IMPLEMENTAÇÃO DIGITAL DE REDES NEURAIS ARTIFICIAIS PARA O CONTROLE DE MOTOR DE INDUÇÃO

ANDRÉ MUNIZ SOARES

CAMPO GRANDE

2006

UNIVERSIDADE FEDERAL DO MATO GROSSO DO SUL
PROGRAMA DE PÓS-GRADUAÇÃO
EM ENGENHARIA ELÉTRICA

IMPLEMENTAÇÃO DIGITAL DE REDES NEURAIS ARTIFICIAIS PARA O CONTROLE DE MOTOR DE INDUÇÃO

Dissertação submetida à
Universidade Federal de Mato Grosso do Sul
como parte dos requisitos para
obtenção do grau de Mestre em Engenharia Elétrica.

ANDRÉ MUNIZ SOARES

Campo Grande, Dezembro de 2006.

IMPLEMENTAÇÃO DIGITAL DE REDES NEURAIS ARTIFICIAIS PARA O CONTROLE DE MOTOR DE INDUÇÃO

André Muniz Soares

'Esta Dissertação foi julgada adequada para obtenção do Título de Mestre em Engenharia Elétrica, Área de Concentração em Energia Elétrica, e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Mato Grosso do Sul - Campo Grande - MS.'

_____

Luciana Cambraia Leite, Dra.
Orientadora

_____

João Onofre Pereira Pinto, Ph.D.
Co-orientador
Coordenador do Programa de Pós-Graduação em Engenharia Elétrica

Banca Examinadora:

_____

Luciana Cambraia Leite, Dra.
Presidente

_____

João Onofre Pereira Pinto, Ph.D.

_____

Luiz Eduardo Borges da Silva, Ph.D.

_____

Flávio Alessandro Serrão Gonçalves, Dr.

_____

Milton Ernesto Romero Romero, Ph.D.

A Deus que sempre esteve ao meu lado nessa caminhada

Aos meus pais e irmãs, pelo amor, dedicação e apoio.

AGRADECIMENTOS

Aos professores Dra. Luciana Cambraia Leite e Dr. João Onofre Pereira Pinto, minha orientadora e co-orientador, respectivamente, pelo apoio e pelas contribuições que foram fundamentais para o desenvolvimento deste trabalho.

Aos Professores Luiz Eduardo Borges da Silva e Milton Ernesto Romero, pelas valiosas contribuições na etapa de Qualificação desta Dissertação de Mestrado.

Às empresas *Texas Instruments*, *Altera* e *Pi componentes*, por acreditarem em meu trabalho e pelas valiosas doações materiais.

Aos professores e funcionários do mestrado e do departamento de Engenharia Elétrica – DEL – UFMS por seu auxílio.

A equipe de pesquisadores do Batlab (Laboratório de Inteligência Artificial, Eletrônica de Potência e Eletrônica Digital), especialmente aos meus grandes amigos e incentivadores Gilberto S. Tatibana e Márcio L. Portella.

Resumo da Dissertação apresentada a UFMS como parte dos requisitos necessários para a obtenção do grau de Mestre em Engenharia Elétrica.

IMPLEMENTAÇÃO DIGITAL DE REDES NEURAIS ARTIFICIAIS PARA O CONTROLE DE MOTOR DE INDUÇÃO

André Muniz Soares

Dezembro/2006

Orientador: Luciana Cambraia Leite, Dra.
Co-orientador: João Onofre Pereira Pinto, Ph.D.
Área de Concentração: Energia Elétrica.
Palavras-chave: acionamento de motores de indução, controle vetorial *sensorless*, neurohardware, FPGAs.
Número de Páginas: 120.

RESUMO: Este trabalho tem como contribuição a implementação de redes neurais artificiais utilizadas no controlador "*sensorless*" do motor de indução trifásico. Estas redes neurais foram representadas como circuitos dentro de um FPGA (*Field Programmable Gate Array*), constituídas por neurônios representados por processadores digitais independentes e sintetizados em VHDL. Para a tarefa de representação da função de ativação sigmoidal dos neurônios em hardware, fez-se uso da técnica de interpolação "*spline*". Como resultado final do trabalho, foi construído um protótipo do controlador do motor de indução trifásico, composto por um FPGA gerenciado por um DSP (processador digital de sinais), controlando circuitos de potência para seu acionamento. O trabalho inicia-se com ênfase em uma revisão bibliográfica sobre estratégias de controle de motores de indução e implementação em hardware de redes neurais. A seguir, o sistema proposto é apresentado, iniciando pela apresentação de seus blocos constituintes, para finalizar com uma descrição dos circuitos do protótipo completo. Finalmente, são apresentados resultados experimentais de testes com o protótipo e conclusões finais.

Abstract of Dissertation presented to UFMS as a partial fulfillment of the requirements for the degree of Master in Electrical Engineering.

A Neural-Network-Digital Implementation for Induction Motor Drive

André Muniz Soares

December / 2006

Advisor: Luciana Cambraia Leite, Dra.
Co-advisor: João Onofre Pereira Pinto, Ph.D.
Area of Concentration: Electrical Energy.
Keywords: induction motors drives, sensorless vector control, neurohardware, FPGAs.
Number of Pages: 120.

ABSTRACT: This work describes a neural-network-based implementation used in 'sensorless' space-vector control for induction motor drives. These neural-networks were represented as circuits inside of a FPGA (Field Programmable Gate Array), employed by neurons performed by independent digital processors and synthesized in VHDL. In order to achieve the building of the sigmoidal activation function for digital implementations of neurons, it has been used the "spline" interpolation technique. As results, it was built a hardware system structure to drive an induction motor, which is composed by a FPGA managed by a DSP (digital signal processor), and so to control the power electronics. In the beginning, great emphasis is laid on description of the various strategies of induction motors control, and the neural networks hardware implementations. Next, the proposed system is presented, and it is goal to give a brief description of the built blocks. In addition, it is presented a description of the final prototype circuits. Then, the experimental parts are presented to demonstrate the performance of tests with the prototype and the work closes with conclusions.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

CA: Corrente alternada.

CC: Corrente contínua.

DSK*: DSP Starter Kit.*

DSP: *Digital Signal Processor.*

EDMA*: Extended Direct Memory Access.*

EMIF: *External Memory Interface.*

FPGA: *Field Programmable Gate Array.*

IGBT: *Insulated Gate Bipolar Transistor.*

LCD*: Liquid Crystal Display.*

MAC*:* multiplicador-acumulador.

MLP: *Multi Layer Perceptron.*

MIT: Motor de Indução Trifásico.

PCLPF: *Programmable Cascaded Low Pass Filter.*

PLL: *Phase Locked Loop.*

PWM: *Pulse Width Modulation* (Modulação por Largura de Pulso).

ROM: *Read Only Memory.*

RNA: Rede Neural Artificial.

RNN: *Recurrent Neural Network* (Rede Neural Recorrente).

SDRAM*: Synchronous Dynamic Random Access Memory.*

SVM: *Space-Vector Modulation* (Modulação por Vetores Espaciais).

UCP: Unidade Central de Processamento.

ULA: Unidade Lógica e Aritmética.

VHDL: *VHSIC Hardware Description Language.*

VHSIC: *Very High Speed Integrated Circuit*.

VLWI: *Very Large Word Instruction.*

CAPÍTULO 1 – INTRODUÇÃO

1.1 Introdução

Este capítulo visa fornecer um embasamento conceitual sobre o tema de acionamento de motores de indução. Inicialmente, é feita uma revisão bibliográfica sobre máquinas de indução e sobre suas técnicas de acionamento. A seguir, são descritas as técnicas escalar e vetorial para controle de motores e a utilização da inteligência artificial como ferramenta útil na implementação em hardware de sistemas para acionamento de máquinas CA. Finalizando, é apresentada a ordem em que estão organizados os capítulos deste trabalho, além do assunto que os aborda. A seção a seguir inicia com uma breve revisão sobre máquinas de indução e seu acionamento em velocidades variáveis.

1.2 O motor de indução e o acionamento em velocidades variáveis

Por um longo tempo, a máquina de indução foi, tradicionalmente, utilizada em aplicações de velocidades constantes e em acionamentos de baixo desempenho, devido às suas características não-lineares, multivariáveis e complexidade nos algoritmos de controle [1].

Desta forma, quando se pensava em acionamentos com velocidade variável, optava-se por utilizar a máquina de corrente contínua, cujas características de desacoplamento entre o fluxo (campo) e o conjugado (armadura) permitem um controle independente da velocidade, possibilitando um excelente desempenho nesse tipo de aplicação [2].

Todavia, as máquinas de corrente contínua têm custos elevados, necessitam de manutenções freqüentes e tornam-se inviáveis para certas aplicações. Ao contrário, a máquina de indução possui construção robusta, menor custo de fabricação e manutenção. A partir da década de 70, com o surgimento da teoria de controle vetorial (princípio de orientação de campo) [3] é que se obteve uma modelagem adequada para o controle mais preciso das máquinas de corrente alternada. Somado ao contínuo avanço da eletrônica de potência, a partir da década de 80, a máquina de indução vem

dominando o mercado de acionamentos em velocidades variáveis e alto desempenho [4].

1.3 O controle da máquina de indução: escalar e vetorial

Inicialmente, o controle da máquina de indução foi tido como escalar, em que se varia, proporcionalmente, tensão e freqüência estatórica, mantendo-se o fluxo de entreferro da máquina constante, não se alterando o torque máximo. Trata-se de um controle relativamente simples, utilizado em sistemas de acionamento em que não se necessite de grandes variações de velocidades e com rápidas respostas e alta eficiência [5].

A técnica de orientação de campo possibilita um desacoplamento entre o conjugado eletromagnético e o fluxo (campo) na máquina de corrente alternada, semelhante à máquina CC com excitação em separado, melhorando assim, suas características dinâmicas, quando comparada com a resposta dos acionamentos convencionais. Isto é conseguido através da definição de um eixo de referência ***q*-*d*** que gira sincronamente com o vetor espacial *fluxo de rotor*. Desta forma, o torque é controlado através da componente ***q*** do vetor espacial da corrente de estator e simultaneamente, o fluxo é controlado através da componente ***d*** do vetor espacial da corrente de estator.

Sua implementação só foi possível a partir da década de 80, com os avanços da eletrônica de potência, de microprocessadores e o surgimento da tecnologia de processadores digitais de sinais (DSP), já que esta técnica necessita de alta complexidade computacional.

Vários métodos de implementação têm sido desenvolvidos, e estas técnicas podem ser classificadas baseadas no modo usado para se determinar o vetor de fluxo rotórico ou estatórico, dentre as quais se destacam:

- Controle por orientação de campo direto: a posição e a magnitude do fluxo rotórico são medidas ou estimadas através de um observador de estado não linear. A desvantagem é que se deve ter um conhecimento, a priori, da constante de tempo elétrica do rotor, a qual varia com a temperatura, freqüência e saturação.
- Controle de campo orientado indireto (COI): a estimação do fluxo de referência (fluxo de rotor) é feita em malha aberta e possui como vantagens: simplicidade

do modelo obtido, boa resposta dinâmica [6, 7]. Uma desvantagem é que a estimação do fluxo depende dos parâmetros da máquina e seu uso incorreto pode causar o acoplamento das equações de torque e fluxo, degradando o desempenho do controle [8, 9].

- Controle Direto de Torque (DTC) e Auto-Controle Direto (DSC): técnicas que se baseiam no controle direto e independente do torque e do fluxo do motor CA, possibilitando rápida resposta de torque e excelente regulação de velocidade [10, 11].
- Controle Robusto e Adaptativo: são técnicas utilizadas para controlar plantas ou processos com incertezas. O Controle a Estrutura Variável (VSC) com modos deslizantes é um tipo de controle robusto, foi proposto na década de 50 e há diversas contribuições na aplicação de acionamentos e controle de máquinas de indução [12, 13, 14].

1.4 Técnicas de controle inteligente

Dentre as técnicas de inteligência artificial aplicadas ao acionamento de máquinas de indução, destacam-se: controle baseado em redes neurais artificiais, controle fuzzy, controle neuro-fuzzy e controle baseado em algoritmos genéticos.

A lógica nebulosa ou fuzzy [15, 16] proporcionou o desenvolvimento de controladores fuzzy, em que algoritmos convertem estratégias de controle baseadas no conhecimento de especialistas em estratégias de controle automático, apresentando resultados superiores àqueles com controle convencional.

A técnica de Algoritmos Genéticos também pode ser utilizada no controle de motores [17]. Dentre as aplicações destacam-se: projeto de estimadores de fluxos para se obter os parâmetros ótimos do controle vetorial de um MI [18], na sintonia de controladores fuzzy, determinação dos parâmetros elétricos de um motor de indução através de técnicas de estimação de parâmetros, etc.

Redes Neurais Artificiais também têm sido uma das ferramentas muito utilizadas na identificação e controle de sistemas dinâmicos não-lineares [19], devido a sua habilidade de aprendizagem e melhorar progressivamente a *performance* de sistemas de controle [20, 21]. A rede neural ainda pode ser combinada com a técnica de lógica fuzzy, surgindo os controladores neuro-fuzzy (ou híbridos). A melhor vantagem sobre os controladores fuzzy convencionais é que a função de pertinência e as regras são

produzidas por um processo automático [22], minimizando a intervenção humana no processo de sintonia.

Desta forma, a aplicação de inteligência artificial em acionamentos de máquinas melhora satisfatoriamente a performance, tornando o sistema mais robusto à variações dos parâmetros e da carga.

Quando se trata de controle *on-line* de máquinas de indução para altos desempenhos, a informação da velocidade é necessária e isto pode ser conseguido através de sua medição direta (*encoder*) ou sua estimação através da medição de tensão e corrente (*sensorless*). Muitos trabalhos foram desenvolvidos com a aplicação da técnica de redes neurais para estimar velocidade, fluxos ou corrente estatórica na máquina de indução [23, 24].

Atualmente, com o desenvolvimento dos DSP's, as técnicas de controle inteligente de motores de indução (especialmente as redes neurais) podem ser implementadas como sistemas de tempo real. Porém, para implementá-las em hardware, neurônios com altas capacidades de processamento são necessários. A seção a seguir apresenta maiores detalhes sobre o tema da implementação de redes neurais em hardware.

1.5 Implementação de redes neurais artificiais (RNA) em hardware

Um dos problemas na implementação de redes neurais em hardware para o controle e acionamento de MI está na complexidade de se representar eletronicamente funções de ativação não-lineares. Muitos modelos de redes neurais foram desenvolvidos com neurônios binários, ou seja, neurônios com entradas e saídas que podem assumir apenas dois valores (**-1** e **1** ou **0** e **1**).

