característica marcante das FPGAs é a sua granularidade. Dispositivos mais simples (de grão fino) são indicados para aplicações de manipulação no nível dos *bits*, enquanto os mais complexos (dispositivos de grão grosso) são indicados para aplicações que envolvem computações mais complexas como manipulações de imagens e outras típicas aplicações de caminho de dados, com manipulação de dados com largura (codificação) de vários *bits*.

A seguir serão apresentadas as maneiras de se programar dispositivos tipo FPGAs.

4.4 Programação de Dispositivos Reconfiguráveis

Para se programar um dispositivo tipo FPGA, pode-se utilizar de 2 técnicas: diagramas esquemáticos ou linguagens de descrição de hardware (HDL – *Hardware Description Language*).

A solução implementada em *hardware* reconfigurável é desenvolvida utilizando ferramentas que a partir da captura do esquemático geram os bits de configuração de um determinado dispositivo reconfigurável. O esquemático é uma representação visual de portas e componentes lógicos combinacionais e sequenciais do hardware que faz parte da solução implementada no sistema reconfigurável. Há necessidade de conhecimento de componentes lógicos e desenvolvimento de *hardware* por parte do desenvolvedor. Geralmente nesse tipo de ferramenta é possível usar componentes mais complexos já prontos, como somadores, ULAs (Unidade Lógica Aritmética) e memórias, por exemplo. Essa abordagem geralmente não é aplicada a projetos grandes por causa da dificuldade que existe em se fazer uma representação gráfica de muitos componentes.

Outra abordagem é a utilização de linguagens de descrição de hardware como VHDL ou Verilog para a parte da solução que é implementada em *hardware*. Programas escritos nessas linguagens são compilados usando-se uma ferramenta que pode gerar os bits de configuração para um dispositivo específico. VHDL é uma linguagem estruturada que oferece a possibilidade de descrever o *hardware* e este ser simulado antes de sua síntese, facilitando a validação ou verificação, tanto em termos de funcionamento quanto em termos de tempos de atraso dos componentes e desempenho, sem a necessidade da prototipação do sistema.

Um programa em VHDL pode ser escrito basicamente usando dois tipos (modelos) de descrição: estrutural e comportamental. Na descrição estrutural, a organização física e topológica do sistema é descrita. Isso quer dizer que são especificadas as entradas e/ou saídas, os componentes lógicos, a interligação deles e os sinais que compõem o sistema. Existem bibliotecas em VHDL que contêm entidades que podem ser usadas nos projetos, como somadores, contadores, multiplicadores. No entanto, na abordagem usando VHDL estrutural é necessário conhecimento de projeto de *hardware*.

Na descrição comportamental, não é necessário descrever a organização física e topológica do sistema, somente o comportamento. Nessa abordagem, são descritas as funções (comportamento) do sistema. Um programa que utiliza esse tipo de descrição possui o mesmo formato de um programa fonte escrito em uma linguagem de programação de alto nível, como por exemplo C++. Essa abordagem diminui a necessidade de conhecimento em projeto de *hardware*, aumentando a facilidade de desenvolvimento do sistema. No entanto, os sistemas gerados a partir desse tipo de descrição podem não ser tão otimizado em questões de desempenho e área de dispositivo ocupada quanto os descritos em VHDL estrutural. Além disso, é necessário conhecer a linguagem e ela ainda é mais parecida com uma descrição de *hardware* do que de *software*.

O VHDL é uma linguagem padronizada pelo *IEEE 1076.1* [51] que oferece as seguintes vantagens para projetos digitais [52]:

- Padronização: reduz confusões e facilita interfaces entre ferramentas, fornecedores, e produtos diferentes;
- Suporte da indústria: com o lançamento de ferramentas de VHDL mais potentes e eficientes cresce também o compromisso das indústrias com esta tecnologia;
- Portabilidade: O mesmo código VHDL pode ser simulado e usado em ferramentas de diversos fornecedores e diferentes estágios de desenvolvimento;
- Capacidade de modelagem: VHDL foi desenvolvido para modelar todos estágios do desenvolvimento. VHDL pode descrever tanto comportamentos matemáticos como também rotinas que envolvem modelos complexos. Isso permite o uso de múltiplas arquiteturas e associá-las com o mesmo projeto durante o desenvolvimento;
- Reusabilidade: VHDL permite a criação de bibliotecas com códigos parametrizados desenvolvidos para uso futuro;

- Independência de tecnologia: As funcionalidades e comportamentos dos componentes podem ser criados em VHDL independente da tecnologia em que estes serão implementados;
- Documentação: VHDL é uma linguagem de alto nível cuja codificação pode ser considerada como uma documentação do projeto;
- Novas metodologias de desenvolvimento: VHDL cria uma nova metodologia que aumenta a produção e diminui o ciclo de desenvolvimento, abaixando os custos.

Capítulo 5

Elaboração da Plataforma Virtual de Ensaios DTC

5.1 Introdução

A plataforma virtual de ensaios DTC foi composta por dois *softwares* de simulação: o MATLAB/Simulink e o Modelsim. O MATLAB/Simulink foi responsável por simular a parte de potência do acionamento, enquanto que o Modelsim a parte de controle. Os dois trabalharam paralelamente, computando dados do projeto bidirecionalmente (co-simulação).

O MATLAB é uma linguagem técnica de computação de alto nível e um ambiente interativo para desenvolvimento de algoritmos, visualização de dados, análise de dados e computação numérica. Usando o MATLAB, pode-se resolver problemas de computação técnica mais rapidamente que com linguagens de programação tradicionais, como C, C++, e Fortran [53].

O MATLAB pode ser utilizado em um grande número de aplicações, incluindo processamento de imagem e sinal, comunicações, controle, teste e medidas, acionamentos elétricos, entre outros. Para cada área de aplicação existem *toolboxes*, que são ferramentas peculiares de tais áreas. Adicionando *toolboxes* estendidos, o ambiente MATLAB resolve classes particulares de problemas nessas áreas de aplicações.

O Simulink é uma plataforma para simulação em multi-domínios, ou seja, pode simular várias áreas de aplicação, ou seja, ferramentas de *toolboxes* diferentes. Ele provê um

ambiente gráfico interativo e permite mudanças de parâmetros dos blocos de suas bibliotecas permitindo ao usuário projetar, simular, implementar, testar o controle e processar sinais do sistema [54].

A biblioteca SimPowerSystems (PSP) complementa o Simulink com ferramentas para modelagem e simulação de circuitos elétricos básicos e complexos sistemas elétricos de potência. Tais ferramentas permitem ao usuário modelar a geração, transmissão, distribuição e consumo de potência elétrica assim como sua conversão em potência mecânica. Juntos, SimPowerSystems e Simulink provêem um eficiente ambiente para modelagem em multi-domínios e projetos de controle. Modelos de acionamentos elétricos que inclui descrições detalhadas do motor, e conversor pode ser encontrados na biblioteca SimPowerSystems. Os conversores e controladores implementam as estratégias mais comuns para controlar a velocidade e torque dos motores [55].

O *software* ModelSim é um compilador de programas VHDL que pode trabalhar simultaneamente com o MATLAB de maneira bidirecional. O *toolbox* Link for ModelSim é uma interface de co-simulação que integra o MATLAB/Simulink em um projeto para ser implementado em *hardware* como FPGAs e/ou ASICs. Isso provê uma rápida comunicação bidirecional entre MATLAB e Simulink e ModelSim HDL Simulator. Projetos tradicionais são executados no Simulink e seu ambiente de simulação suporta simulações com linguagens múltiplas do MATLAB, C, C++ e blocos Simulink. Adicionando o HDL a estas linguagens múltiplas, Link for Modelsim integra algoritmos e projetos de sistemas com implementação em *hardware* [56].

5.2 Parâmetros da Plataforma Virtual

As simulações da estratégia DTC realizadas neste trabalho foram realizadas em dois ambientes distintos interligados: o MATLAB/Simulink e o Modelsim. A Figura 5.1 mostra um diagrama de como é feita tal co-simulação.

Figura 5.1 - Co-simulação Simulink - ModelSim.