De fato, em muitas implementações utilizam-se funções de ativação *hard-limit* ou funções lineares saturadas, com a finalidade de evitar a complexidade envolvida na construção de uma função de ativação sigmoidal. O problema desta abordagem está no modelo distanciar-se demasiadamente da complexidade do neurônio biológico [25]. Estas e outras desvantagens encontradas nos primeiros modelos binários foram superadas pelo neurônio analógico. Neurônios implementados com circuitos analógicos são rápidos e possuem reduzida complexidade, permitindo a construção de redes neurais com grande densidade de neurônios e baixo tempo de resposta. Entretanto, a implementação de neurônios analógicos possui algumas desvantagens, tais como: baixa

imunidade a ruídos elétricos e a necessidade do uso de resistores de alta precisão para representação dos pesos sinápticos [26, 27].

Devido a tais problemas, pesquisas voltadas a implementação de redes *multi-layer perceptron* (MLP's) para emulação do comportamento do neurônio biológico têm atraído a atenção da comunidade científica [28, 29, 30]. Alguns autores implementaram uma rede MLP com funções de ativação lineares [31], outros [32] implementaram uma rede neural para identificar, em tempo real, a velocidade do motor, também utilizando funções lineares para modelar neurônios digitais.

Este trabalho tem como contribuição a implementação dos neurônios artificiais como processadores digitais independentes, utilizando funções de ativação sigmoidais representadas em hardware por interpolação. Para permitir reduzidos erros de representação numérica nos cálculos, os neurônios trabalham com valores em ponto flutuante de 32 bits (IEEE 754). A estrutura e o conteúdo dos capítulos deste trabalho são apresentados na seção a seguir.

1.6 Descrição dos capítulos

Esta dissertação possui seis capítulos e as descrições de cada um deles segue abaixo:

- Capítulo 2: apresenta o equacionamento da máquina de indução com orientação de campo de estator, bem como a visão geral do sistema de acionamento. O capítulo apresenta os dois subsistemas compondo o controlador do motor deste trabalho: um subsistema de potência e um de processamento de sinais.
- Capítulo 3: detalha a estrutura interna dos subsistemas "Estimador de Fluxo" e "Controle PWM Space-Vector Baseado em Redes Neurais", descritos de forma geral no capítulo 2. Estes subsistemas representam duas redes neurais artificiais modeladas nos trabalhos [25] e [26] e que serviram de base para a implementação proposta neste trabalho.
- Capítulo 4: detalha arquitetura de hardware do protótipo do controlador do motor de indução proposto neste trabalho, ou seja, como são os circuitos representando todos os subsistemas modelados matematicamente conforme apresentado nos capítulos 2 e 3. O capítulo utiliza linguagem de descrição de hardware (VHDL) e de programação de alto nível (C) para explicar como são e como funcionam as plataformas de

implementação utilizadas na construção do protótipo, ou seja: um DSP (processador digital de sinais) e um FPGA (*Field Programmable Gate Array*).

- Capítulo 5: faz a apresentação do protótipo final e de resultados experimentais obtidos. O capítulo inclui fotos das placas de circuito impresso do protótipo, além de formas de onda obtidas em testes de funcionamento realizados com o protótipo.
- Capítulo 6: destina-se à apresentação das conclusões finais e de propostas para futuros trabalhos.

1.7 Considerações finais

Este capítulo fez uma breve revisão bibliográfica sobre acionamento de motores de indução, implementação em hardware de redes neurais artificiais e sobre a ordem como estão apresentados os capítulos deste trabalho. O objetivo principal foi situar o leitor sobre alguns desafios e perspectivas existentes na área de controle inteligente de motores de indução. Ao final do trabalho, espera-se que o leitor adquira uma visão global dos conceitos e das soluções utilizadas na implementação do controlador do motor proposto.

CAPÍTULO 2 - CONTROLE DE MOTOR DE INDUÇÃO COM ORIENTAÇÃO DO CAMPO ESTATÓRICO

2.1 Introdução

A proposta deste trabalho é a implementação em hardware de um controlador do motor de indução trifásico, utilizando como base uma modelagem de sistema idealizada em trabalhos anteriores. Este capítulo apresenta esta modelagem a partir da qual foi criada uma arquitetura de hardware, descrita em maiores detalhes no capítulo 4. Inicialmente é apresentada uma visão geral dos blocos compondo a estrutura do sistema, e então, cada bloco é apresentado separadamente O objetivo do capítulo é a apresentação do sistema discutindo somente a função e o equacionamento dos seus blocos constituintes.

A seção a seguir inicia o capítulo com uma descrição estrutural geral do controlador de motor de indução proposto.

2.2 Visão geral do sistema

Conforme descrito no capítulo 1, existem várias abordagens retratadas na literatura para controle inteligente de motores de indução. O controlador de motor proposto neste trabalho entra na categoria de controle inteligente de máquinas de indução, por ser baseado no uso de redes neurais artificiais, uma das áreas de estudo da Inteligência Artificial.

Iniciando pela apresentação geral do sistema de controle de motor de indução, a figura 2.1 mostra um diagrama geral do sistema previamente modelado nos trabalhos anteriores [25] e [26]. Este sistema representa um controle em malha fechada "*sensorless*". Isto significa um controle de motor baseado na estimação de grandezas da máquina controlada e não no sensoriamento ou medição direta destas grandezas. Uma das principais grandezas a ser estimada pelo sistema é o fluxo estatórico do motor. A partir desta grandeza, podem-se estimar vários outros parâmetros úteis em um controle inteligente do motor de indução.

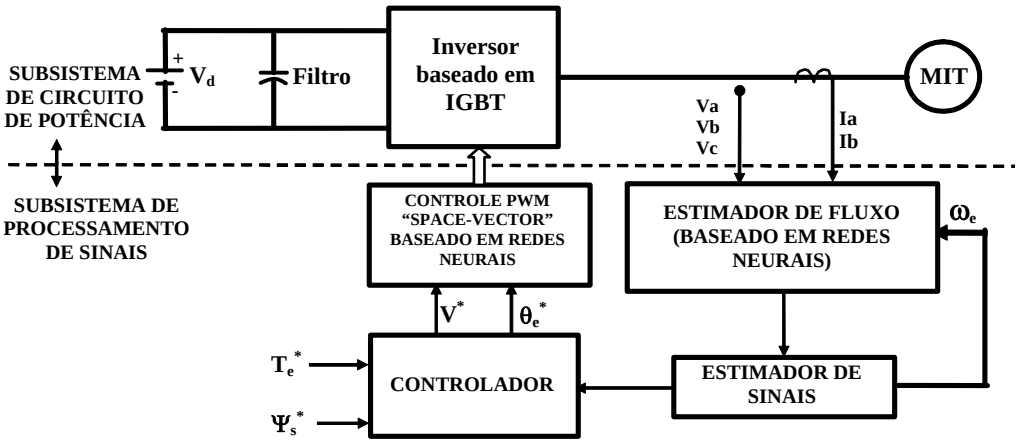


Figura 2. 1 - Diagrama do controlador proposto.

De acordo com a figura 2.1, o sistema é constituído de dois subsistemas, um "Subsistema de circuito de potência" (acima da linha tracejada), e um "Subsistema de processamento de sinais" (abaixo da linha tracejada).

O propósito do "Subsistema de circuito de potência" é transformar energia elétrica armazenada no "*Link* CC" (fonte "V_d" mais "Filtro") em energia mecânica rotacional, entregue ao motor de indução trifásico (MIT). Esta transformação está centrada no trabalho do módulo localizado entre o "*Link* CC" e o "MIT", ou seja, o "Inversor baseado em IGBT". Este módulo representa um conversor CC/CA de potência, trifásico, e controlado por sinais PWM (Modulação por Largura de Pulso) providos pelo "Sistema de processamento de sinais".

O "Subsistema de processamento de sinais" representa o foco deste trabalho. Ele representa a parte digital do controlador do motor de indução trifásico, sendo baseado em redes neurais artificiais. É composto pelos módulos "Controlador", "Controle PWM Space-Vector Baseado em Redes Neurais", "Estimador de Fluxo" e pelo "Estimador de Sinais".

Aprofundando um pouco mais a análise sobre cada um dos dois subsistemas apresentados nesta visão geral, a seção a seguir inicia descrevendo aspectos do "Subsistema de circuito de potência".

2.3 Subsistema de circuito de potência

Embora não seja o foco deste trabalho, é necessária uma descrição das principais características dos elementos compondo o "Subsistema de circuito de potência". Este

subsistema representa a parte de potência do controlador proposto neste trabalho, estando sob controle do "Subsistema de processamento de sinais". A seção a seguir inicia apresentando a estrutura e o funcionamento do "Motor de indução trifásico".

2.3.1 Motor de indução trifásico

O elemento eletro-mecânico do sistema, transformador de potência elétrica em potência mecânica, é o motor de indução trifásico (MIT). Um MIT é equivalente a um transformador trifásico, porém com enrolamento secundário curto-circuitado e livre para realizar movimento rotacional. Um diagrama construtivo do motor de indução é mostrado na figura 2.2.

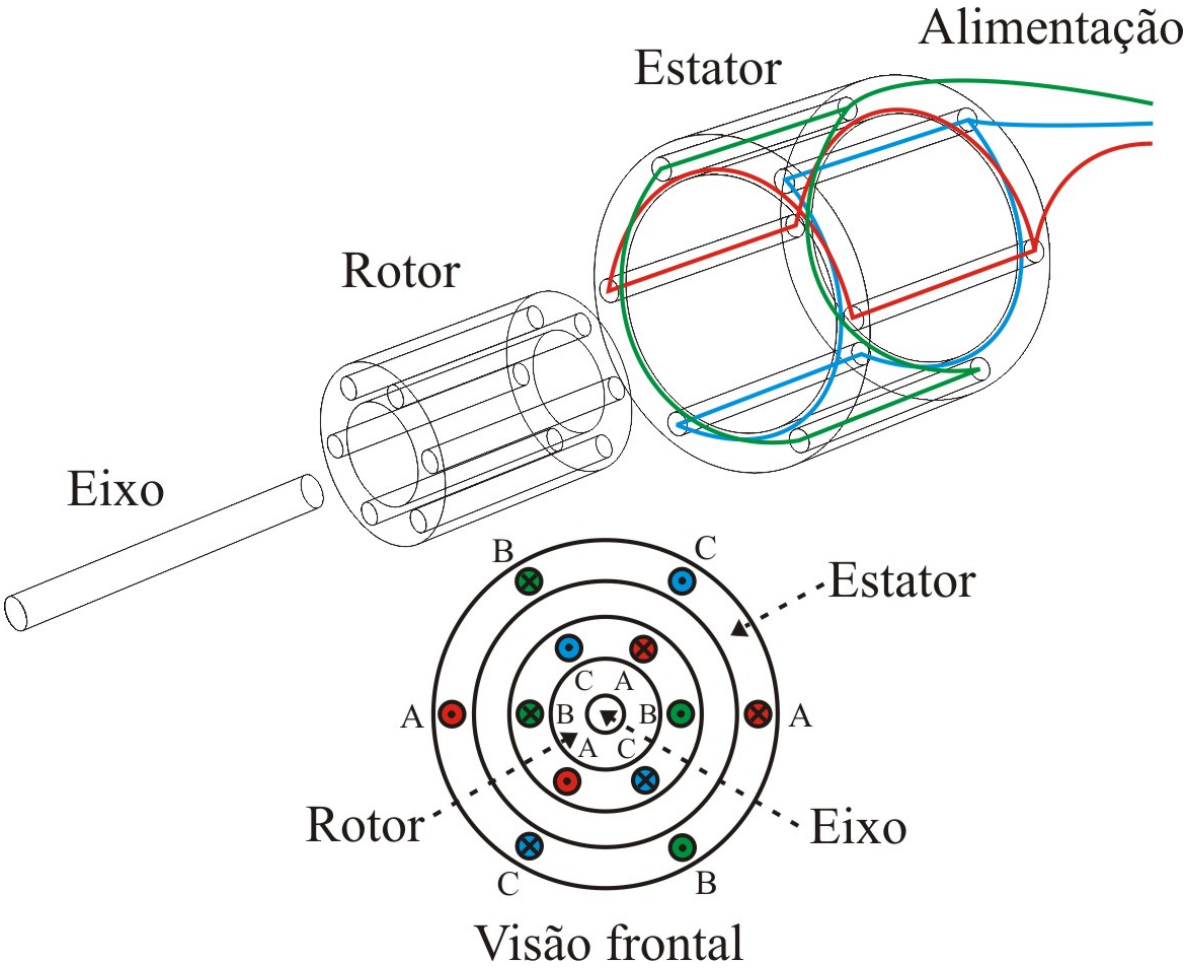


Figura 2. 2 - Aspecto físico de um motor de indução.

A figura 2.2 apresenta uma visão espacial e uma frontal dos elementos básicos que constituem um motor de indução: estator, rotor e o eixo.

Em caso de alimentação trifásica, o estator (ou armadura) é envolvido por três enrolamentos formando dipolos elétricos (bobinas) defasados espacialmente de 120 graus. Na visão frontal da figura 2.2, o estator representa o anel mais externo e os enrolamentos das bobinas representam os círculos coloridos no anel (podendo estar entrando (×) ou saindo (•) do plano da página). O controle das formas de onda de alimentação (tensão e corrente) das bobinas no estator constitui o objetivo final de um controlador do motor de indução. Em muitas aplicações práticas, a alimentação aplicada ao estator consiste de ondas senoidais providas por uma rede trifásica (partida direta), dispensando o uso de um sistema controlador para o motor.

O rotor representa o elemento girante do motor. Na figura 2.2 ele está representado pelo conjunto de barras presas por anéis em suas extremidades, formando o que é conhecido como "gaiola de esquilo". Em alguns tipos de rotores, a estrutura em forma de gaiola é substituída por enrolamentos semelhantes aos do estator, porém, não acessíveis fora do motor. Na visão frontal da figura 2.2, o rotor representa o anel central e os círculos coloridos neste anel representam as barras da gaiola do rotor (sendo percorridas por correntes entrando (×) e saindo (•) do plano da página).

Durante o funcionamento do motor, o rotor gira devido ao torque (conjugado) produzido por uma força magnética aplicada ao rotor. As formas de onda aplicadas às bobinas do estator geram um campo magnético resultante girando em torno do eixo central do motor. A variação de fluxo desse campo magnético induz uma tensão no rotor, esta tensão então acaba gerando correntes rotóricas. Estas correntes, sob a ação do campo magnético do estator, originam a força magnética geradora do torque no rotor. Como conseqüência, o rotor gira tentando alcançar o campo girante, de modo a diminuir a corrente induzida no rotor (resultado da lei de "*Lenz*").

Preso ao rotor está o eixo do motor, mostrado como o círculo central na visão frontal da figura 2.2. O eixo simplesmente representa a haste de transmissão da energia mecânica do rotor para cargas externas conectadas ao motor.

Prosseguindo com a apresentação dos circuitos de potência do controlador de MIT deste trabalho, a seção a seguir apresenta o segundo elemento de destaque compondo o "Subsistema de circuito de potência", o módulo "Inversor Baseado em IGBT".

2.3.2 Inversor Baseado em IGBT

Conforme já mencionado na seção anterior, para que se possa controlar um motor de indução deve-se aplicar em seus terminais de entrada uma forma de onda de alimentação controlada. O módulo “Inversor Baseado em IGBT” representa um circuito de potência para o controle da amplitude e da freqüência das tensões de alimentação aplicadas ao motor. Amplitude e freqüência de tensões CA podem ser variadas utilizando diferentes tipos de conversores. O “Inversor Baseado em IGBT” representa um conversor alimentado por fonte de tensão, chaveado e em ponte completa trifásica. Para um melhor entendimento, a figura 2.3 apresenta um diagrama do circuito do inversor utilizado neste trabalho.

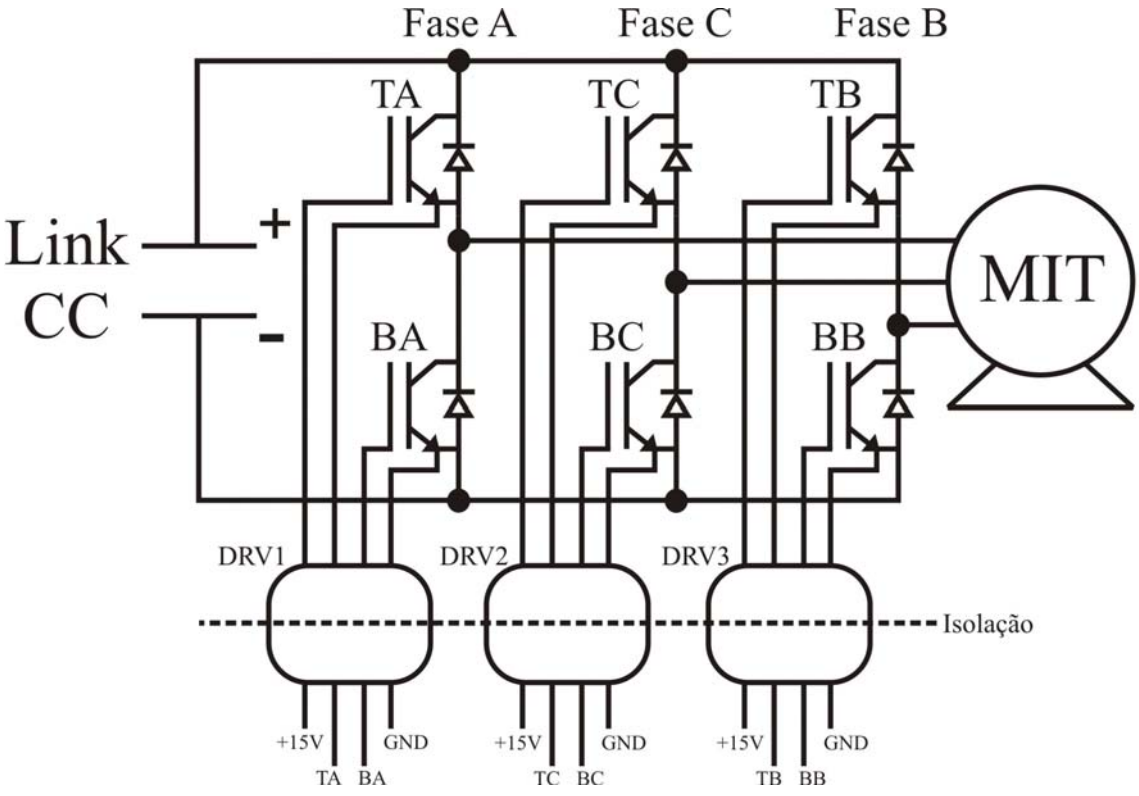


Figura 2. 3 - Módulo “Inversor Baseado em IGBT”.

O circuito da figura 2.3 gera uma tensão alternada de saída através do chaveamento de uma tensão CC (*Link* CC), utilizando, para isto, um circuito composto por três pares de chaves ("pernas" do inversor) interligadas, formando uma ponte completa. Neste caso, as chaves utilizadas são módulos IGBT (Transistor Bipolar de Gate Isolado). Na figura 2.3, os IGBTs estão representados pelas siglas indicando suas posições na ponte: “TA” (topo fase A), “BA” (baixo fase A), “TB” (topo fase B), “BB”

(baixo fase B), “TC” (topo fase C), “BC” (baixo fase C). Por se tratarem de dispositivos de potência, os IGBTs são controlados por meio de circuitos de disparo de comando de pulso de gate (“DRV1”, “DRV2”, “DRV3”, na figura 2.3). Este circuito de disparo representa uma interface de hardware com isolação permitindo que circuitos de controle de baixa potência consigam comandar os IGBTs da ponte. Adicionalmente, este circuito de disparo é capaz de filtrar ruídos dos sinais de chaveamento enviados aos IGBTs e é capaz de gerar um “tempo morto” sempre que duas chaves de uma mesma “perna” do inversor são comutadas ao mesmo tempo, evitando um curto-circuito momentâneo nesta “perna”. De acordo com a figura 2.3, cada circuito de disparo deve receber uma alimentação externa de +15V CC e deve receber dois sinais de comando: um para chave de topo (como “TA” para o “DRV1”) e outro para chave de baixo (como “BA” para o “DRV1”).

Um dos objetivos finais do controle do inversor da figura 2.3, é a construção de tensões alternadas para alimentação do motor, com o mínimo de conteúdo harmônico de baixa freqüência, já que altas freqüências podem ser eliminadas mais facilmente através de filtros apropriados. A aplicação de uma seqüência adequada de comutação das chaves da ponte inversora, a cada período de amostragem “T_s” (inverso da freqüência de chaveamento), gera três ondas defasadas de 120 graus, compostas por pulsos, simulando ondas CA senoidais, para alimentação do motor de indução. A amplitude e a freqüência destas pseudo-senóides são responsabilidades do controlador externo comandando o inversor, enquanto a técnica utilizada para comutação das chaves da ponte é conhecida como modulação. Neste trabalho a técnica de modulação utilizada é a “Modulação por Vetores Espaciais” (ou “*Space-Vector Modulation*”), implementada pelos circuitos digitais do “Subsistema de Processamento de Sinais”, descrito na seção a seguir.

2.4 Subsistema de Processamento de Sinais

Representando o “cérebro” do controlador de motor de indução deste trabalho, existe o “Subsistema de Processamento de Sinais”. A função deste subsistema é comandar o “Subsistema de circuito de potência” a partir do processamento de informações fornecidas pelo usuário e de informações medidas do motor de indução controlado. No protótipo final proposto neste trabalho, o “Subsistema de Processamento de Sinais” é constituído por um DSP (Processador Digital de Sinais) em interação direta com um FPGA (*Field Programmable Gate Array*).

O funcionamento do “Subsistema de processamento de sinais” é baseado na execução contínua de ciclos de trabalho (iterações). Cada iteração começa com a medição de tensões e correntes trifásicas dos terminais de entrada do MIT. A seguir estes dados sofrem transformações de eixo e são submetidos a uma seqüência de subsistemas composta pelo "Estimador de Fluxo", "Estimador de Sinais", "Controlador", e "Controle PWM Space-Vector Baseado em Redes Neurais". O resultado final de cada iteração é a atualização de parâmetros de temporização de sinais de comando aplicados ao módulo “Inversor de Baseado em IGBT” (Subsistema de circuito de potência). No decorrer desta seção, uma iteração de trabalho do “Subsistema de Processamento de Sinais” é descrita através da apresentação das funções executadas pelos seus blocos constituintes. Inicialmente, na seção a seguir, é descrito o bloco “Estimador de Fluxo”.

2.4.1 Bloco Estimador de Fluxo

Durante um ciclo de trabalho, o primeiro bloco a trabalhar é o "Estimador de Fluxo". Sua finalidade é estimar o fluxo de estator atual da máquina de indução usando medições de tensão e corrente dos terminais de entrada da MIT.

Este bloco recebe como entradas as tensões das fases A, B e C (V_a, V_b, V_c), e também as correntes das fases A e B (i_a, i_b). Em um sistema real, estes sinais são primeiramente condicionados por filtros do tipo passa-baixa para eliminação de ruídos, com correção dos atrasos indesejáveis, antes de serem aplicados às entradas do bloco "Estimador de Fluxo". Dentro do bloco, o primeiro processamento realizado sobre estes sinais é uma transformação dos eixos A-B-C para o eixo estacionário d-q-0, conforme indicado pelas equações 2.1 a 2.3.

$$i_c = -i_a - i_b \tag{2.1}$$

$$\begin{bmatrix} V_{qs}^{s'} \\ V_{ds}^{s'} \end{bmatrix} = \begin{bmatrix} 2/3 & -1/3 & -1/3 \\ 0 & -1/\sqrt{3} & 1/\sqrt{3} \end{bmatrix} \cdot \begin{bmatrix} V_a \\ V_b \\ V_c \end{bmatrix} \tag{2.2}$$

$$\begin{bmatrix} i_{qs}^{s'} \\ i_{ds}^{s'} \end{bmatrix} = \begin{bmatrix} 2/3 & -1/3 & -1/3 \\ 0 & -1/\sqrt{3} & 1/\sqrt{3} \end{bmatrix} \cdot \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} \tag{2.3}$$

Na equação 2.1 está representada a estimação de corrente da fase C através de uma redundância analítica usando as correntes das fases A e B. Na equação 2.2, as tensões das três fases são convertidas nas grandezas: $V_{qs}^{s'}$ (projeção sobre eixo q) e $V_{ds}^{s'}$ (projeção sobre o eixo d). Na equação 2.3 ocorre o mesmo tipo de transformação, porém as correntes das três fases sofrem transformações para as componentes $i_{qs}^{s'}$ (projeção sobre eixo q) e $i_{ds}^{s'}$ (projeção sobre eixo d).

As tensões e correntes projetadas para o eixo d-q-0, geradas pelas equações 2.1 a 2.3, são então utilizadas nas equações 2.4 e 2.5.

$$V_{qs}^{s''} = V_{qs}^{s'} - i_{qs}^{s'} \cdot R_s \qquad (2.4)$$

$$V_{ds}^{s''} = V_{ds}^{s'} - i_{ds}^{s'} \cdot R_s \qquad (2.5)$$

As equações 2.4 e 2.5 representam as tensões de estator da MIT no eixo d-q-0 ($V_{qs}^{s''}$ e $V_{ds}^{s''}$), ao serem consideradas as quedas de tensão $i_{qs}^{s'} \cdot R_s$ e $i_{ds}^{s'} \cdot R_s$, devidas à resistência de estator R_s.

As principais saídas do bloco "Estimador de fluxo" são as estimações do fluxo de estator nos eixos d e q, obtidas matematicamente por meio das integrações representadas pelas equações 2.6 e 2.7.

$$\Psi_{ds}^{s} = \int V_{ds}^{s''} dt \qquad (2.6)$$

$$\Psi_{qs}^{s} = \int V_{qs}^{s''} dt \qquad (2.7)$$

No sistema proposto neste trabalho as integrações das equações 2.6 e 2.7 são realizadas utilizando redes neurais agindo como um filtro passa-baixa programável. Mais especificamente, utiliza-se uma RNN-MLP (rede neural recorrente híbrida com rede *Multi Layer Perceptron*) a ser detalhadamente descrita no próximo capítulo. Esta rede além das entradas $V_{qs}^{s''}$ e $V_{ds}^{s''}$ utiliza a entrada ω_e (freqüência), calculada pelo bloco "Estimador de Sinais" (descrito na seção seguinte). A estrutura da rede RNN-MLP utilizada realiza integrações sem erro de "*offset*", além de efetuar compensações de deslocamentos de fase gerados pela filtragem analógica dos sinais de tensões e correntes trifásicas (entradas do bloco "Estimador de Fluxo").

Como saída final, além dos valores de fluxo estimado (ψ_{qs}^{s} e ψ_{ds}^{s}), o bloco "Estimador de Fluxo" disponibiliza para o próximo bloco no sistema (o "Estimador de Sinais") as correntes $i_{qs}^{s'}$ e $i_{ds}^{s'}$, necessárias neste último bloco.

No protótipo final proposto neste trabalho, o bloco "Estimador de Fluxo" foi inserido dentro do FPGA como uma estrutura interligada de processadores, constituindo uma rede neural. Maiores detalhes sobre como este bloco foi implementado em hardware digital são fornecidos no capítulo 4. Resultados experimentais e simulações do funcionamento deste bloco são apresentados no capítulo 5.