Como mostrado na Figura 5.1, primeiramente um projeto base é descrito no ambiente MATLAB/Simulink. Este projeto base conta com os elementos peculiares dos *toolboxes* do Simulink além de um bloco que habilita a troca de dados entre o Simulink e o ModelSim, encontrado na biblioteca Link for ModelSim. Neste bloco, descreve-se os elementos de entrada e saída do programa VHDL que será implementado no ModelSim, assim como os *clocks* do programa VHDL, além de escolher o modo que será feita a co-simulação (memória ou *socket*). Este bloco pode ser visto na Figura 5.2.

Figura 5.2 - Bloco utilizado para co-simulação MATLAB - ModelSim.

As entradas e saídas deste bloco são conectadas a blocos do Simulink, de acordo com especificações do projeto. Após o término da construção em ambiente Simulink, e após a compilação do código VHDL no ModelSim, a co-simulação já poderá ser realizada. Para tanto, é necessário alguns comandos no MATLAB e no ModelSim para sincronizar os dois programas (Apêndice B). Feito isso, o acionamento já poderá ser simulado.

Neste trabalho, a estratégia de acionamento DTC foi simulada. A parte de potência, como por exemplo, motor e inversor foi implementada utilizando blocos do Simulink. Já a parte de controle do acionamento, como modelagem do motor, setor espacial do fluxo, controladores de histerese e tabela de chaveamento, foi implementada em um código VHDL simulado no Modelsim [57] [58] [59]. A Figura 5.3 mostra o ambiente em que cada bloco do DTC foi simulado.

Figura 5.3 - Ambiente de simulação dos blocos do DTC.

A parte de instrumentação (conversores A/Ds) também foi realizada em ambiente Simulink. Os conversores A/Ds foram assumidos de 16 *bits* e discretizam as correntes de estator para entrarem como dados binários no código VHDL. Foram utilizados conversores A/Ds que trabalham com frequência de amostragem de 40 kHz, pois é uma frequência bastante razoável para o controle da estratégia.

O motor de indução utilizado foi do tipo gaiola de esquilo e seus parâmetros são mostrados na Tabela 5.1 a seguir:

Tabela 5.1 - Parâmetros do motor simulado.

<i>Parâmetros do Motor</i>	
Potência	1,5 cv
Par de Pólos	2
Resistência do Estator	7,56 Ω
Indutância do Estator	0,35085 H
Resistência do Rotor	3,84 Ω
Indutância do Rotor	0,35085 H
Indutância Mútua	0,33615 H
Inércia	0,027 kg.m ²
Fator Fricção	0,0001 N.m.s

O inversor do sistema simulado possui chaves ideais, já que o IGBT pode ser considerado uma chave ideal para o nível de frequência exigida na simulação. Este inversor foi escolhido porque trabalha bem na faixa de frequência escolhida para um bom controle da estratégia DTC, 20 kHz. Nas simulações, o *ripple* do retificador foi desconsiderado.

O programa VHDL que controla o motor tendo como base a estratégia DTC foi dividido em duas partes. A primeira parte, responsável por cálculos, opera em uma frequência de 40 kHz. A segunda parte, que contém a tabela de chaveamento do inversor, opera a uma frequência de 20 kHz. Isto foi feito para melhorar a precisão dos cálculos e também chavear o inversor em sua frequência de operação.

A primeira etapa realizada no programa no ModelSim foi converter os valores binários de entrada em representação de ponto flutuante seguindo o padrão *IEEE – 754* para números de precisão simples (32 *bits*). A Figura 5.4 mostra a configuração binária para tal representação.

Figura 5.4 - Número de precisão simples *IEEE 754 (single format)*.

Posteriormente, foram executados os cálculos do modelo do motor e suas respectivas comparações com seus valores de referência, além de determinar o setor espacial em que o fluxo se encontra. As saídas dos comparadores de histerese, além do setor espacial do fluxo, foram entradas para a parte do programa responsável pela tabela de chaveamento do inversor. Esta tabela então indicava a melhor alternativa para o controle DTC naquele instante e acionava os *gates* das chaves do inversor para manter o controle desejado.

5.3 Simulações

Foram feitas três simulações do sistema descrito anteriormente, e estas foram divididas em dois casos:

- Caso 1: controle de torque e o fluxo do motor (malha aberta de velocidade);
- Caso 2: controle de fluxo e velocidade do motor (malha fechada de velocidade)
 - Simulação 1: inversão da velocidade do motor;
 - Simulação 2: inserção de carga no eixo do motor.

Os resultados destas simulações serão verificados e analisados no capítulo seguinte.

5.3.1 Caso 1: Malha Aberta de Velocidade

A primeira simulação foi do circuito em malha aberta ilustrado pela Figura 5.5. Nesta simulação foi determinado um fluxo de referência de 0,8 wb. O torque de referência inicial do motor foi fixado inicialmente em 12 N.m e após 0,25s alterado para -10 N.m. Uma carga de 6 N.m foi inserida no eixo do motor no instante 0,1 s.

Figura 5.5 - Simulação da estratégia DTC em malha aberta.

5.3.2 Caso 2: Malha Fechada de Velocidade

5.3.2.1 Simulação 1: Inversão da Velocidade do Motor

O controle da velocidade do motor sem carga no eixo foi feito nesta simulação, utilizando o mesmo código da simulação anterior. O circuito utilizado pode ser visto na Figura 5.6. Foi utilizado um controlador PI para estabelecer a referência de torque necessária para o controle de velocidade desejado. A máxima referência de torque para este caso foi limitada em 25 N.m, para não danificar a mecânica do motor. O PI utilizado foi:

$$G_{PI}(s) = \frac{0,5s + 1}{s} \quad (5.1)$$

Para a dinâmica mais lenta do motor, foi estipulada uma frequência de amostragem de 1 kHz e o PI obtido foi devidamente discretizado para esta situação.

Nesta simulação foi fixada uma referência de fluxo de 0,8 wb. A referência de velocidade foi inicialmente estabelecida em 80 rad/s, mudando para -60 rad/s após um período de 1 s.

Figura 5.6 - Simulação da estratégia DTC com controle de velocidade.

5.3.2.2. Simulação 3: Inserção de Carga no Eixo do Motor

Esta simulação pode ser vista pela Figura 5.6 mostrada anteriormente. Um fluxo de 0,8 wb foi fixado como referência e a referência da velocidade foi designada para 80 rad/s. Após um intervalo de tempo de 1 s, uma carga de 4 N.m é acoplada ao eixo do motor.

Capítulo 6

Resultados das Simulações

6.1 Introdução

Neste Capítulo são apresentados os resultados das três simulações descritas no capítulo anterior.

A primeira simulação trata-se da verificação de funcionamento do código DTC aplicando-se uma referência de fluxo e uma referência de torque. A referência de fluxo foi mantida fixa, enquanto que a de torque sofreu uma alteração durante o período de simulação. Ainda nesta simulação, uma carga foi acoplada ao eixo do motor para causar uma situação próxima da real. Os resultados são apresentados e analisados.

Na segunda simulação foi feito um controle de velocidade do motor usando a técnica de controle direto do torque. Enquanto que a referência de fluxo foi mantida constante, a referência de torque foi alterada de acordo com as necessidades de velocidade do motor. Uma mudança da referência de velocidade foi realizada durante o período simulado. Os resultados são exibidos neste capítulo.

Para finalizar, os resultados da terceira simulação são apresentados na última parte deste Capítulo. Nesta simulação, também é feito um controle de velocidade do motor, porém, uma carga é acoplada ao seu eixo em um determinado instante.

6.2 Resultado das Simulações

6.2.1 Caso 1: Malha Aberta de Velocidade

Os resultados apresentados neste item são relativos à simulação do Caso 1 descrito no capítulo anterior. A Figura 6.1 mostra o fluxo de estator estimado pelo programa VHDL, enquanto que o fluxo medido no motor é ilustrado pela Figura 6.2. Nota-se que há uma pequena diferença entre o fluxo de estator estimado e o medido. As oscilações ocorridas no fluxo medido se devem à necessidade de manter o torque perto do seu valor de referência, afetando assim a dinâmica do fluxo. Porém, esta diferença não altera a dinâmica do motor e muito menos do acionamento.