2.4.2 Bloco Estimador de Sinais

Este bloco realiza os cálculos com os sinais produzidos pelo bloco "Estimador de Fluxo", listados nas equações 2.8 a 2.16.

$$\hat{\psi}_s = \sqrt{(\psi_{ds}^{s})^2 + (\psi_{qs}^{s})^2} \qquad (2.8)$$

$$\Theta_e = \sin^{-1}(\psi_{qs}^{s} / \hat{\psi}_s) \qquad (2.9)$$

$$i_{ds} = i_{qs}^{s'} \cdot \cos\Theta_e - i_{ds}^{s'} \cdot \sin\Theta e \qquad (2.10)$$

$$i_{qs} = i_{qs}^{s'} \cdot \sin\Theta_e + i_{ds}^{s'} \cdot \cos\Theta e \qquad (2.11)$$

$$i_{dq} = \left(\frac{\sigma \cdot L_s \cdot i_{qs}^{2}}{\psi_{ds} - \sigma \cdot L_s \cdot i_{ds}} \right) \qquad (2.12)$$

$$\psi_{ds} = \psi_{qs}^{s} \cdot \cos\Theta_e - \psi_{ds}^{s} \cdot \sin\Theta e \qquad (2.13)$$

$$\psi_{qs} = \psi_{qs}^{s} \cdot \sin\Theta_e + \psi_{ds}^{s} \cdot \cos\Theta e \qquad (2.14)$$

$$T_e = \frac{3 \cdot P}{4} \cdot \left[\psi_{ds} \cdot i_{qs} - \psi_{qs} \cdot i_{ds} \right] \qquad (2.15)$$

$$\omega_e = \frac{\left[V_{qs}^{s''} \cdot \psi_{ds}^{s} - V_{ds}^{s''} \cdot \psi_{qs}^{s} \right]}{\hat{\psi}_s^{2}} \qquad (2.16)$$

A equação 2.8 representa a obtenção do fluxo de estator ($\hat{\psi}_s$) através do cálculo do módulo do vetor representado pelas coordenadas de eixo estacionário (ψ_{qs}^{s} e ψ_{ds}^{s}, calculados nas equações 2.6 e 2.7). Na equação 2.9 é obtido o ângulo de rotação estimado Θ_e através do arco-seno do valor de fluxo de eixo estacionário q (ψ_{qs}^{s}) dividido pelo fluxo calculado em 2.8. As equações 2.10 e 2.11 estimam as correntes em eixo girante (i_{qs} e i_{ds}) a partir do ângulo de rotação obtido na equação 2.9 e dos valores de corrente ($i_{qs}^{s'}$ e $i_{ds}^{s'}$) já calculados nos primeiros estágios do bloco anterior ("Estimador

de Fluxo"). As equações 2.13 e 2.14 também utilizam o ângulo Θ_e, porém juntamente com os fluxos de estator (ψ_{qs}^{s} e ψ_{ds}^{s}), obtêm-se valores de fluxo de estator no eixo girante (ψ_{qs} e ψ_{ds}). A equação 2.12 computa o valor da corrente de compensação de desacoplamento (i_{dq}) utilizando o fluxo obtido na equação 2.13 (ψ_{ds}), as correntes obtidas nas equações 2.10 e 2.11 (i_{qs} e i_{ds}), e utilizando os parâmetros do MIT predefinidos: $L_S^{'} = \sigma \cdot L_S$ (indutância transitória de estator). Na equação 2.15 é calculado o torque (conjugado) da máquina considerando o número de pólos da máquina (P), além dos fluxos ψ_{ds} e ψ_{qs} (calculados nas equações 2.13 e 2.14) e das correntes obtidas nas equações 2.10 e 2.11 (i_{qs} e i_{ds}). Por fim, na equação 2.16, é calculada a freqüência de rotação do motor (ω_e), a partir do quadrado do módulo do fluxo de estator ($\hat{\psi}_s$), dos fluxos (ψ_{qs}^{s} e ψ_{ds}^{s}), e das tensões transformadas ($V_{qs}^{s''}$ e $V_{ds}^{s''}$) calculadas conforme as equações 2.4 e 2.5 pelo bloco anterior (o "Estimador de Fluxo"). Este sinal de freqüência (ω_e) volta para o bloco "Estimador de Fluxo", como já mencionado, para atualização dos valores das grandezas estimadas por este bloco.

Como pode ser observado, é extensa a carga de cálculos a ser realizada pelo "Estimador de Sinais". Várias são as formas de implementação em hardware deste bloco. Maiores detalhes referentes à implementação do bloco "Estimador de Sinais" neste trabalho são apresentados no capítulo 4. As saídas deste bloco são direcionadas para o bloco "Controlador", descrito em maiores detalhes na seção a seguir.

2.4.3 Bloco Controlador

O bloco "Controlador" representa um ponto de convergência do sistema de controle de motor de indução. Este bloco recebe tanto sinais de retroação providos pelo bloco "Estimador de Sinais" (T_e, $\hat{\psi}_s$, i_{dq}, i_{qs} e i_{ds}), quanto o sinal de torque desejado (T_e^*) e de fluxo desejado ($\hat{\psi}_s^*$) fornecidos pelo usuário do sistema. Estes sinais estão representados na figura 2.4 compondo o subsistema "Laços de Controle de Torque e Fluxo" pertencente ao bloco "Controlador".

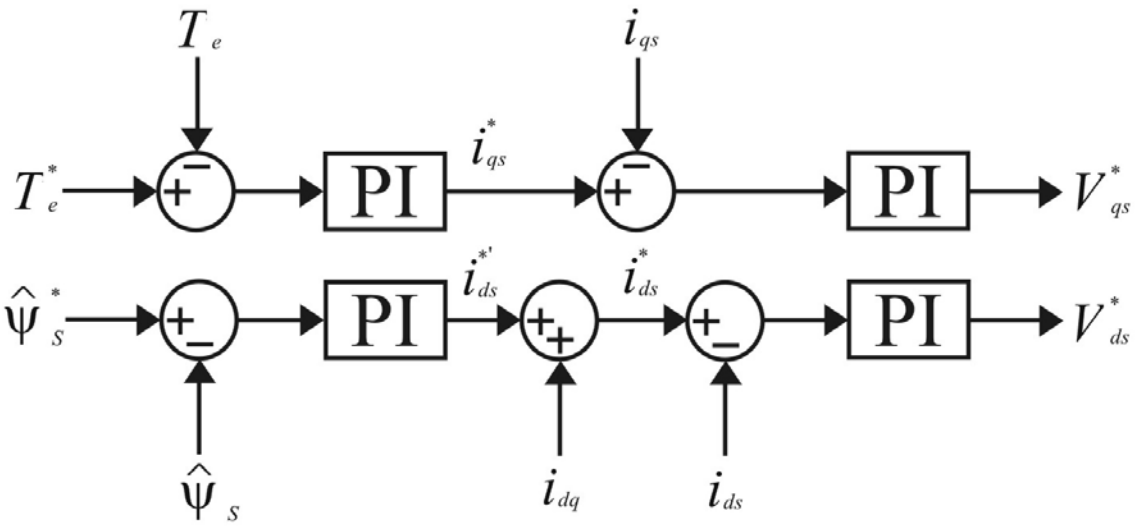


Figura 2. 4 - Diagrama do subsistema "Laços de Controle de Torque e Fluxo".

O subsistema da figura 2.4 é composto por um laço de controle de torque que inicialmente gera um valor de erro entre o torque estimado (T_e) e o torque desejado (T_e^*). Este sinal de erro entra em um controlador “PI” (proporcional e integral), o qual gera um valor de corrente de estator desejado (i_{qs}^*). A diferença entre a corrente desejada (i_{qs}^*) e a estimada (i_{qs}) representa a entrada de um segundo controlador “PI”. A saída deste segundo “PI” representa a tensão desejada de estator no eixo girante q (V_{qs}^*).

O segundo laço compondo o subsistema da figura 2.4 inicia gerando um sinal de erro entre o fluxo desejado pelo usuário ($\hat{\psi}_s^*$) e o fluxo estimado pelo sistema ($\hat{\psi}_s$). De forma análoga ao primeiro laço, um controlador “PI” gera uma corrente desejada ($i_{qs}^{*'}$) de primeiro estágio. Esta corrente, no estágio seguinte, é adicionada com i_{dq} (corrente de compensação de desacoplamento estimada) gerando a corrente desejada (i_{ds}^*). A diferença entre esta corrente (i_{ds}^*) e a estimada pelo sistema (i_{ds}) comanda o segundo “PI” do laço na geração da tensão de estator desejada no eixo girante d (V_{ds}^*).

Como operação final realizada pelo bloco "Controlador", outro de seus subsistemas, chamado "*Vector Rotator*" (figura 2.5), processa as saídas V_{qs}^* e V_{ds}^* geradas pelo subsistema "Laços de Controle de Torque e Fluxo".

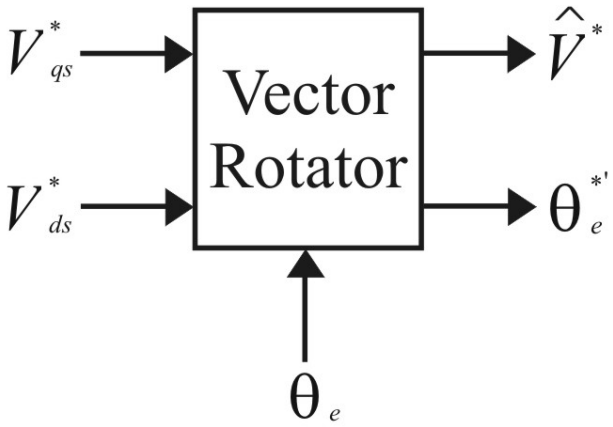


Figura 2. 5 - Subsistema "*Vector Rotator*".

Este processamento realizado pelo subsistema "*Vector Rotator*" é descrito pelas equações 2.17 e 2.18, consistindo no cálculo de representação do vetor magnitude de tensão ($\hat{V}^*$) e no cálculo do ângulo de orientação deste vetor ($\theta_e^{*'}$).

$$\hat{V}^* = \sqrt{(V_{qs}^*)^2 + (V_{ds}^*)^2} \qquad (2.17)$$

$$\theta_e^{*'} = \theta_e + \tan^{-1}\left(\frac{V_{qs}^*}{V_{ds}^*}\right) \qquad (2.18)$$

No protótipo composto por DSP mais FPGA deste trabalho, o bloco "Controlador", como o bloco "Estimador de Sinais" já descrito, está construído como uma rotina de programação para execução pelo DSP. Um usuário externo interage com o bloco "Controlador" através de uma interface com usuário interligada ao DSP. Maiores detalhes sobre os circuitos constituintes desta interface e sobre o funcionamento do bloco "Controlador" construído, são fornecido no capítulo 4.

As saídas do subsistema "*Vector Rotator*" ($\hat{V}^*$ e $\theta_e^{*'}$) são aplicadas às entradas do bloco "Controle PWM Space-Vector Baseado em Redes Neurais", última unidade da seqüência de blocos constituindo o "Subsistema de processamento de sinais". Na seção a seguir são apresentados maiores detalhes a respeito do bloco "Controle PWM Space-Vector Baseado em Redes Neurais".

2.4.4 Bloco Controle PWM Space-Vector Baseado em Redes Neurais

Este bloco recebe os sinais calculados pelo bloco "Controlador", ou seja, $\hat{V}^*$ e $\theta_e^{*'}$, e os traduz em comandos de ativação (*turn-on*) e desativação (*turn-off*) (pulsos PWM) enviados para as chaves eletrônicas presentes no "Inversor baseado em IGBT"

(Subsistema de circuito de potência). O trabalho deste bloco consiste em propiciar uma modulação, a "*Space-Vector Modulation* ou SVM". A SVM, descrita em [10], representa um algoritmo com a finalidade de criar uma forma de onda desejada a partir de uma fonte de tensão constante "V_d" (ou Link CC). Neste caso esta forma de onda desejada consiste em três senóides defasadas de 120 graus, com amplitudes e freqüências controladas, aplicadas aos terminais de entrada de um motor de indução. Para realizar seu trabalho o bloco "Controle PWM Space-Vector Baseado em Redes Neurais" conta com a estrutura representada na figura 2.6.

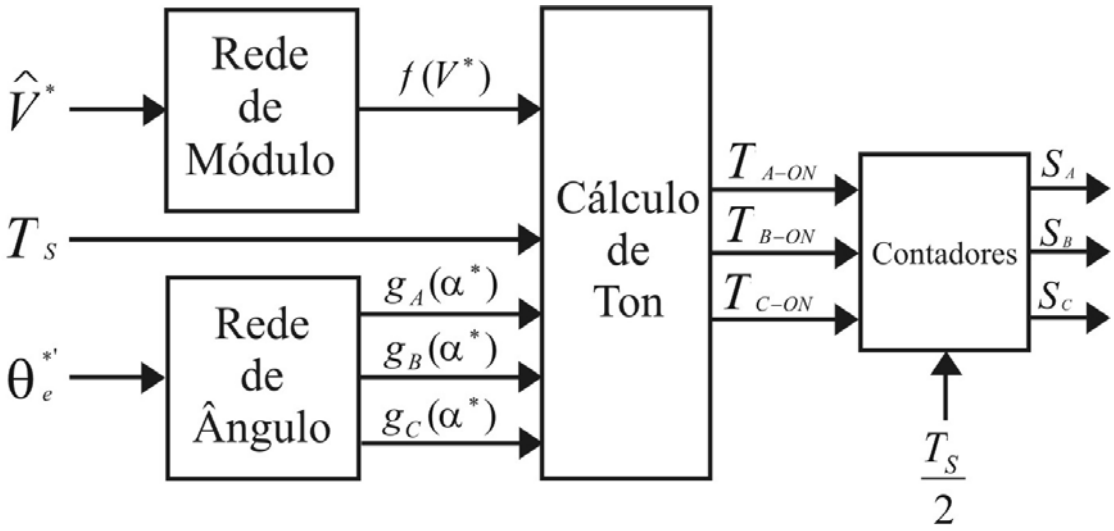


Figura 2. 6 - Diagrama do Controle PWM Space-Vector Baseado em Redes Neurais.

O diagrama da figura 2.6 mostra um subsistema que inicia processando as entradas $\hat{V}^*$ e $\theta_e^{*'}$ através de duas redes neurais: a de "Modulo" e a de "Ângulo", respectivamente. Maiores detalhes sobre a estrutura interna destas redes são fornecidos no capítulo 3. A "Rede de Módulo" realiza o cálculo da função $f(V^*)$, enquanto a "Rede de Ângulo" realiza os cálculos da função $g_A(\alpha^*)$ para a fase A, $g_B(\alpha^*)$ para a fase B e $g_C(\alpha^*)$ para a fase C. As saídas $f(V^*)$ e $g(\alpha^*)$, dentro do sub-bloco "Cálculo de Ton", são relacionadas com o período de amostragem dos sinais PWM (T_s) conforme as equações 2.19 a 2.21.

$$T_{A-ON} = f(V^*) \cdot g_A(\alpha^*) + T_s/4 \qquad (2.19)$$

$$T_{B-ON} = f(V^*) \cdot g_B(\alpha^*) + T_s/4 \qquad (2.20)$$

$$T_{C-ON} = f(V^*) \cdot g_C(\alpha^*) + T_s/4 \qquad (2.21)$$

Os três tempos de "*turn-on*" ($T_{A-ON}, T_{B-ON}, T_{C-ON}$) são utilizados como entradas para o sub-bloco "Contadores" na figura 2.6. Neste sub-bloco está um contador do tipo crescente/decrescente, responsável por gerar uma forma de onda triangular de período "T_S". Comparando-se esta onda triangular com os tempos de "*turn-on*" obtêm-se os sinais PWM de saída, conforme ilustrado na figura 2.7.