A Figura 6.3 mostra o torque eletromagnético estimado pelo programa VHDL e o torque medido no eixo do motor é exibido na Figura 6.4.

Figura 6.1 - Fluxo de estator estimado pelo programa VHDL (Caso 1).

Figura 6.2 – Fluxo de estator medido no motor (Caso 1).

Figura 6.3 - Torque eletromagnético estimado pelo programa VHDL (Caso 1).

Figura 6.4 - Torque eletromagnético medido no motor (Caso 1).

Nota-se que os torques medido e estimado são bem parecidos. Mesmo com a inclusão de uma carga no eixo do motor, o torque não é alterado e permanece junto à sua referência. O estimador de torque utilizado no trabalho também é simples, o que torna inevitável os altos *ripples* de torque vistos. Estimadores mais precisos já foram descritos anteriormente no trabalho.

As componentes das correntes de estator, real e imaginária, medidas e estimadas são comparadas nas Figuras 6.5 e 6.6, respectivamente. As Figuras 6.7 e 6.8 ilustram as componentes do fluxo de estator, real e imaginária, nesta ordem.

Nota-se através das Figuras 6.5 – Figura 6.8 que os resultados estimados e medidos foram muito próximos, validando o código proposto.

A Figura 6.9 mostra o comportamento das correntes trifásicas do estator no período da simulação. A velocidade do motor é mostrada na Figura 6.10.

Figura 6.5 - Componente real da corrente de estator (Caso 1).

Figura 6.6 - Componente imaginária da corrente de estator (Caso 1).

Figura 6.7 - Componente real do fluxo de estator (Caso 1).

Figura 6.8 - Componente imaginária do fluxo de estator (Caso 1).

Figura 6.9 - Correntes trifásicas do estator (Caso 1).

Figura 6.10 - Velocidade do motor (Caso 1).

Analisando-se a Figura 6.9. nota-se a mudança de duas fases no instante 0,25s, proporcionado pela mudança do sentido de torque do motor. O período entre os instantes 0,3 s e 0,4 s mostra o comportamento da corrente quando a velocidade é bastante reduzida.

A taxa de crescimento da velocidade do motor é alterada no instante 0,1 s, instante este que é acoplada uma carga de 6 N.m ao eixo. No instante 0,25 s, a referência de torque é trocada de sentido, fazendo com que a velocidade do motor diminua e após um pequeno tempo mude seu sentido de rotação.

6.2.2 Caso 2: Malha Fechada de Velocidade

Como descrito no capítulo anterior, para o Caso 2 foram feitas duas simulações: a primeira com reversão de velocidade e a segunda com inserção de carga no eixo do motor. Os resultados de cada uma delas serão apresentados na sequência.

6.2.2.1 Simulação 1: Inversão da Velocidade do Motor

Como descrito no capítulo anterior, nesta simulação a referência da velocidade do motor foi alterada de 80 rad/s para -60 rad/s no instante 1 s. A Figura 6.11 apresenta a comparação da velocidade medida do motor com seu valor de referência.

A velocidade descreve um comportamento condizente com o controlador PI empregado. Porém, o período escolhido para a simulação é curto para que a mesma possa entrar em regime. Não foi escolhido um período de simulação maior devido à limitação de iterações do programa ModelSim.

O torque comandado pelo PI (limitado a 25 N.m) para que a velocidade descrevesse a trajetória requerida tornou-se o torque de referência da estratégia DTC. O torque estimado pelo programa VHDL, assim como o medido no motor podem ser vistos pelas Figura 6.12 e 6.13 respectivamente.

Figura 6.11 - Velocidade do motor (Caso 2, Simulação 1).

Figura 6.12 - Torque eletromagnético estimado pelo programa VHDL (Caso 2, Simulação 1).

Figura 6.13 - Torque eletromagnético medido no motor (Caso 2, Simulação 1).

As figuras anteriores mostram a boa resposta do programa em relação ao torque comandado, fazendo com que a dinâmica do motor fosse muito satisfatória.

Os fluxos de estator estimado e medido podem ser vistos respectivamente pelas Figuras 6.14 e 6.15. A comparação do fluxo estimado com o fluxo medido são um pouco diferentes devido ao descrito anteriormente, ou seja, para manter o torque perto do seu valor de referência, a dinâmica do fluxo é afetada.

As componentes real e imaginária das correntes podem ser vistas nas Figuras 6.16 e 6.17 respectivamente. As Figuras 6.18 e Figura 6.19 ilustram, nesta sequência, as componentes real e imaginária do fluxo de estator.

Figura 6.14 - Fluxo de estator estimado pelo programa VHDL (Caso 2, Simulação 1).

Figura 6.15 - Fluxo de estator medido no motor (Caso 2, Simulação 1).

Figura 6.16 - Componente real da corrente de estator (Caso 2, Simulação 1).

Figura 6.17 - Componente imaginária da corrente de estator (Caso 2, Simulação 1).

Figura 6.18 - Componente real do fluxo de estator (Caso 2, Simulação 1).

Figura 6.19 - Componente imaginária do fluxo de estator (Caso 2, Simulação 1).

Nota-se que as grandezas vistas pelas figuras anteriores têm uma alteração no instante 1 s. Isto é causado devido à mudança na referência de velocidade e, por conseguinte, no torque. Os valores medidos e estimados também foram bastante aproximados, validando ainda mais o código proposto.

As correntes trifásicas do estator são ilustradas pela 6.20. Pela figura verifica-se que as mesmas sofrem perturbações no instante 1 s devido à mudança da referência da velocidade que causa uma grande mudança na referência de torque.

Figura 6.20 - Correntes trifásicas do estator (Caso 2, Simulação 1).

6.2.2.2 Simulação 2: Inserção de Carga no Eixo do Motor

Como descrito no capítulo anterior, nesta simulação uma carga de 4 N.m foi acoplada ao eixo do motor no instante 1 s. A curva de velocidade do motor pode ser vista na Figura 6.21 a seguir:

Figura 6.21 - Velocidade do motor (Caso 2, Simulação 2).

A velocidade sofre uma ligeira redução quando a carga é acoplada ao eixo do motor, porém em um curto espaço de tempo ela retoma seu valor de referência.

Os torques estimado e medido, em função do torque comandado, podem ser vistos nas Figuras 6.22 e Figura 6.23 respectivamente. O valor da referência de torque sofre um aumento quando a carga é inserida no motor. O valor do torque estimado ficou muito próximo do valor do torque medido.

As Figuras 6.24 e Figura 6.25 ilustram os valores do fluxo estimado e medido no motor respectivamente. As componentes real e imaginária das correntes de estator podem ser vistas nas Figuras 6.26 e Figura 6.27 respectivamente. A Figura 6.28 ilustra a componente real do fluxo de estator, enquanto que a Figura 6.29 a sua componente imaginária. As correntes trifásicas do estator são ilustradas pela Figura 6.30.

Figura 6.22 - Torque eletromagnético estimado pelo programa VHDL (Caso 2, Simulação 2).

Figura 6.23 - Torque eletromagnético medido no motor (Caso 2, Simulação 2).

Figura 6.24 - Fluxo de estator estimado pelo programa VHDL (Caso 2, Simulação 2).

Figura 6.25 - Fluxo de estator medido no motor (Caso 2, Simulação 2).

Figura 6.26 - Componente real da corrente de estator (Caso 2, Simulação 2).

Figura 6.27 - Componente imaginária da corrente de estator (Caso 2, Simulação 2).

Figura 6.28 - Componente real do fluxo de estator (Caso 2, Simulação 2).

Figura 6.29 - Componente imaginária do fluxo de estator (Caso 2, Simulação 2).

Figura 6.30 - Correntes trifásicas do estator (Caso 2, Simulação 2).

As figuras anteriores mostram que os resultados medidos e estimados foram bastante próximos, fazendo com que o código VHDL seja validado. As correntes trifásicas de estator ou qualquer componente da corrente ou do fluxo não sofreram alterações bruscas quando a carga foi inserida. A única mudança foi observada na velocidade do motor e no seu torque eletromagnético.

Capítulo 7

Conclusões e Sugestões para Trabalhos Futuros

7.1 Conclusões

Foi apresentada neste trabalho uma nova forma de se projetar um acionamento tendo o *hardware* reconfigurável como tecnologia alvo. Os resultados apresentados foram bastante satisfatórios e validam o método apresentado.

A estratégia de controle de torque direto (DTC) aplicada a motores de indução foi amplamente estudada e analisada. Em muitos trabalhos, o algoritmo que seria utilizado era simulado manualmente em um compilador de *hardware* e posteriormente implementado, com um grande risco de sofrer alterações. A metodologia apresentada reduz drasticamente os riscos de alteração do código na implementação, já que o mesmo é simulado já embutido no sistema.

Um outro diferencial do trabalho é a implementação em ponto flutuante, visto que trabalhos anteriores eram feitos usando ponto fixo. A aritmética em ponto flutuante requer mais espaço no dispositivo em termos de implementação, além de executar operações de maneira mais lenta devido à sua arquitetura complexa. Porém, as vantagens conseguidas com sua utilização são bem maiores. A aritmética em ponto flutuante é de fácil manipulação, ou seja, não necessita mudança de escalas, além de conseguir abranger uma gama maior de números se comparada à aritmética em ponto fixo.

Em relação à estratégia DTC simulada, a dispersão do torque eletromagnético depende muito da largura de histerese de torque e da frequência do controle. Quanto menor a largura da banda de histerese de torque e maior a frequência do controle, menor será a dispersão do torque eletromagnético obtido.

O programa proposto mostrou-se válido, pois o torque, que é o elemento fundamental a ser controlado nesta técnica, não sofreu grandes alterações em seus valores, sendo que os medidos e estimados foram bastante próximos em todas as simulações realizadas. Além disso, todas as outras grandezas foram bem controladas como proposto pela estratégia.

O setor espacial onde se encontra o fluxo foi estimado por comparação. Foram apresentados três métodos para se determinar o setor espacial do fluxo, e o método por comparação foi escolhido por não conter operações trigonométricas, sendo que estas requerem um esforço computacional muito grande.

Foram escolhidos conversores A/D s de 16 bits e 40 kHz, pois satisfazem bem o tipo de controle desejado além de seu baixo custo em uma futura implementação. O algoritmo executa as operações matemáticas também com uma frequência de 40 kHz, porém, os pulsos para chaveamento do inversor foram atribuídos a uma frequência de 20 kHz (inversor IGBT). Estas configurações foram aprovadas para o controle desejado, e um projeto com uma precisão maior necessitaria de um inversor MOSFET, que suporta uma frequência de chaveamento maior, e A/D s mais rápidos, além de uma pequena alteração no algoritmo que o permitisse trabalhar em uma taxa de frequência maior. A estimação de velocidade nas duas simulações em que foi necessária foi feita a uma frequência de 1 kHz com resultados excelentes.

O projeto abordou uma nova maneira de se estruturar e simular um algoritmo. A tecnologia alvo adotada foi a de dispositivos reconfiguráveis, que vem crescendo incrivelmente e que vem tomando espaço no mercado industrial em todo mundo.

7.2 Sugestões para Trabalhos Futuros

Este trabalho inicia uma nova metodologia para verificação do desempenho de um algoritmo que será implementado em *hardware*. A computação reconfigurável é um campo que vem sendo muito explorado e vem ocupando, cada vez mais, lugar de destaque no setor industrial. A metodologia apresentada economiza tempo e consequentemente dinheiro, pois reduz riscos de mudanças significativas no código fonte, se tornando muito útil.

A compreensão da técnica DTC, além de outras técnicas de controle, permite o aprofundamento do conhecimento na área de acionamento com a possibilidade do aperfeiçoamento, ou até mesmo com a criação de novas e mais eficientes estratégias de controle.

Assim, este trabalho tem perspectivas de extensão, tais como:

- Implementação física da estratégia apresentada;
- Construção de algoritmos para outras técnicas de acionamento usando a metodologia do trabalho em questão.

Capítulo 8

Referências Bibliográficas

- [1] BUJA, G.; CASADEI, D.; SERRA, G. DTC-Based for Induction Motor Drives. *23rd IEEE International Conference on Industrial Electronics, Control and Instrumentation*, 9-14 November 1997, vol. 4, p. 1506 - 1516.
- [2] LINS, ZD. *Controle Direto de Torque para Motores de Indução – Estudo e Implementação*. Campinas: Faculdade de Engenharia Elétrica e Computação; Universidade de Campinas, 2001. 170 p. Tese de Doutorado em Engenharia Elétrica.
- [3] BARBI, I. *Teoria Fundamental do Motor de Indução*. Florianópolis: Editora da UFSC, 1985.
- [4] VAS, P. *Sensorless Vector and Direct Torque Control*, New York, USA: Oxford University Press Inc, 1998.
- [5] KAZMIERKOWSKI, M. P.; TUNIA, H. *Automatic Control of Converter-Fed Drives*, Elsevier Ed., 1994.
- [6] BLASCHKE, F. The Principle of Field Orientation Applied to the New Transvector Closed Loop Control System for Rotating Field Machines. *Siemens Rev.*, May 1972, vol. 39, p. 217-220.
- [7] ANDRADE, D. A. *Dynamic Control of Inverter-Fed Cage Induction Motors*. Leeds: Electronic & Electrical Engineering Department; University of Leeds, 1994. 212 p. Tese de Doutorado em Engenharia Elétrica.
- [8] HOLTZ, J. The Representation of AC Machine Dynamics by Complex Signal Flow Graphs. *IEEE Transactions on Industrial Electronics*, 1995, vol. 42, p. 263 – 271.
- [9] KOVÁCS, K. P.; RÁCZ, P. *Transient Vorgänge in Wechselstrom – Maschinen*, Budapest, Verlag der Ungar. Akad. D. Wissensch., 1959.
- [10] FERREIRA, SB. *Estudo e Avaliação do Desempenho de Estratégias de Controle Direto de Torque em Máquinas de Indução*. Porto Alegre: Faculdade de Engenharia; Pontifícia Universidade Católica, 2004. 147 p. Dissertação de Mestrado em Engenharia Elétrica.

- [11] TAKAHASHI, I.; NOGUCHI, T. A New Quick Response and High – Efficiency Control Strategy of an Induction Motor. *IEEE Industry Applications Society Annual Meeting*, 1985, p. 496 – 502.
- [12] DEPENBROCK, M.; BAADER, U. Direct Self Control (DSC) of Inverters-Fed Induction Machine: A Basis for Speed Control without Speed Measurement. *IEEE Transactions on Industry Applications*, May/June 1992, vol. 28, p. 581 – 588.
- [13] XU, X.; DONCKER, R.; NOVOTNY, D. W. A Stator Flux Oriented Induction Machine Drive. *IEEE Power Electronics Specialists Conference*, 1988, p. 870 – 876.
- [14] NASH, J. N. Direct Torque Control, Induction Motor Vector Control without an Encoder. *IEEE Transactions on Industry Applications*, March/April 1997, vol. 33, p. 333 – 341.
- [15] TIITINEN, P.; POHJALAINEN, P.; LALU, J. The Next Generation Motor Control Method: Direct Torque Control (DTC). *In Proceedings EPE Chapter Symposium*, Lausanne, Switzerland, 1994.
- [16] FAIZ, J.; HOSSIENI, S. H.; GHANEI, M.; KEYHANI, A., PROCA, A. Direct Control of Induction Motors for Electric Propulsion Systems. *Electric Power Systems Research*, 1999, p. 95 – 1001.
- [17] VASUDEVAN, M.; ARUMUGAM, R. New Direct Torque Control Scheme of Induction Motor for Electric Vehicles. *5th Asian Control Conference*, July 2004, vol. 2, p. 1377 – 1383.
- [18] LIU, Y.; ZHU, Z. Q.; HOWE, D. Direct Torque Control of Brushless DC Drives with Reduce Torque Ripple. *IEEE Transactions on Industry Applications*, March/April 2005, vol. 41, p. 599 – 608.
- [19] LASCU, C.; BOLDEA, I.; BLAABJERG, F. A Modified Direct Torque Control for Induction Motor Sensorless Drive. *IEEE Transactions on Industry Applications*, January/February 2000, vol. 36, p. 122 – 130.
- [20] XIAMNIN, M.; YOULIN, G. Extended Kalman Filter for Speed Sensorless DTC Based on DSP. *Proceedings of the 4th IEEE World Congress on Intelligent Control and Automation*, 10-14 June 2002, vol. 1, p. 119 – 122.
- [21] ARIAS, A.; ROMERAL, J. L.; ALDABAS, E.; JAYNE, M. G. Fuzzy Logic Direct Torque Control. *Proceedings of the IEEE International Symposium on Industrial Electronics*, 4-8 December 2000, vol. 1, p. 253 – 258.
- [22] YANG, J. Q.; HUANG, J. A New Full Speed Range Direct Torque Control Strategy for Induction Machine. *Proceedings of IEEE International Conference on Machine Learning and Cybernetics*, 26-29 August 2004, vol. 1, p. 568 – 573.
- [23] SAKARUNG, P.; CHATRATANA, S. Neural Network Mapping of Hybrid FOC – DTC Induction Drive. *IEEE Region 10 Conference TENCON*, 21-24 November 2004, vol. 4, p. 467 – 470.