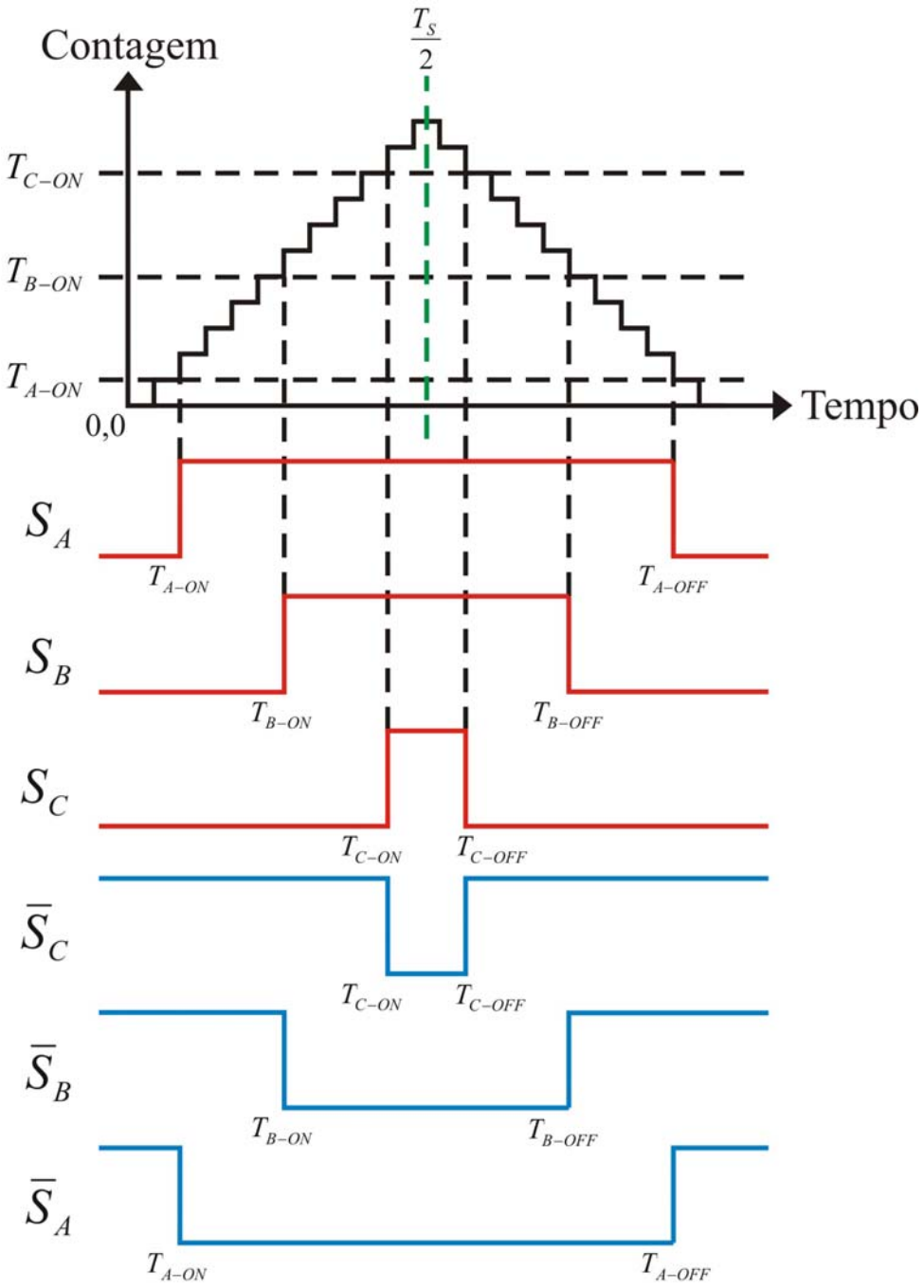


Figura 2. 7 - Exemplo de geração de sinais PWM.

A figura 2.7 apresenta um exemplo de geração de onda triangular de referência simétrica e com instante de pico localizado em "Ts/2". Os tempos "T_{A-ON}", "T_{B-ON}", "T_{C-ON}" são representados na figura por linhas tracejadas horizontais. As saídas PWM "S_A","S_B","S_C" estão representadas como ondas em vermelho, enquanto suas opostas ("$\overline{S}_A$","$\overline{S}_B$","$\overline{S}_C$") são representas por ondas em cor azul. Conforme pode ser notado da figura 2.7, a geração de um pulso em uma saída ocorre somente quando o valor de contagem (na triangular) for maior que a linha horizontal representando o tempo de "*turn-on*" relacionado com a saída. Variações no comprimento dos pulsos são obtidas deslocando-se para cima ou para baixo as linhas tracejadas horizontais, representando dos tempos de ativação.

2.5 Considerações finais

O controlador do motor de indução descrito de forma geral, neste capítulo, é composto por vários blocos formando um sistema em malha fechada, executando um trabalho computacionalmente intensivo e exigente de alta velocidade de processamento. Esta modelagem do sistema, desenvolvida em trabalhos anteriores, visa unificar em uma abordagem uma estratégia de modulação vetorial e uma estratégia de estimação das grandezas do motor, ambas baseadas em redes neurais artificiais. Antes de ser apresentada a contribuição deste trabalho, ou seja, a arquitetura de hardware do protótipo desenvolvido, o capítulo 3 detalha as estruturas internas das redes neurais de modulação e estimação mencionadas. Estas estruturas representam bagagem de conhecimento, desenvolvida em trabalhos anteriores, que serviu como base para o início deste trabalho.

CAPÍTULO 3 - ESTRUTURA INTERNA DOS BLOCOS "CONTROLE PWM SPACE-VECTOR BASEADO EM REDES NEURAIS" E "ESTIMADOR DE FLUXO"

3.1 Introdução

O capítulo 2 apresentou de forma geral todos os blocos envolvidos no funcionamento do controlador do motor de indução deste trabalho. Neste capítulo, serão descritas as estruturas internas dos blocos "Controle PWM Space-Vector Baseado em Redes Neurais" e "Estimador de Fluxo". No capítulo 4, são apresentadas arquiteturas de hardware para implementação de todos os blocos do "Subsistema de Processamento de Sinais". A seção a seguir inicia o capítulo, apresentando a modelagem neural do bloco "Controle PWM Space-Vector Baseado em Redes Neurais".

3.2 Estrutura do bloco "Controle PWM Space-Vector Baseado em Redes Neurais"

Conforme já mencionado no capítulo 2, o bloco "Controle PWM Space-Vector Baseado em Redes Neurais" traduz valores de magnitude e ângulo de referência em comandos de entrada em condução (*turn-on*) e bloqueio (*turn-off*). Estes comandos são enviados como pulsos PWM para chaves eletrônicas presentes no módulo "Inversor baseado em IGBT" ("Subsistema de circuito de potência"). A modelagem do bloco "Controle PWM Space-Vector Baseado em Redes Neurais" foi inicialmente desenvolvida em [25] e [26]. Nestes trabalhos foram obtidas expressões para os tempos de condução e bloqueio através de testes do algoritmo SVM. Estas expressões, depois de simplificadas e unificadas permitiram expressar os tempos de condução e bloqueio de forma gráfica como funções do ângulo de referência de modulação "θ_e". A figura 3.1 apresenta exemplos destes gráficos de tempo de ativação para três regiões de modulação diferentes. As curvas da figura 3.1 foram geradas utilizando um período de amostragem "Ts" igual a 100μs (um dos valores preestabelecidos em [25] e [26]). Três regiões de modulação foram consideradas: Submodulação, Sobremodulação-Modo 1 e Sobremodulação-Modo2.

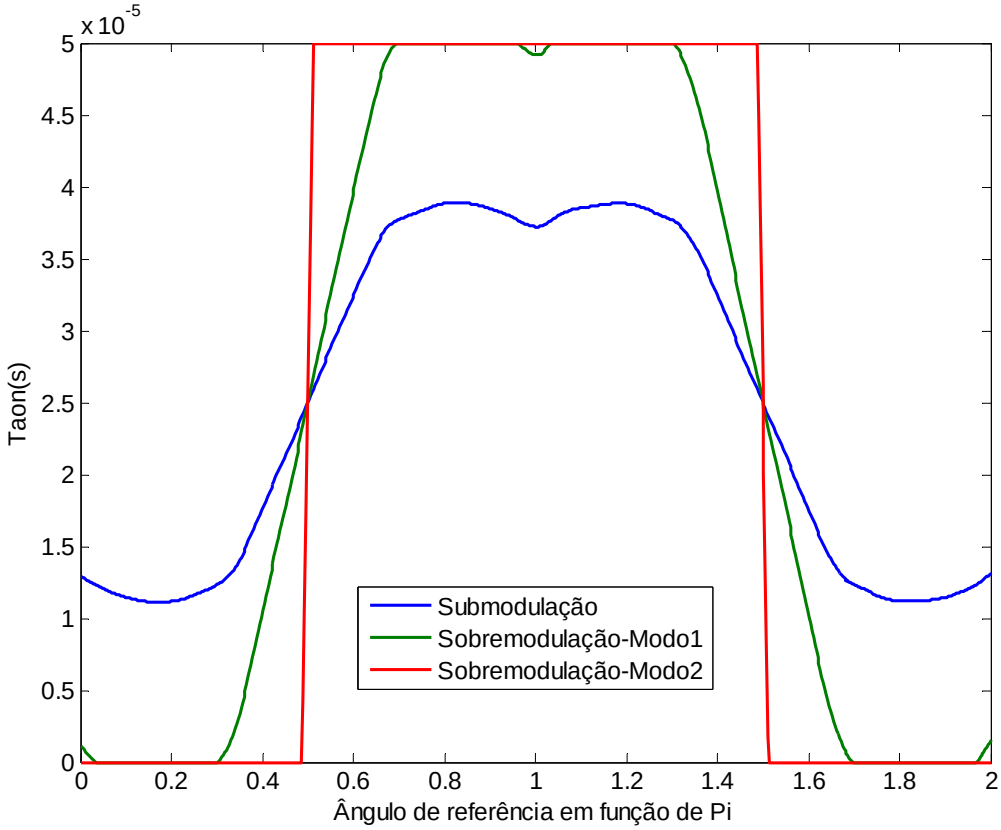


Figura 3. 1 - Gráficos de tempo de ativação gerados por SVM.

O que diferencia uma região de modulação da outra é o chamado índice de modulação (m). O índice de modulação varia entre 0 e 1, sendo expresso pela equação 3.1.

$$m = \frac{V^* \cdot \pi}{2 \cdot Vd} \qquad (3.1)$$

Na equação 3.1 o índice de modulação "m" está relacionado com a magnitude da tensão de referência "V^*" e com a tensão de "Link CC" (ou "V_d", como já apresentado na descrição do "Subsistema de Circuito de Potência" no capítulo 2). A região de Submodulação é definida pelo intervalo 0<m<0,907, a região de Sobremodulação-Modo 1 é definida pelo intervalo 0,907<m<0,952, e a região de Sobremodulação-Modo 2 pelo intervalo 0,952<m<1. Graficamente, as diferenças entre as regiões de modulação ficam evidentes, conforme pode ser notado na figura 3.1. A curva de tempo de ativação para região de Submodulação (azul) não apresenta "achatamentos" como ocorre para os valores de "Ta-on" iguais a 0 e "Ts/2" na curva para a região de "Sobremodulação-

Modo1" (verde). Já a curva para região de "Sobremodulação-Modo 2" (vermelha) apresenta-se praticamente quadrada, e também está limitada entre 0 e "Ts/2".

Utilizando os dados de tempo de ativação mencionados, as redes neurais utilizadas neste trabalho foram modeladas e treinadas em [25] e [26] para depois comporem o bloco "Controle PWM Space-Vector Baseado em Redes Neurais". Um diagrama destas redes é apresentado na figura 3.2.

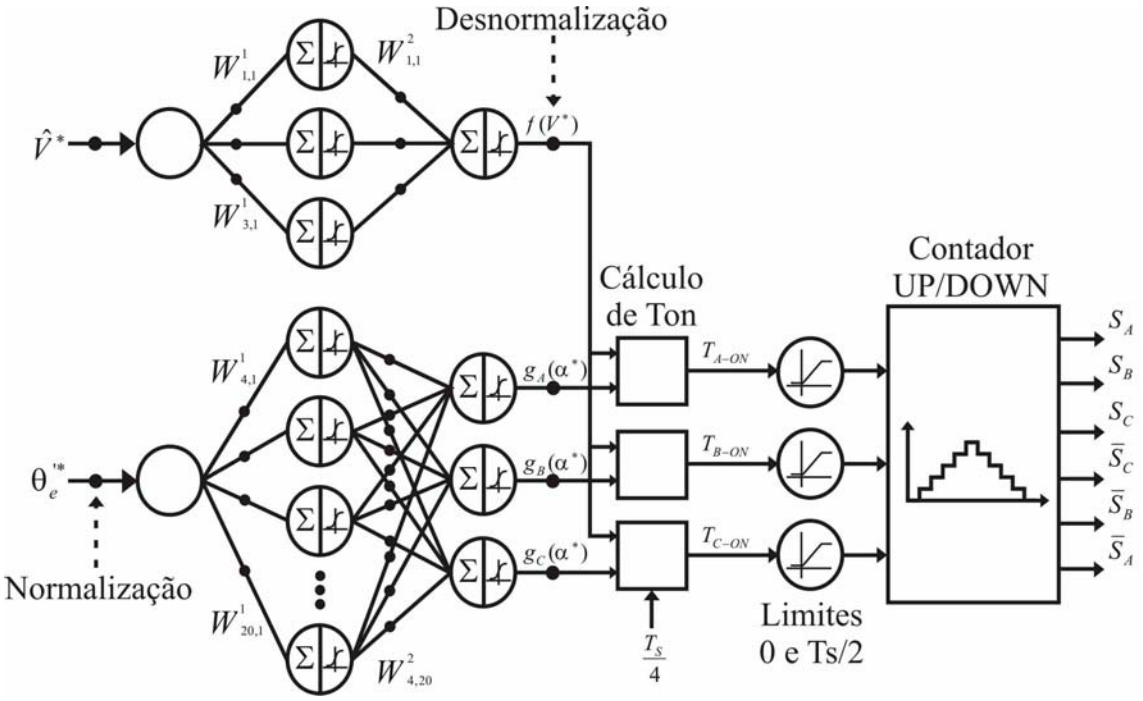


Figura 3. 2 - Diagrama do Controle PWM Space-Vector Baseado em Redes Neurais.

A figura 3.2 apresenta tanto a "Rede de módulo" (rede no topo da figura) quanto à "Rede de ângulo", como redes MLP (*Multi Layer Perceptron*), treinadas utilizando o algoritmo "*Back-Propagation*". A "Rede de módulo" é composta por uma camada de entrada com três neurônios e uma camada de saída composta por um neurônio. Todos os neurônios possuem função de ativação sigmóide e trabalham com pesos sinápticos fixos. Já a "Rede de ângulo" apresenta estrutura mais complexa, contando com 18 neurônios na camada de entrada e três neurônios na camada de saída. Também nesta rede todos os neurônios ponderam suas entradas com pesos fixos e geram saídas a partir de funções de ativação sigmoidais. Operações de normalização de entradas e desnormalização de saídas finais, realizadas pelas duas redes neurais, traduzem valores manipulados dentro das redes para escalas correspondentes ao restante do sistema.