- [24] MARINO, P.; D'INCECCO, M.; VISCIANO, N. A Comparison of Direct Torque Control Methodologies for Induction Motor. *Proceedings of IEEE Port Power Tech*, 10-13 September 2001, vol. 2, p. 6.
- [25] KAZMIERKOWSKI, M. P.; BUJA, G. Review of Direct Torque Control Methods for Voltage Source Inverter-Fed Induction Motors. *29th IEEE Annual Conference of the Industrial Electronics Society*, 2-6 November 2003, vol. 1, p. 981-991
- [26] LASCU, C.; TRZYNADLOWSKI, A. M. A Sensorless Hybrid DTC Drive for High-Volume Low-Cost Applications. *IEEE Transactions on Industrial Electronics*, October 2004, vol. 51, p. 1048 – 1055.
- [27] DAMIANO, A.; GATTO, G.; MARONGIU, I.; PERFETTO, A. An Improved Multilevel DTC Drive. *32nd Annual Power Electronics Specialists Conference*, 17-21 June 2001, vol. 3, p. 1452 – 1457.
- [28] TOLİYAT, H. A.; XU, H. A Novel Direct Torque Control (DTC) Method for Five-Phase Induction Machines. *5th Annual IEEE Applied Power Electronics Conference and Exposition*, 6-10 February 2000, vol. 1, p. 162 – 168.
- [29] RICCI, F.; LE-HUY, H. An FPGA-Based Rapid Prototyping Platform for Variable-Speed Drives. *28th Annual Conference of the Industrial Electronics Society*, 5-8 November 2002, vol. 2, p. 1156 – 1161.
- [30] YATIM, A. H. M.; IDRIS, N. R. N. Implementation of Direct Torque Control of Induction Machine Utilizing TMS320C31 Digital Signal Processor. *6th IEEE International Symposium on Signal Processing and its Applications*. 13-16 August 2001, vol. 2, p. 627 – 630.
- [31] KIM, S. J.; LEE, H. J.; KIM, S. K.; KWON, Y. A. ASIC Design for DTC Based Speed Control of Induction Motor. *Proceedings of IEEE International Symposium on Industrial Electronics*, 12-16 June 2001, vol. 2, p. 956 – 961.
- [32] FEKI, S.; FAKHFAKH, A.; MASMOUDI, N.; HERVE, Y. Virtual Prototyping and System Level Optimization with VHDL-AMS Application for the DTC Control. *IEEE International Conference on Industrial Technology*, 8-10 December 2004, vol. 1, p. 61 – 65.
- [33] LLOR, A.; ALLARD, B.; LIN-SHI, X.; RETIF, J. M. Comparison of DTC Implementations for Synchronous Machines. *35th IEEE Annual Power Electronics Specialists Conference*, 20-25 June 2004, vol. 5, p. 3581 – 3587.
- [34] JOHNSON, A. T. *In-Line Electric Motor Driven Compressor*. United States of America. Patente industrial, no. 365362. 30-07-1999.
- [35] ABB: DIRECT TORQUE CONTROL [on-line]. [19-07-2006]. Available from: <http://www.emeraldinsight.com/Insight/ViewContentServlet?Filename=/published/emeraldaabstractonlyarticle/pdf/0490220206.pdf>.

[36] ABB DEVICES OVERVIEW [on-line]. [19-07-2006]. Available from:
[http://library.abb.com/GLOBAL/SCOT/SCOT211.nsf/VerityDisplay/C125698F006840E9C1256CB400538AF8/\\$File/63_047se.pdf](http://library.abb.com/GLOBAL/SCOT/SCOT211.nsf/VerityDisplay/C125698F006840E9C1256CB400538AF8/$File/63_047se.pdf)>.

[37] DRIVE ACS 6000 [on-line]. [19-07-2006]. Available from:
[http://library.abb.com/GLOBAL/SCOT/SCOT216.nsf/VerityDisplay/669E6D5C4C933236C1256EEC002EFA8E/\\$File/ACS%206000%20Rev.A%20PT.pdf](http://library.abb.com/GLOBAL/SCOT/SCOT216.nsf/VerityDisplay/669E6D5C4C933236C1256EEC002EFA8E/$File/ACS%206000%20Rev.A%20PT.pdf)>.

[38] RECORD-BREAKING ELECTRIC CAR WILL RELY ON ABB TECHNOLOGY [on-line]. [19-07-2006]. Available from:
<http://www.abb.com/global/abbzh/abbzh251.nsf!OpenDatabase&db=/global/abbzh/abbzh250.nsf&v=553E&e=us&url=/global/seitp/seitp202.nsf/0/60042AD9EAA7DCBF85256E3E007AA512!OpenDocument>>.

[39] FILM MAKER SLASHES COSTS WITH ABB DRIVES [on-line]. [19-07-2006]. Available from:
<http://www.abb.com/global/gbabb/gbabb901.nsf!OpenDatabase&db=/global/gbabb/gbabb905.nsf&v=F4A&e=us&url=/global/seitp/seitp202.nsf/0/573C5963D87F64EDC1256DB4003CAE90!OpenDocument>>.

[40] DIRECT DRIVE SYSTEM FOR THE PULP AND PAPER INDUSTRY [on-line]. [19-07-2006]. Available from:
[http://www.abb.com/global/seitp/seitp161.nsf/0/a38b24a879951902c2256f0000369e79/\\$file/DirectDrive.pdf](http://www.abb.com/global/seitp/seitp161.nsf/0/a38b24a879951902c2256f0000369e79/$file/DirectDrive.pdf)>.

[41] RELIABLE MARINE PROPULSION [on-line]. [19-07-2006]. Available from:
[http://www.abb.com/GLOBAL/SEITP/SEITP161.NSF/viewunid/0D1D2C6CF1DFF72AC1256961004DC1B7/\\$file/Reliable0501.pdf](http://www.abb.com/GLOBAL/SEITP/SEITP161.NSF/viewunid/0D1D2C6CF1DFF72AC1256961004DC1B7/$file/Reliable0501.pdf)>.

[42] GENERATORS AND DRIVES FOR WIND POWER [on-line]. [19-07-2006]. Available from:
[http://library.abb.com/GLOBAL/SCOT/scot259.nsf/VerityDisplay/5BCCDBCD256F7493C2256DA600427CDC/\\$File/Generators%20and%20Drives%20for%20Wind%20Power%20brochure.pdf](http://library.abb.com/GLOBAL/SCOT/scot259.nsf/VerityDisplay/5BCCDBCD256F7493C2256DA600427CDC/$File/Generators%20and%20Drives%20for%20Wind%20Power%20brochure.pdf)>.

[43] ROLLING MILL AUTOMATION THROUGHOUT THE WORLD [on-line]. [19-07-2006]. Available from:
<http://www.abb.com/global/deabb/deabb200.nsf!OpenDatabase&db=/global/deabb/deabb204.nsf&v=21B2&e=ge&url=/global/seitp/seitp202.nsf/0/5231F457D9DF16E8C1256DAA002CCBEB!OpenDocument>>.

[44] VARIABLE SPEED DRIVES (VSD) ACS 600 OIL PALS PRU [on-line]. [19-07-2006]. Available from:
[http://www.abb.com/GLOBAL/SEITP/seitp161.nsf/viewunid/F6E98C691AB3150885256B3A007CB2AA/\\$file/ACS+600OILver+1.1OilField+Application.pdf](http://www.abb.com/GLOBAL/SEITP/seitp161.nsf/viewunid/F6E98C691AB3150885256B3A007CB2AA/$file/ACS+600OILver+1.1OilField+Application.pdf)>.

[45] COMPUTAÇÃO RECONFIGURÁVEL: CONCEITOS, TENDÊNCIAS E APLICAÇÕES [on-line]. [19-07-2006]. Available from:
<http://www.inf.ucp.br/nsb/downloads/arquivos/ComputacaoReconfiguravel.pdf>>.

- [46] VILLASENOR, J.; MANGIONE-SMITH, W. H. Configurable Computing. Scientific American, June 1997, vol. 276, p. 54 - 9.
- [47] ORDONEZ, E. D. M.; SILVA, J. L. *Computação Reconfigurável – Experiências e Perspectivas*. Fundação de Ensino Eurípedes Soares da Rocha (FEESR), 2000.
- [48] Anais do I Seminário de Computação Reconfigurável, Editor Carlos Augusto Paiva da Silva Martins, Instituto de Informática, PUC Minas, 2001.
- [49] VICENTELLI, A. S. Some Considerations on Field Programmable Gate Arrays and their Impact on System Design. *Lectures Notes in Computer Science*, June 1993, p. 26 – 34.
- [50] RODRIGUES, MI. *Solução para Aritmética em Ponto Flutuante Implementada em Computação Reconfigurável*. São Carlos: Instituto de Ciências Matemáticas e Computação; Universidade de São Paulo, 2004. 88 p. Dissertação de Mestrado em Ciências da Computação e Matemática Computacional.
- [51] AMERICAN NATIONAL STANDARD, *IEEE Standard VHDL Language Reference Manual (IEEE Std 1076-1993)*, Institute of Electrical and Electronics Engineers, 1993.
- [52] CHANG, K. C. *Digital Design and Modeling with VHDL and Synthesis*, IEEE Computer Society, 1997.
- [53] MATLAB 7.2 [on-line]. [19-07-2006]. Available from: <<http://www.mathworks.com/products/matlab/description1.html>>.
- [54] SIMULINK – Simulation and Model-Based Design [on-line]. [19-07-2006]. Available from: <<http://www.mathworks.com/products/simulink>>.
- [55] SIMPOWERSYSTEMS 4.2 – Model and Simulate Electrical Power Systems [on-line]. [19-07-2006]. Available from: <<http://www.mathworks.com/products/simpower>>.
- [56] LINK FOR MODELSIM 2.0 – Cosimulate and Verify VHDL and Verilog Using ModelSim [on-line]. [19-07-2006]. Available from: <<http://www.mathworks.com/products/modelsim>>.
- [57] CASTOLDI, M. C.; AGUIAR, M. L. Simulation of DTC Strategy in VHDL Code for Induction Motor Control. *Proceedings of IEEE International Symposium on Industrial Electronics*, 9-13 July 2006.
- [58] CASTOLDI, M. C.; AGUIAR, M. L.; OLIVEIRA Jr., A. A.; MONTEIRO, J. R. B. A. Protótipo Virtual da Estratégia DTC Aplicada a Motores de Indução Usando Linguagem VHDL. *Congresso Brasileiro de Autômata*, 3-6 October 2006.
- [59] CASTOLDI, M. C.; AGUIAR, M. L.; OLIVEIRA Jr., A. A.; MONTEIRO, J. R. B. A. A Rapid Prototype Design to Investigate the FPGA- Based DTC Strategy Applied to the Speed Control of Induction Motors. *Proceedings of IEEE International Conference on Industrial Technology*, 15-17 December 2006.
- [60] KOREN, I. *Computer Arithmetic Algorithms*, Prentice-Hall, 1993.

- [61] HWANG, K. *Computer Arithmetic: Principles, Architecture and Design*, John Wiley & Sons, 1976.
- [62] FLORES, I. *The Logic of Computer Arithmetic*, Prentice-Hall, 1963.
- [63] GOLDBERG, D. *What Every Computer Scientist Should Know About Floating-Point Arithmetic*, ACM Computing Surveys, March 1991, vol. 23, p. 5 – 48.
- [64] AMERICAN NATIONAL STANDARD, *ANSI/IEEE Std 754-1985, IEEE Standard for Binary Floating-Point Arithmetic*, Institute of Electrical and Electronics Engineers, 1985.
- [65] QUIXILICA FLOATING POINT FPGA CORES [on-line]. [19-07-2006]. Available from: <http://www.techway.fr/pdf/quxilica/quxilica_floating_point_fpga_cores.pdf>.
- [66] GOLDBLATT, K. *Adapting ASIC Designs for Use with Spartan FPGAs*, Xilinx Application Note, 1998.
- [67] Link for ModelSim For Use with MATLAB and Simulink [on-line]. [21-10-2006]. Available from:
< http://www.mathworks.com/access/helpdesk/help/pdf_doc/modelsim/modelsim.pdf>

Apêndice A

Aritmética de Ponto Flutuante

A.1. Introdução

A notação em ponto flutuante representa basicamente números reais na notação científica. A notação científica é geralmente constituída por um número base e um expoente. Por exemplo, o número 123,456 podia ser representado como $1,23456 \times 10^2$. Na representação hexadecimal, o número 123,abc pode ser representado como $1,23abc \times 16^2$. Ponto flutuante resolve um grande número de problemas da representação. A notação em ponto fixo tem uma janela fixa de representação, que a limita de representar números muito grandes ou muito pequenos. Também, de ponto fixo é comum uma perda da precisão quando dois números grandes são divididos. Ponto flutuante, na outra mão, emprega uma forma de precisão apropriada à escala do número.

A.2. Análise da Aritmética de Ponto Flutuante

Um grande problema com a aritmética de ponto flutuante é que ela não segue as regras padrões da álgebra. Números inteiros também não seguem as regras padrões da álgebra, porque a representação deles é feita com números finitos de *bits*. Não se pode representar nenhum valor inteiro acima do ‘valor máximo’ ou abaixo do ‘valor mínimo’ suportado por determinada representação binária. Além deste problema, um outro é encontrado na representação em ponto flutuante: números inteiros são um subconjunto dos números reais.

Assim, os valores de ponto flutuante representam o mesmo conjunto de inteiros infinitos. Contudo, há um número infinito de valores entre quaisquer dois números reais, tornando pior o problema. Entretanto, da mesma forma que foram limitados os valores na faixa entre máximos e mínimos, não se pode representar todos os valores entre essa faixa [60] [61] [62] [63].

Para representar números reais, o mais comum é o uso da notação científica, usando a maior parte dos dígitos para representar a mantissa e a menor parte para representar o expoente (Figura A.1). Assim, estes números de ponto flutuante podem somente ser representados através de um número específico de dígitos significativos. Isto tem um grande impacto na operação com aritmética de ponto flutuante. É fácil ver o impacto de utilizar precisão limitada na aritmética de ponto flutuante: adotando um formato simplificado de ponto flutuante decimal, por exemplo, com uma mantissa com três dígitos significativos e um expoente decimal com dois dígitos, sendo a mantissa e o expoente com valores sinalizados.

Expoente	Parte Fracionária (Mantissa)
----------	------------------------------

$$\text{Valor} = \text{mantissa} * \text{base}^{(\text{expoente})}$$

Figura A.1 - Representação para números em ponto flutuante.