As saídas pós-normalizadas da “Rede de módulo” ($f(V^*)$) e da “Rede de ângulo” ($g_A(\alpha^*), g_B(\alpha^*)$ e $g_C(\alpha^*)$) são utilizadas juntamente com a entrada “Ts/4” em três módulos multiplicadores e acumuladores, obtendo-se como resultados os valores de tempo de ativação: “T_{A-ON}”, “T_{B-ON}”, “T_{C-ON}”. Estes tempos após passarem por blocos que os limitam entre 0 e “Ts/2”, são aplicados aos “Contadores *UP/DOWN*” para geração dos sinais PWM, conforme já explicado no capítulo 2. Os blocos de limitação entre 0 e “Ts/2” causam os “achatamentos” observados nas curvas de tempo de ativação das regiões de Sobremodulação dos modos 1 e 2.

No módulo "Inversor baseado em IGBT" ("Subsistema de circuito de potência"), as saídas de PWM “S_A”, “S_B”, “S_C”, “$\overline{S}_A$”, “$\overline{S}_B$”, “$\overline{S}_C$” acionam IGBTs pertencentes aos três “ramos” diferentes, conforme mostrado na figura 3.3.

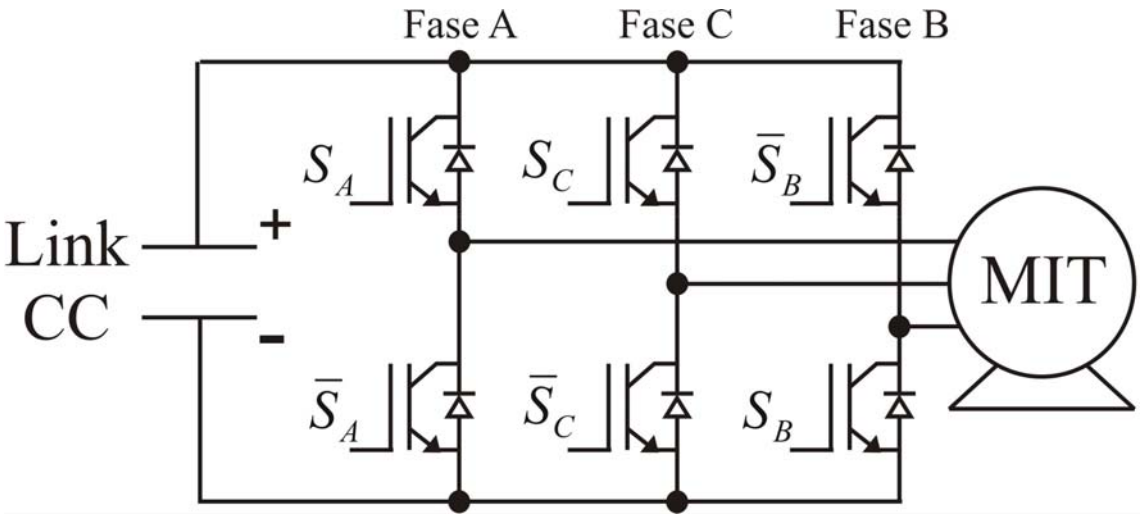


Figura 3. 3 - Interconexão das saídas PWM com inversor de potência e MIT.

Na figura 3.3 a saída “S_A” e sua oposta “$\overline{S}_A$” formam o ramo correspondente à fase A do sistema, o par “S_B” e “$\overline{S}_B$” corresponde à fase B, enquanto “S_C” e “$\overline{S}_C$” representam a fase C, aplicadas ao motor de indução (MIT).

Maiores detalhes sobre técnicas e ferramentas de programação para modelagem e treinamento da “Rede de módulo” e da “Rede de ângulo” estão em [25] e [26]. A seção a seguir abre a segunda parte deste capítulo, descrevendo a estrutura do bloco “Estimador de Fluxo”.

3.3 Estrutura do bloco “Estimador de Fluxo”

Informações sobre a planta controlada, no caso o motor de indução, devem ser obtidas para permitir que um controlador em malha fechada possa alterar adequadamente parâmetros controlados desta planta. A estimação é um método indireto de aquisição de informação sobre a planta, substituindo o uso de sensores considerados inviáveis. O bloco “Estimador de Fluxo” estima o fluxo estatórico do motor de indução através da estrutura de rede apresentada na figura 3.4, idealizada anteriormente em [25].

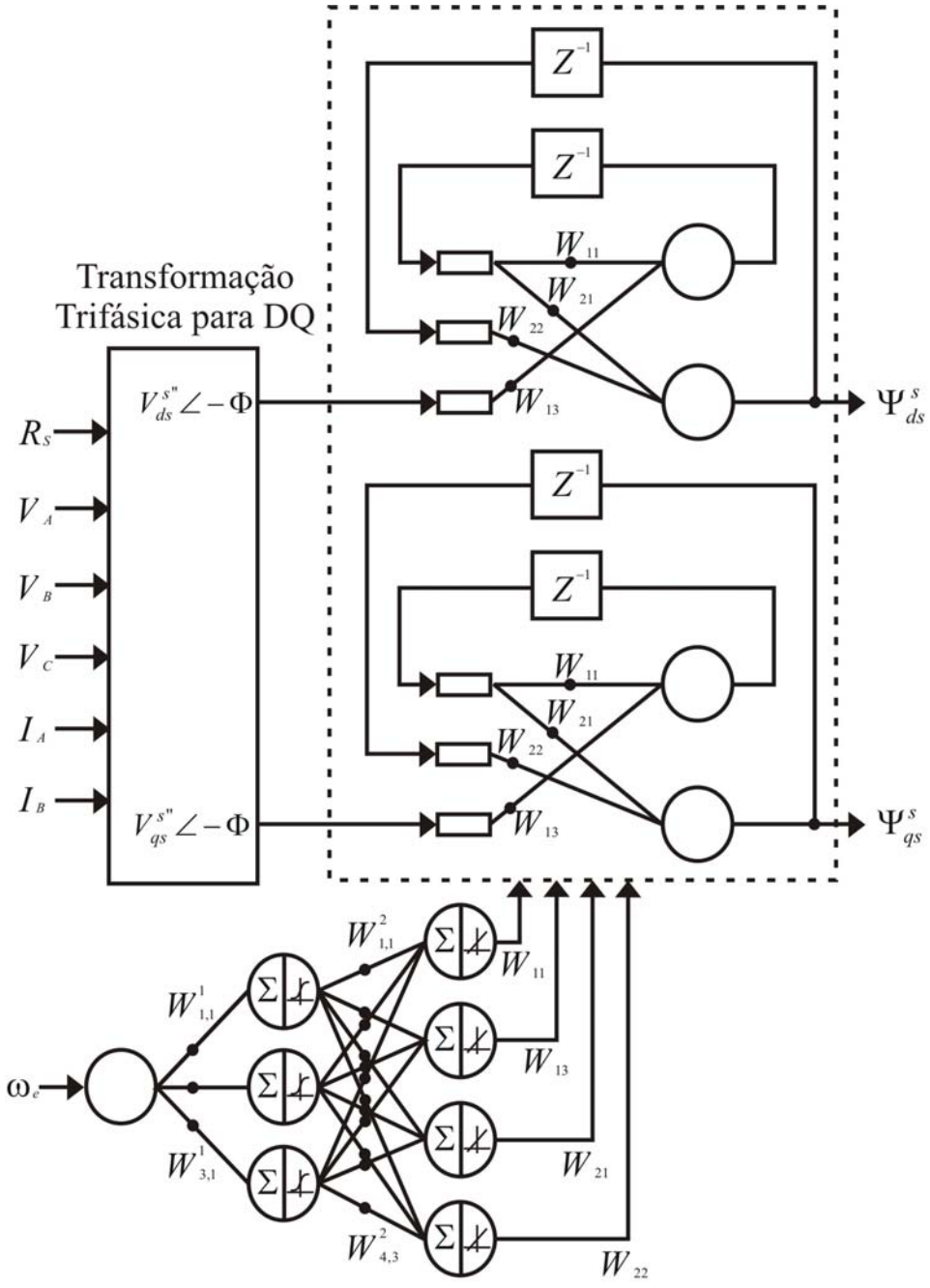


Figura 3. 4 - Redes recorrente e MLP compondo a "RNN-MLP".

A figura 3.4 apresenta um sistema híbrido (rede RNN-MLP), composto por duas redes neurais interligadas. O propósito deste sistema é a estimação de fluxo de estator através de uma estratégia de integração utilizando estágios cascateados de filtros passa baixa (PCLPF), modelados utilizando redes neurais. Segundo [25], esta estrutura de PCLPFs permite a realização de integrações de tensões de entrada dentro de uma larga faixa de freqüência, sem o efeito indesejável de deslocamento (*offset*) CC na resposta.

A rede neural na parte inferior da figura 3.4, uma MLP, é composta por uma camada de entrada com três neurônios com função de ativação sigmóide, e uma camada de saída com quatro neurônios com função de ativação linear. Esta rede recebe uma única entrada, "ω_e" (freqüência de rotação), gerada e normalizada pelo bloco "Estimador de Sinais", apresentado no capítulo 2, seção 2.2. A cada novo início de ciclo de trabalho, a rede MLP usa "ω_e" para gerar, como saída, quatro sinais representados pesos: "W_{11}", "W_{13}", "W_{21}", "W_{22}". Para efetuar seus cálculos, esta rede MLP conta com uma matriz de pesos sinápticos fixos, obtidos por treinamento realizado em trabalho anterior [25].

A rede localizada na parte superior da figura 3.4 (no interior do retângulo tracejado) corresponde a uma RNN (rede neural recorrente). Esta rede possui como entradas pesos sinápticos variáveis, ou seja, as saídas "W_{11}", "W_{13}", "W_{21}", "W_{22}" da rede MLP. Como entradas, esta rede recebe as tensões "$V_{qs}^{s"} \angle -\Phi$" e "$V_{ds}^{s"} \angle -\Phi$", resultantes dos cálculos do bloco "Transformação Trifásica para DQ" (à esquerda na figura 3.4).

O bloco "Transformação Trifásica para DQ" realiza uma transformação de espaço de representação das entradas da rede "RNN-MLP". Esta transformação (transformada de *Park*) obtém as tensões "$V_{qs}^{s"} \angle -\Phi$" e "$V_{ds}^{s"} \angle -\Phi$", a partir de entradas medidas diretamente dos terminais de entrada do motor de indução do sistema. Estas entradas são as correntes de fase "I_A", "I_B", e as tensões de fase "V_A", "V_B", "V_C". Além destas entradas, é utilizado um parâmetro construtivo do motor, ou seja, a resistência de estator (R_S na figura 3.4), obtida por meio de ensaios práticos. Devido à necessidade prática de pré-filtragem dos sinais de tensões e correntes utilizados como entradas pelo bloco "Transformação Trifásica para DQ", as tensões de saída "V_{qs}" e "V_{ds}" estão representadas na figura 3.4 com um deslocamento de fase "$-\Phi$".

A rede RNN-MLP é sistema com memória, ou seja, utilizam em seus cálculos atuais valores obtidos, anteriormente, pela própria rede. Esta característica de trabalhar com dados atrasados no tempo é representada pelos blocos "Z^{-1}" na figura 3.4. Além disso, as funções de ativação dos neurônios da rede "RNN" não são representadas na figura 3.4, já que estes neurônios correspondem a blocos multiplicadores de duas entradas.

Como saídas, a rede RNN gera os fluxos "Ψ_{ds}" e "Ψ_{qs}", representando projeções nos eixos "d" e "q" do fluxo de estator no instante de amostragem, pelo bloco "Estimador de Fluxo", das entradas "I_A", "I_B", "V_A", "V_B", "V_C".

Detalhes sobre as técnicas de treinamento e ferramentas de modelagem, utilizadas na criação das redes do bloco "Estimador de Fluxo" são fornecidos em [25]. Resultados de operação destas redes são apresentados no capítulo 5 deste trabalho.

3.4 Considerações finais

O capítulo 4 voltará a descrever todos os blocos do "Subsistema de processamento de sinais" analisados neste capítulo e no capítulo 2. Porém, haverá enfoque para a descrição da arquitetura e do funcionamento do hardware utilizado na implementação de cada um desses blocos. Tal enfoque faz uso da teoria de máquinas de estados finitos, de conceitos relacionados com linguagens de descrição de hardware (VHDL), e de conceitos sobre representação numérica em ponto flutuante.

Detalhes adicionais sobre algoritmo SVM e estimação de sinais em controle "*sensorless*" podem ser encontradas em [25] e [26].

CAPÍTULO 4 - DESCRIÇÃO DO HARDWARE DO SISTEMA PROPOSTO

4.1 Introdução

Neste capítulo, o "Subsistema de Processamento de Sinais" é novamente analisado, porém, o enfoque está na descrição da arquitetura e do funcionamento do hardware desenvolvido para implementá-lo. Este hardware representa a parte experimental, isto é, o protótipo propriamente dito do trabalho. Os resultados experimentais sobre seu funcionamento serão apresentados no capítulo 5.

4.2 Descrição geral

O hardware do "Subsistema de processamento de sinais" é um protótipo fisicamente composto por três placas de circuito impresso (PCIs): o "Subsistema Digital", o "Subsistema Analógico", e a "Interface com o Usuário". O "Subsistema Digital" representa o circuito de processamento digital central, contendo o DSP (um TMS320C6711 da *Texas Instruments*), o FPGA (um Stratix 2 EP2S60F484C3 da *Altera*), um conversor analógico para digital (ADS8364), além de outros circuitos de suporte (como memórias, registrador paralelo, portas lógicas, gerador de sinal de clock, etc.).

O "Subsistema Analógico" corresponde a um conjunto de circuitos analógicos divididos para atender duas tarefas distintas: leitura e condicionamento de sinais de tensões e correntes obtidos do MIT, e circuito de comando de *gate* das chaves de potência do "Inversor baseado em IGBT". A função de leitura é realizada utilizando sensores de tensão AC (transformadores de medição) e sensores de corrente por efeito Hall. A função de condicionamento de sinais é realizada por filtros "passa-baixa" construídos utilizando amplificadores operacionais. Já a função de gatilho das chaves de potência do Inversor é realizada por circuitos opto-acopladores para isolação e conversão de níveis de tensão de gatilho (do nível digital 3,3V para o nível de potência 15V).