Em notação científica, ao se adicionar ou subtrair dois números, é necessário o ajuste de seus expoentes para o mesmo valor. Por exemplo, antes de adicionar 2.23e1 à 6.56e0, ajustam-se os valores para que eles tenham o mesmo expoente. Um caminho a seguir é converter 6.56e0 para 0.656e1 e então adicionar à 2.23e1, produzindo como resultado 2.886e1. Infelizmente, o resultado não cabe em três dígitos significativos; assim, é preciso truncá-lo em três dígitos significativos. Arredondamento, geralmente produz uma melhor precisão, e assim obtém-se 2.89e1 (Figura A.2). Como se pode ver, a perda de precisão afeta a exatidão do valor.

Figura A.2 - Comparação entre arredondamento e truncamento.

No exemplo anterior foi possível arredondar o resultado porque os cálculos foram feitos com cinco dígitos significativos. Se os cálculos estivessem limitados a três dígitos significativos, isto é, a FPU (*Float Point Unit*) trabalhando internamente com três dígitos, ocorreria o truncamento do último dígito, resultando assim 2.88e1 ao invés de 2.89e1, que contém um erro maior do que o resultado arredondado. Com isso conclui-se que: dígitos extras durante a computação melhoram a exatidão dos resultados.

A perda de exatidão durante um simples cálculo não seria muito preocupante. No entanto, quando se calcula uma seqüência de operações em ponto flutuante, o erro pode ir acumulando e produzir um grande efeito sobre o resultado final.

A multiplicação e a divisão não sofrem dos mesmos problemas da adição e subtração já que não necessitam ajuste de expoente antes da operação. Tudo que é preciso fazer é adicionar os expoentes e multiplicar a mantissa (ou subtrair o expoente e dividir a mantissa). Multiplicação e divisão não produzem particularmente maus resultados. No entanto, eles tendem a multiplicar algum possível erro já existente nos operadores.

Freqüentemente, aplicando transformações normais da álgebra, pode-se reorganizar uma equação de modo que as multiplicações e divisões ocorram primeiro. Por exemplo, supondo a realização do cálculo: $x \cdot (y + z)$. Para isso, soma-se y com z e depois multiplica esse resultado por x . No entanto, tem-se mais exatidão no resultado final transformando a equação

$x * (y + z)$ em $x * y + x * z$, e proceder aos cálculos realizando as multiplicações primeiro, evitando, assim, multiplicar o erro produzido pela adição [50].

Multiplicações e divisões também têm seus próprios problemas. Um destes problemas pode ocorrer ao se multiplicar dois valores muito grandes ou muito pequenos, possibilitando assim o aparecimento de *overflow* ou *underflow* (um resultado maior ou menor que os valores máximos e mínimos suportados). A mesma situação acontece ao dividir um pequeno valor por um grande valor ou ao dividir um grande valor por um pequeno valor. Esses casos precisam ser tratados em particular.

A comparação em ponto flutuante é muito perigosa. Valores trazem imprecisões com eles devido a algum cálculo anterior. Assim, não se deve comparar dois valores de ponto flutuante para testar sua igualdade. Diferentes cálculos podem produzir o mesmo resultado (matematicamente), porém, podem diferenciar em algum dos seus dígitos significativos. Para fazer tal comparação, determina-se qual o erro (tolerância) permitido na comparação e verificar se os valores estão dentro desta faixa de erro. Este valor do erro precisa ser cuidadosamente escolhido. Ele deve ser maior que a soma dos erros produzidos nos cálculos. O valor exato dependerá do formato particular do ponto flutuante e da maneira que os cálculos são realizados.

A.3. Padrão IEEE - 754

A fim de padronizar a representação e a implementação de operações que envolvem cálculo em ponto flutuante um grupo de trabalho da *IEEE Computer Society* foi formado e após muito trabalho foi definido o padrão *ANSI/IEEE Std 754-1985* [64]. Este novo padrão tinha como objetivo facilitar a portabilidade de programas entre diferentes computadores.

O formato da palavra binária em ponto flutuante tem um total de bits dado por $m_w + e_w$, onde m_w e e_w são as definições genéricas do tamanho da mantissa e do expoente

respectivamente. Um número em ponto flutuante é dividido em três campos como mostrado na Figura 5.3. [65]:

Figura A.3 - Formato de ponto flutuante.

O valor de um número f , representado em ponto flutuante e normalizado pelo *IEEE-754* é calculado de acordo com (A.1):

$$f = (-1)^S 1.M \times 2^{E-E_{bias}} \quad (\text{A.1})$$

onde:

M - mantissa fracionária;

S - bit do sinal;

E - expoente;

E_{bias} - excesso do expoente.

O expoente E , tem um tamanho de e_w bits e é uma representação sem sinal, onde seu valor binário é a soma dos valores dos expoentes com sinal originais e o excesso do expoente (E_{bias}). O bit do sinal S indica se o número é positivo ou negativo. Seu valor é zero para um número positivo e 1 para um negativo. A mantissa fracionária M é uma fração ponto fixo sem sinal, em que $1.0 \leq \text{mantissa} < 2.0$. O bit mais significativo é sempre 1, e é omitido, assim somente os bits m_w-1 são armazenados.

O padrão *IEEE-754* [64] padroniza o tamanho de bits da mantissa e do expoente para a representação de número em ponto flutuante, como mostrado na Tabela A.1. Também estão reservados outros dois formatos estendidos para ponto flutuante: ‘simples estendido’ e ‘duplo estendido’, com 43 bits ou mais e com 79 bits ou mais, respectivamente.

Tabela A.1 - Padrão de ponto flutuante IEEE – 754.

<i>Item</i>	<i>Precisão Simples</i>	<i>Precisão Dupla</i>
Bits de sinal	1	1
Bits de expoente	8	11
Bits de mantissa	23	52
Total de bits	32	64
Excesso do expoente	127	1023
Intervalo do expoente	-126 a +127	-1022 a 1023
O menor normalizado	2^{-126}	2^{-1022}
O maior normalizado	Aprox. 2^{+128}	Aprox. 2^{+1024}
Intervalo decimal	Aprox. 10^{-38} a 10^{+38}	Aprox. 10^{-308} a 10^{+308}
O menor, não normalizado	Aprox. 10^{-45}	Aprox. 10^{-324}

Além dos números normalizados, o padrão *IEEE-754* provê quatro outros tipos listados na Figura A.4.

Em números não-normalizados, o *bit* “1” implícito à esquerda do ponto binário se torna um “0”. Números não-normalizados podem distinguir-se dos normalizados, pois não é permitido aos números normalizados ter o expoente apenas de zeros.

Quando ocorre um *overflow* é fornecida uma representação especial para infinito, que consiste em um expoente com todos os *bits* em “1” (não permitido para números normalizados) e uma mantissa igual a zero.

Este número pode ser usado como operando que se comporta como as regras matemáticas usuais de infinito. Por exemplo, infinito mais qualquer coisa é infinito. Similarmente, qualquer número dividido por zero resulta em infinito. O resultado é indefinido quando se tem uma operação de infinito dividido por infinito. Para tratar deste caso usa-se

outro formato especial, denominado NaN (*Not a Number*). Este pode também ser tratado como operando com resultados imprevisíveis [66].

Normalizado		
$\begin{matrix} + \\ - \end{matrix}$	$0 < E < \text{Max}$	Qualquer padrão de bits
Não - Normalizado		
$\begin{matrix} + \\ - \end{matrix}$	0	Qualquer padrão de bits diferente de zero
Zero		
$\begin{matrix} + \\ - \end{matrix}$	0	0
Infinito		
$\begin{matrix} + \\ - \end{matrix}$	1111111...1	0
Not-a-Number		
$\begin{matrix} + \\ - \end{matrix}$	1111111...1	Qualquer padrão de bits diferente de zero

Figura A.4 - Tipos de notação em ponto flutuante definidos pelo padrão *IEEE- 754*.

Apêndice B

Co-Simulação Simulink/ModelSim

O *toolbox* Link for ModelSim é uma *interface* de co-simulação que integra ferramentas do ambiente Simulink (*Electronic Design Automation* – EDA) para desenvolvimento de FPGAs e ASICs. Tal interface provê um *link* bidirecional rápido entre o simulador de linguagem de descrição de *hardware* da Mentor Graphics, ModelSim, e os produtos MATLAB e Simulink, da MathWorks, para verificação e co-simulação de projetos diretos de *hardware*. A integração destas ferramentas permite aos usuários a aplicação do que cada produto tem de melhor [67]:

ModelSim – modelagem de *hardware* em HDL e simulação;

MATLAB – computação numérica, desenvolvimento de algoritmo, e visualização;

Simulink – simulação de sistemas com modelos complexos;

A *interface* Link for ModelSim consiste de funções do MATLAB e comandos do Modelsim para estabelecer *links* de comunicação entre os dois ambientes. Uma biblioteca, para incluir projetos HDL em modelos do Simulink para co-simulação, é adicionada ao Simulink.

Em geral se faz a simulação de um programa genérico testando-o separadamente com entradas específicas (inspeção manual). O Link for ModelSim conecta ferramentas que tradicionalmente são usadas para realizar partes específicas e em separado no processo de desenvolvimento do projeto. Por conectar estas ferramentas, o *toolbox* simplifica a verificação, permitindo uma co-simulação direta da implementação e da especificação original. O resultado final é uma significativa economia de tempo e eliminação de erros encontrados na comparação e inspeção manuais.

B.1 Conexão Simulink/ModelSim

Quando conectado ao Simulink, o ModelSim responde aos estímulos recebidos dos blocos de co-simulação do modelo Simulink (Figura 5.1). A inicialização da sessão de co-simulação deve ocorrer no Simulink. Uma vez que a sessão é iniciada, pode-se usar o ModelSim e o Simulink para monitorar o progresso da simulação e os resultados.

Como mostrado na Figura B.1 a seguir, múltiplos blocos de co-simulação em um modelo Simulink pode estimular várias instâncias do ModelSim, usando unicamente portas *socket* TCP/IP (*Transmission Control Protocol/Internet Protocol*). Tipicamente numa ligação TCP/IP existe aquele designado de servidor (ModelSim), que abre um *socket* e espera passivamente por ligações de um lado, e o cliente (Simulink) do outro. O cliente inicia a ligação enviando um pacote TCP/IP e espera-se que o servidor aceite a ligação enviando um outro pacote. Se, durante um determinado espaço de tempo, esse pacote não for recebido ocorre um *timeout* e o pacote do cliente é reenviado. O estabelecimento da ligação é concluído por parte do cliente, confirmando a aceitação do servidor.

Figura B.1 - Conexão Simulink/Modelsim.

B.2 Modos de Comunicação

O modo de comunicação utilizado pelo Link for ModelSim para conectar o ModelSim ao MATLAB ou Simulink depende um tanto se a simulação será realizada em uma configuração de sistema simples (local) ou em uma configuração de rede. Se o ModelSim e os

produtos da MathWorks podem ser executados localmente no mesmo sistema, e sua aplicação requerer apenas um canal de comunicação, o projetista pode escolher a entre comunicação com memória compartilhada ou *socket* TCP/IP. Comunicação com memória compartilhada provê um desempenho ótimo e é o modo padrão de comunicação. O Modo de *socket* TCP/IP é mais versátil. Pode-se usá-lo para configurações de sistemas locais e em rede.

Para configurações em que o ModelSim e os produtos da MathWorks residem em diferentes sistemas, cada sistema deve ser configurado para a *Ethernet* e deve-se usar a comunicação por *socket* TCP/IP.

B.3 Conectando Simulink e ModelSim

Este tópico mostra os passos básicos de configuração para conectar uma aplicação no ModelSim que usa o Simulink e o bloco de co-simulação HDL para verificar um código nesta linguagem. O bloco de co-simulação HDL simula um *hardware* aplicando sinais de entrada e verificando os sinais de saída de um modelo HDL em simulação no ModelSim. Neste tutorial, serão descritas as etapas para co-simular um modelo VHDL. Para melhor compreensão do processo, o método será dividido em três etapas. A primeira será executada no ModelSim, enquanto que a segunda no ambiente Simulink. A terceira etapa será executada com os dois programas trabalhando simultaneamente.

Primeira Etapa – ModelSim

Um cenário típico do Simulink e do ModelSim é criar um modelo para um componente de *hardware* específico no ModelSim que posteriormente precisará ser integrado em um modelo Simulink. A fonte HDL deve ser compilada no ambiente do ModelSim para que este gere os códigos de simulação necessários quando ocorrer a co-simulação. Os

comandos de criação e compilação de códigos HDL no Modelsim podem ser encontrados em um tutorial específico no HELP deste programa.

Segunda Etapa – Simulink

Nesta etapa, será mostrado como configurar o bloco de co-simulação HDL (Figura 5.2). Primeiramente deve-se criar o sistema a ser simulado no ambiente simulink. Posteriormente deve-se configurar o bloco de co-simulação, que representa o modelo escrito em VHDL. Alguns dos aspectos importantes desta configuração são delineados na sequência.

Na janela “*Ports*” devem ser configurados os sinais de entrada e saída do código VHDL. No campo *Full HDL Name*, descrever o sinal que está sendo inserido, assim como sua natureza (entrada ou saída). Além disso, os períodos das saídas (apenas das saídas) devem ser determinados e descritos no campo *Sample Time*. É importante salientar que o bloco de co-simulação HDL é um bloco de execução discreto e portanto somente a cada período especificado é que irá ocorrer um passo de execução do VHDL no ModelSim. Com as configurações acima, a janela *Ports* deve aparecer parcialmente como visto pela Figura B.2 a seguir:

Figura B.2 - Configuração da janela *Ports*.

Na sequência, a janela *Clocks* deve ser configurada. No campo *Full HDL Name*, especificar o *path* do sinal. Posteriormente selecionar *Rising* ou *Falling* no campo *Edge*. Especificar o período no campo *Period*. A janela *Clocks* deve aparecer como mostra a Figura B.3:

Figura B.3 - Configuração da janela *Clocks*.

Posteriormente, a janela *Timescales* deve ser configurada designando que 1 segundo no Simulink corresponda a 1 segundo no ModelSim, como mostrado pela Figura B.4 a seguir:

Figura B.4 - Configuração da janela *Timescales*.

Para finalizar a configuração do bloco de co-simulação HDL, deve-se configurar os parâmetros de conexão (*Connection*). Selecionar *socket* em *Connection method*. Esta opção especifica que Simulink e ModelSim irão se comunicar via um *socket* TCP/IP designado. Observar que dois campos adicionais, *Port number or service* e *Host name*, são agora visíveis. Notar que, por causa da opção *ModelSim running on this computer* estar selecionado por *default*, o campo *Host Name* está desabilitado. Nesta configuração, ambos Simulink e ModelSim são executados no mesmo computador. No campo *Port number or service*, especifica-se o número 4449 para a porta do *socket* ou, se esta porta não estiver disponível no sistema, outro número de porta válida. A janela *Connection* deve aparecer como indicado pela Figura B.5:

Figura B.5 - Configuração da janela *Connection*.

A janela *Tcl* não precisa ser modificada. Com bloco de co-simulação configurado, basta agora conecta-lo no sistema do Simulink para finalizar esta etapa.

Terceira Etapa – Simulink/ModelSim

Para executar o co-simulação, o ModelSim deve ser inicializado a partir de um comando na janela do ambiente MATLAB: `vsim('socketsimulink', 4449)`, para conexão via

socket TCP/IP. Este comando indica o tipo de conexão e o número da porta selecionada para co-simulação. Este comando abrirá o ModelSim. O comando *vsimulink* (usado para abrir as instâncias da entidade VHDL para co-simulação com o Simulink) deve ser executado na sequência no programa ModelSim. Os programas agora, estão conectados.

Iniciar a simulação no ambiente Simulink. Os dois programas são executados simultaneamente. Se for necessário interromper uma simulação, deve-se pará-la primeiramente no Simulink e posteriormente no ModelSim.

Há um tutorial eletrônico no HELP do MATLAB explicando passo a passo os comandos para conectar o Simulink e o ModelSim para co-simulação. A MathWorks também disponibiliza um tutorial na internet [67].