A "Interface com o Usuário" representa um circuito impresso com display de cristal líquido (LCD) e teclado. A partir deste circuito o usuário visualiza o estado do sistema e entra com parâmetros de funcionamento desejados. O diagrama da figura 4.1 apresenta uma idéia geral da arquitetura do protótipo proposto neste trabalho.

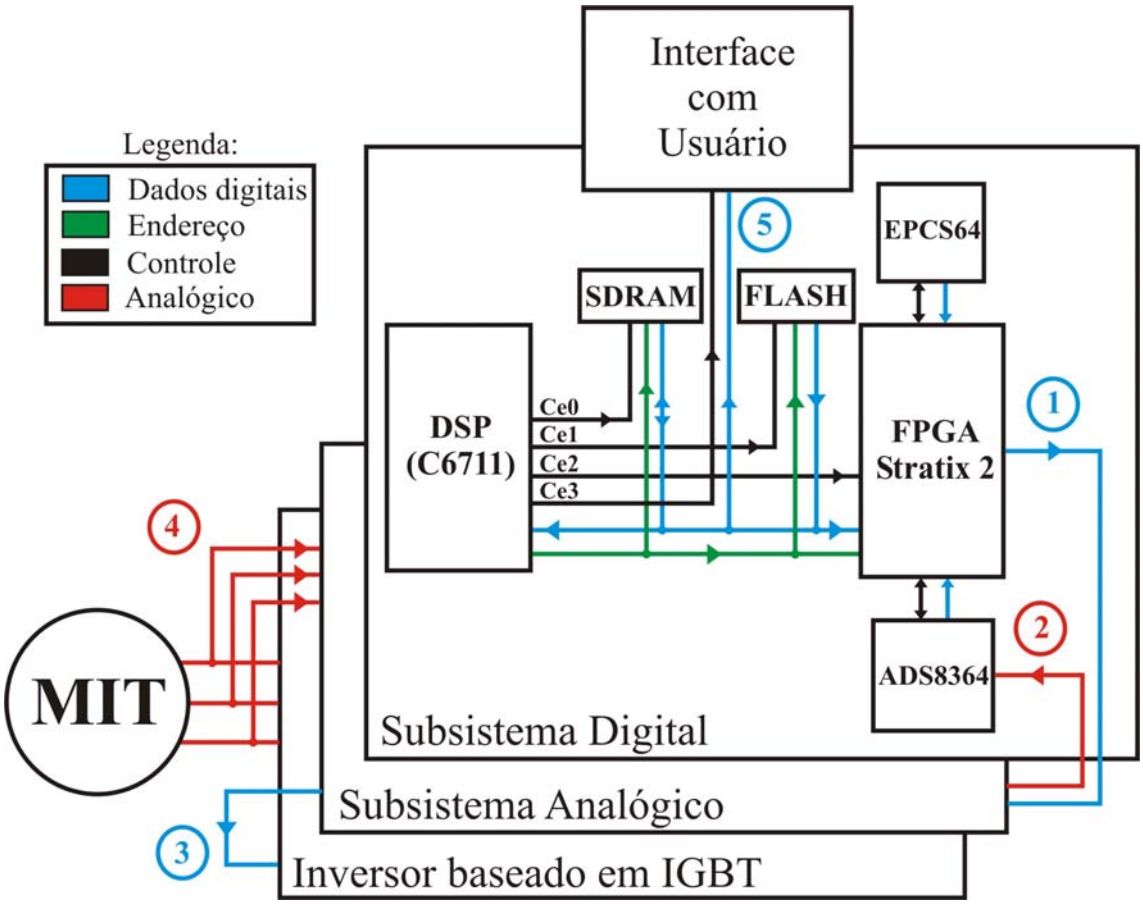


Figura 4. 1 - Protótipo do controlador do MIT baseado em redes neurais.

Na figura 4.1 são evidenciadas as três placas de circuito impresso mencionadas: o "Subsistema Digital", o "Subsistema Analógico" e a "Interface com o Usuário". Além disso, estão representados os componentes de potência "Inversor baseado em IGBT" e MIT (máquina de indução trifásica).

Os números 1, 2, 3, e 4 circunscritos, presentes na figura 4.1, realçam algumas das interligações importantes. O número 1 representa o conjunto de seis saídas digitais PWM (S_A, S_B, S_C e suas opostas), vindas do FPGA, servindo como circuito de comando das chaves de potência do inversor por meio dos circuitos opto-acopladores no "Subsistema Analógico". O número circunscrito 3 representa as próprias ligações dos circuitos opto-acopladores no "Subsistema Analógico" com os circuitos de acionamentos dos IGBTs (chaves) no inversor de potência. O número 2 representa o conjunto de sinais de tensão e corrente da MIT, lidos pelos sensores no "Subsistema Analógico" e convertidos para digital pelo conversor (ADS8364) no "Subsistema Digital". O número 4 representa a interligação do "Inversor baseado em IGBT" com a MIT e, além disso, a interligação dos terminais de entrada da MIT com os sensores de

tensão e corrente no "Subsistema Analógico". O número 5 representa o barramento de controle e de dados interligando a "Interface com o Usuário" com a placa do "Subsistema Digital".

Conforme mostrado na figura 4.1, juntamente com o DSP e o FPGA, o "Subsistema Digital" ainda inclui os módulos importantes: “SDRAM”, “FLASH”, “EPCS64”, e “ADS8364”. A “SDRAM” constitui uma unidade de memória volátil para armazenamento de variáveis manipuladas pelo sistema operativo do DSP e de variáveis utilizadas nos cálculos matemáticos do DSP. O módulo de memória não volátil “FLASH” armazena o programa executado pelo DSP. Este programa é transferido para o DSP através de um processo de *boot* do próprio DSP. O módulo “EPCS64” também é uma memória não volátil, porém, sua função é armazenar um arquivo de configuração enviado para o FPGA durante o processo de *boot* do FPGA. Por fim, o módulo “ADS8364” representa, como já mencionado, o conversor analógico para digital. Este conversor digitaliza cinco sinais analógicos vindos do “Subsistema Analógico” a cada ordem de início enviada pelo FPGA. Os resultados destas conversões são utilizados pelo FPGA em seus cálculos internos. Os módulos mencionados brevemente nesta seção de descrição geral serão descritos com mais detalhes nas seguintes, conforme for sendo oportuno.

O enfoque deste capítulo está na descrição do hardware do protótipo do controlador do motor de indução utilizando como base a discussão dos capítulos 2 e 3. Seguindo esta idéia, os blocos do controlador do MIT serão descritos seguindo a ordem obedecida no capítulo 2, ou seja: "Estimador de Fluxo", "Estimador de Sinais", "Controlador", e "Controle PWM Space-Vector Baseado em Redes Neurais". Fisicamente, estes são sistemas implementados pelo núcleo de processamento DSP mais FPGA, pertencente ao “Subsistema Digital”. Porém, como um ciclo completo de controle do MIT envolve também o "Subsistema Analógico" e a "Interface com o Usuário", serão inicialmente apresentados subsistemas não mencionados no capítulo 2.

Na seção a seguir será descrito o "Subsistema Analógico" do controlador do motor de indução. Seguindo um encadeamento de sistemas, nas seções posteriores serão descritos os blocos do “Subsistema Digital” em si. O objetivo é finalizar o capítulo com uma descrição da malha fechada, compondo o “Subsistema de Processamento de Sinais” como um todo.

4.3 Subsistema Analógico (medição e condicionamento de sinais)

Cada ciclo de trabalho do controlador do motor de indução proposto inicia-se pela medição de tensões e correntes dos terminais de entrada do motor controlado. Esta tarefa é realizada pelos elementos na placa de circuito impresso do "Subsistema Analógico". Basicamente, os circuitos do “Subsistema Analógico” realizam medição e condicionamento de sinais coletados dos terminais de entrada da MIT. A figura 4.2 mostra um diagrama dos estágios envolvidos para a medição de tensões de entrada da MIT.

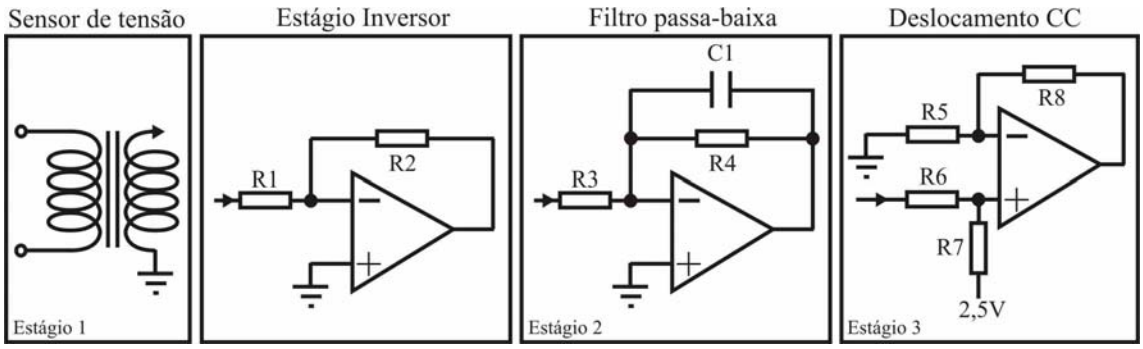


Figura 4. 2 - Medição de tensão de entrada da MIT.

A figura 4.2 representa um diagrama de circuito de medição de tensão de fase na entrada do MIT. O protótipo real replica o sistema da figura 4.2 três vezes, uma vez para cada tensão trifásica necessária (V_a ,V_b ,V_c). No diagrama da figura 4.2, o estágio 1 representa um sensor de tensão, fisicamente representado no protótipo por um transformador de medição que reduz a tensão de fase lida, de 220V AC para 6V AC. Já o estágio 2 apresenta uma dupla função, a primeira consiste em atenuar possíveis componentes de alta freqüência do sinal através de um filtro passa-baixa com freqüência de corte próxima a 300Hz. A segunda função consiste em atenuar a amplitude do sinal medido, tornando seu valor de pico a pico dentro da faixa de -2,5V a +2,5V. Inevitavelmente, a existência de um filtro como o do estágio 2 causa um deslocamento de fase de 180 graus no sinal medido. Para corrigir este problema, no protótipo final foi incluído um “Estágio Inversor” (circuito com amplificador operacional na entrada do filtro do estágio 2 na figura 4.2). No estágio 3 o sinal filtrado fornecido pelo estágio 2 é deslocamento da faixa de -2,5V a +2,5V para a faixa de 0 a 5V. Este deslocamento do sinal para uma faixa acima de 0 é uma exigência para as entradas analógicas do

conversor ADS8364. Detalhes adicionais dos circuitos de medição de tensão estão no apêndice A deste trabalho.

De forma análoga ao descrito para medições de tensão, medições de corrente também envolvem estágios, conforme mostrado na figura 4.3.

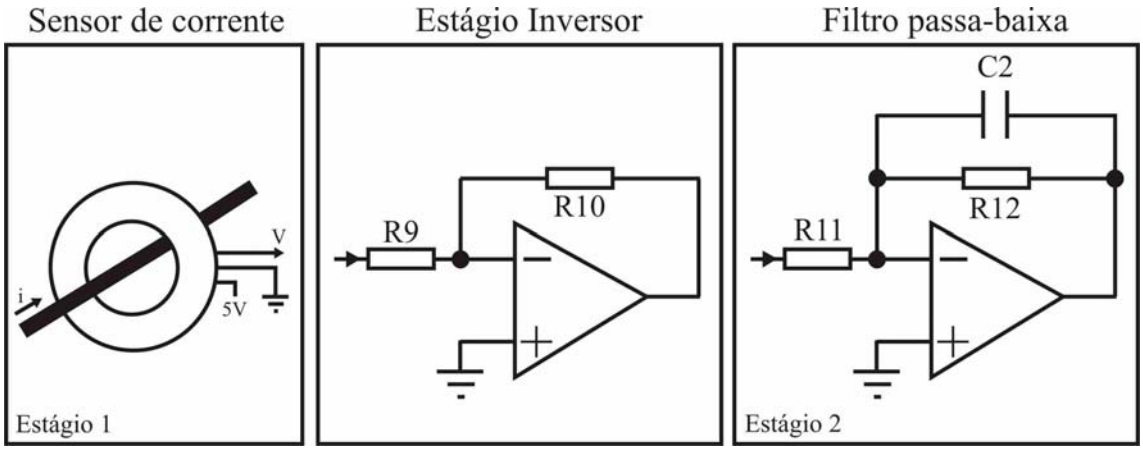


Figura 4. 3 - Medição de corrente.

Na figura 4.3 o estágio 1 representa um sensor Hall (LTS-15NP) alimentado por 5V e que transforma o valor de corrente de fase medido em um valor de tensão proporcional. Este valor de tensão já está dentro da faixa de 0 a 5V e é filtrado pelo circuito passa-baixa do estágio 2 após passagem pelo "Estágio Inversor" anterior. O circuito da figura 4.3 é replicado duas vezes na placa de circuito impresso do "Subsistema Analógico". Através desta replicação, é possível medir as correntes de fase I_a e I_b, sendo I_c obtida analiticamente conforme já mencionado no capítulo 2. Os circuitos completos para medição de corrente de fase podem ser encontrados no apêndice A deste trabalho.

O "Subsistema Analógico" realiza um processo de adequação das amplitudes dos sinais de corrente e tensão para compatibilidade com o padrão de entrada do conversor A/D. Porém, para possibilitar o processamento realizado pelos subsistemas que recebem estes sinais, é necessária a conversão destes sinais para representação digital. Tal tarefa é realizada pelo conversor descrito na seção a seguir, o ADS8364.

4.4 Conversão analógica para digital (ADS8364)

Os sinais condicionados fornecidos pelos circuitos de medição descritos na seção 4.3 (ou seja, os sinais proporcionais a I_a, I_b, V_a, V_b, e V_c já condicionados) vão para o conversor analógico para digital ADS8364, situado na placa de circuito impresso do "Subsistema Digital". Este conversor possui seis entradas analógicas diferenciais no total, e é capaz de converter simultaneamente estas seis entradas analógicas para valores digitais de 16 bits (porém apenas 14 bits significativos), armazenando-os no próprio conversor. A conversão para digital é necessária para que os sinais de medição analógicos possam ser depois processados pelos próximos sistemas compondo o "Estimador de Fluxo", situados dentro do FPGA (ainda na placa do "Subsistema Digital"). A figura 4.4 representa de forma geral a interligação dos sinais analógicos de entrada condicionados I_a, I_b, V_a, V_b, e V_c com o conversor ADS8364.

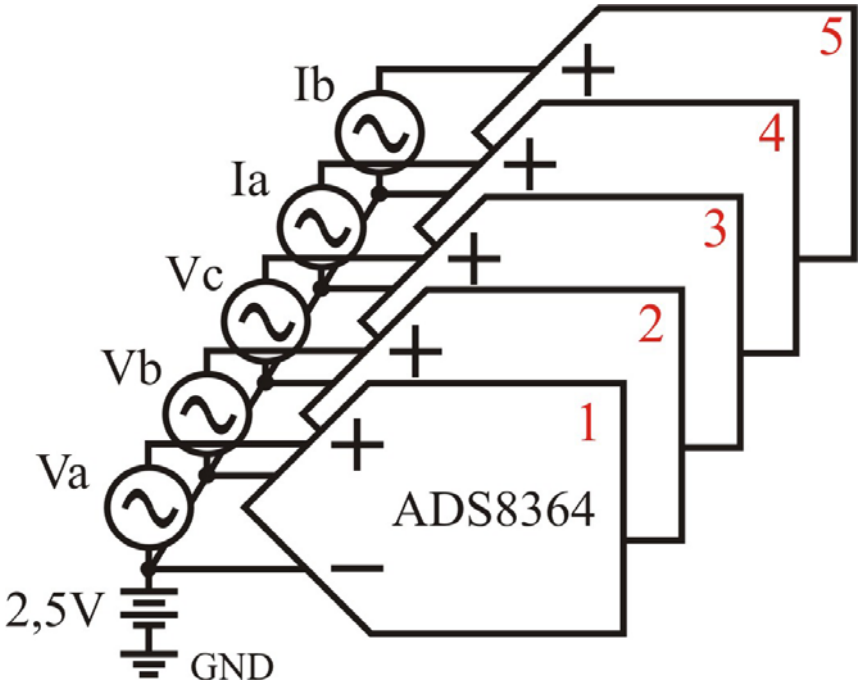


Figura 4. 4 - Interligação com o conversor ADS8364.

Na figura 4.4 os cinco sinais analógicos já condicionados são representados por senóides atenuadas e com deslocamento DC de +2,5V em relação ao GND. O conversor ADS8364 está representado somente por cinco de seus circuitos (ou canais) internos de amostragem e conversão. As entradas na figura 4.4 estão interligadas com o ADS8364 no chamado modo "*Single-Ended Input*".

Quanto ao funcionamento, basicamente cada operação de conversão se inicia quando três sinais de "HOLD" de entrada do conversor ADS8364 são colocados em nível 0 pelo FPGA. Aproximadamente depois de 16 ciclos de um sinal de *clock* fornecido ao conversor, os dados resultantes desta conversão são carregados para registradores internos no ADS8364. Com uma freqüência de *clock* máxima de 5MHz, o conversor ADS8364 é capaz de capturar 250K amostras de cada canal de entrada por segundo (considerando a ocorrência de uma conversão a cada 20 períodos do *clock* de 5MHz).

No final de cada conversão do ADS8364, o FPGA envia sinais de controle para uma interface paralela assíncrona do conversor, e palavras de 16 bits nos registradores internos do ADS8364, resultantes da conversão, vão sendo enviadas ao FPGA uma de cada vez. O *clock* de 5 MHz mencionado não é utilizado durante esta troca de dados entre o FPGA e o conversor, pelo próprio fato de se estar utilizando uma interface assíncrona nesta transferência. Na realidade, este sinal de 5 MHz só necessário para o funcionamento dos circuitos de conversão dentro do ADS8364. Conforme será explicado adiante, sinais de *clock* de 10 MHz e 20 MHz são utilizados em máquinas de estado dentro do FPGA para controlar a interface assíncrona do conversor ADS8364 e trazer as palavras de 16 bits para o FPGA.

Cada palavra de 16 bits representa um número binário em complemento de dois, correspondente a um valor analógico dentro da faixa de 0 a 5V. A representação em complemento de dois divide a faixa total de representação em 16 bits ao meio, com uma metade representando números acima da referência 2,5V e a outra representando números abaixo dessa referência. A tabela 4.1 mostra como o ADS8364 representa em 16 bits alguns valores importantes.

Tabela 4. 1 - Codificação em 16 bits das entradas analógicas do ADS8364.

Código de saída em 16 bits	Tensão representativa
0111111111111111b	4,999924V
...	...
0000000000000001b	2,500038V
0000000000000000b	2,5V
1111111111111111b	2,499962V
...	...
1000000000000000b	0V

Com os sinais quantizados de tensões e correntes já disponibilizados na representação digital, entra em ação o bloco “Estimador de Fluxo”, descrito em maiores detalhes na seção a seguir.

4.5 Hardware do bloco Estimador de Fluxo

O hardware do bloco "Estimador de Fluxo" é composto por três subsistemas digitais principais: a "Interface com ADS", o "Módulo de Transformação Trifásica", e a "Rede Recorrente Híbrida com MLP". Estes subsistemas estão interligados internamente em um FPGA “Stratix 2”. A função e a estrutura destas várias unidades necessárias à estimação de fluxo são descritas nesta seção.

4.5.1 Interface com ADS

As palavras de 16 bits resultantes das conversões do dispositivo ADS8364 são trazidas dos registradores internos deste dispositivo para dentro do FPGA por meio de uma unidade residente no próprio FPGA. Esta unidade, a "Interface com ADS", representa a primeira interação de um sistema interno do FPGA com o ambiente exterior. Um diagrama de entradas e saídas da "Interface com ADS" é mostrado na figura 4.5.

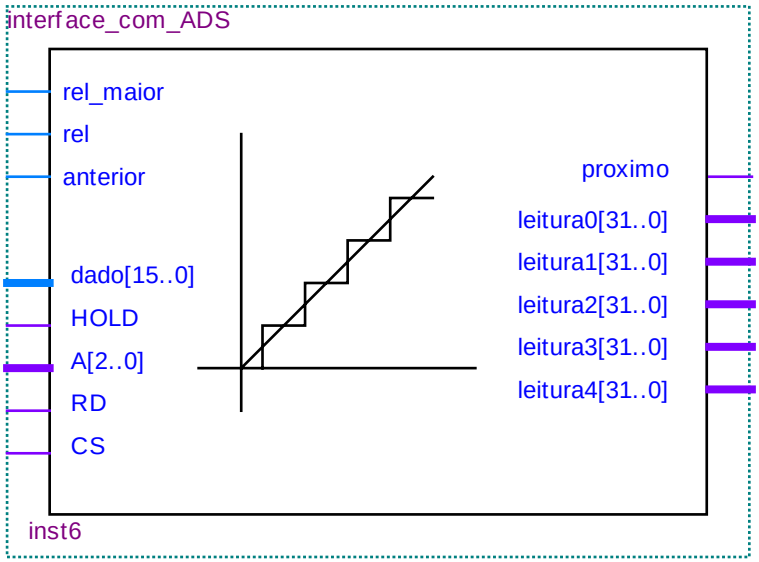


Figura 4. 5 - Unidade "Interface com ADS", dentro do FPGA.

A unidade apresentada na figura 4.5 representa um sistema descrito em VHDL e que, basicamente, funciona como uma máquina de estados finitos. Sua principal função

é adquirir e transformar as palavras resultantes das conversões do ADS8364 em valores de 32 bits padrão IEEE 754 (representação binária em ponto flutuante).

Nas laterais da figura 4.5 estão as entradas e saídas da unidade. Um novo ciclo de operação da "Interface com ADS" é sempre iniciado por uma borda de subida (pulso) aplicada na entrada "anterior". Este pulso de comando é sempre enviado pelo dispositivo mestre da placa de circuito do "Subsistema Digital", ou seja, o DSP TMS320C6711 (não presente na figura 4.5). Já a entrada "rel_maior" (figura 4.5) recebe um sinal de clock de 20MHz, sinal que serve de referência para definir os momentos em que a "Interface com ADS" amostra a entrada "anterior". A cada borda de subida do sinal "rel_maior" um processo em VHDL na "Interface com ADS" verifica se o estado da entrada "anterior" é igual a 1. Em caso afirmativo é iniciado um conjunto de transições de estados representando um ciclo de operação completo da "Interface com ADS". Estas transições entre estados estão sincronizadas com um sinal de *clock* de 10MHz aplicado na entrada "rel." (figura 4.5). Optou-se pela utilização de uma menor freqüência para "rel" (10MHz em vez de 20MHz) devido às restrições de temporização. Se as transições entre estados da "Interface com ADS" ocorressem a partir de um clock de 20MHz, acabariam sendo enviados comandos para ADS8364 que desobedeceriam aos limites de operação do próprio ADS8364. Ambos os sinais de clock (10MHz e 20MHz) fisicamente são gerados por um circuito gerador de *clock* dentro do próprio FPGA, utilizando como referência um sinal de *clock* enviado para o FPGA pelo DSP TMS320C6711 do protótipo.

Prosseguindo com a descrição entrada-saída da figura 4.5, ainda aparecem os sinais "dado[15..0]", "HOLD", "A[2..0]", "RD", e "CS". Estes representam sinais de dados e de controle interligando fisicamente a "Interface com ADS" no FPGA com a interface paralela assíncrona do dispositivo externo ADS8364. Em "dado[15..0]" chegam as palavras de 16 bits resultantes das conversões do ADS8364. O sinal "HOLD" representa um comando que a "Interface com ADS" envia para o ADS8364 para iniciar uma nova conversão. Fisicamente "HOLD" está interligado às três entradas de início de conversão "HOLDA", "HOLDB" e "HOLDC" do ADS8364 (ver detalhes nos apêndices). Os sinais restantes "A[2..0]", "RD", e "CS", representam comandos de endereço e ativação de escrita que a "Interface com ADS" envia para interface paralela assíncrona do conversor ADS8364.

Observando o lado direito da figura 4.5, visualiza-se mais acima a saída chamada "próximo". Esta saída emite um pulso em nível lógico 1 quando a "Interface

com ADS" termina todo um ciclo de operação. Esta sinalização avisa outros sistemas no FPGA que as saídas da "Interface com ADS" estão estabilizadas e podem ser lidas.

Ainda à direita da figura 4.5, podem ser vistas cinco saídas de 32 bits "leitura0[31..0]" a "leitura4[31..0]". Estas saídas são representações, em ponto flutuante (IEEE 754), das palavras de 16 bits recebidas do conversor ADS8364. A "Interface com ADS" converte palavras de 16 bits para 32 bits utilizando a equação 4.1.

$$Valor_{32bits} = (Valor_{16bits\_ADS8364} * 2^{-16} + 1) * 2^{0} = Valor_{16bits\_ADS8364} * 2^{-16} + 1 \qquad (4.1)$$

A equação 4.1 representa em ponto flutuante um número em ponto fixo sem parte fracionária e de 16 bits ($Valor_{16bits\_ADS8364}$), inicialmente deslocando 16 casas para direita deste valor em ponto fixo. Desta forma, isto faz com que o valor em ponto fixo represente a mantissa de um valor em ponto flutuante. A adição do valor 1 na equação 4.1 é necessária pois a representação em ponto flutuante IEEE 754 normalizada pressupõe a existência do dígito binário 1 da unidade, mesmo que este dígito não apareça na mantissa da representação binária do número. O bit de sinal do número em ponto flutuante gerado é igual a zero para amostras de tensões ou de correntes com módulo 0 ou positivo (ou seja, maior ou igual a 2,5V após condicionamento). Já amostras de tensões ou correntes com módulo negativo (abaixo de 2,5V após condicionamento) são representadas utilizando um bit de sinal igual a um. O expoente do número em ponto flutuante gerado é definido como sendo igual a "01111111_b" (zero ou 2^0 no padrão IEEE 754 de 32 bits).

Números em "$Valor_{16bits\_ADS8364}$" com módulo negativo (abaixo de 2,5V após condicionamento) devem ser pré-processados pela "Interface com ADS" antes da aplicação da equação 4.1. Estes números, na realidade, são recebidos do conversor ADS8364 como números negativos na representação binária em complemento de 2. Durante o pré-processamento a "Interface com ADS" utiliza a equação 4.2 para inverter bit a bit a variável $Valor_{16bits\_ADS8364}$, obtendo $\overline{Valor_{16bits\_ADS8364}}$, e depois adicionar este resultado com o valor "0000000000000001_b".

$$Valor_{16bits\_final} = \overline{Valor_{16bits\_ADS8364}} + 0000000000000001_b \qquad (4.2)$$

O número em ponto fixo representado por "$Valor_{16bits\_final}$" consiste no valor realmente utilizado como entrada na equação 4.1 para casos de palavras de 16 bits em ponto fixo, representando amostras de medições com módulos negativos.

No ciclo de trabalho do bloco "Estimador de Fluxo", os valores em ponto flutuante produzidos pela "Interface com ADS" são direcionados para os subsistemas seguintes. Estes subsistemas são responsáveis por operações como transformação trifásica (equações 2.1 a 2.3) e estimação das componentes do fluxo de estator nos eixos dq (equações 2.6 e 2.7). Detalhes adicionais são fornecidos na seção a seguir.

4.5.2 Transformação Trifásica e Rede Recorrente Híbrida com MLP

Para finalizar um ciclo de operação do bloco "Estimador de Fluxo" entra em operação o último de seus subsistemas, a "Rede Recorrente Híbrida com MLP" (RNN-MLP). Uma descrição detalhada da estrutura interna da rede “RNN-MLP” foi apresentada no capítulo3, seção 3.3. A figura 4.6 apresenta a rede “RNN-MLP” dentro do FPGA, implementada em VHDL utilizando o software Quartus 2 da Altera.

A estrutura complexa apresentada na figura 4.6 organiza a rede RNN-MLP como um conjunto de processadores digitais (neurônios) interligados, formando, a primeira vista, duas camadas. Somente existem duas camadas de neurônios na figura 4.6, pois a segunda camada da rede MLP (mostrada com quatro neurônios na seção 3.3 do capítulo 3) utiliza, na figura 4.6, os mesmos processadores que compõem a rede neural RNN. Esta estratégia de compartilhamento de recursos consiste em uma otimização do espaço utilizado em hardware. Além dos neurônios, na figura 4.6 estão destacados os módulos “Controlador de Camada 01” e “Controlador de Camada 02”. Cada um destes módulos é responsável por fazer com que os cálculos de cada neurônio da camada gerenciada pelo controlador sejam realizados paralelamente. A presença destes controladores permite, ainda, sincronizar o fluxo de dados entre camadas de uma rede, utilizando uma estratégia de serialização do fluxo de dados entre as camadas adjacentes. De acordo com esta estratégia, deve-se garantir que o controlador da camada 02 somente ative sua camada de neurônios quando o controlador da camada 01 tiver avisado sobre o término dos cálculos da camada 01.

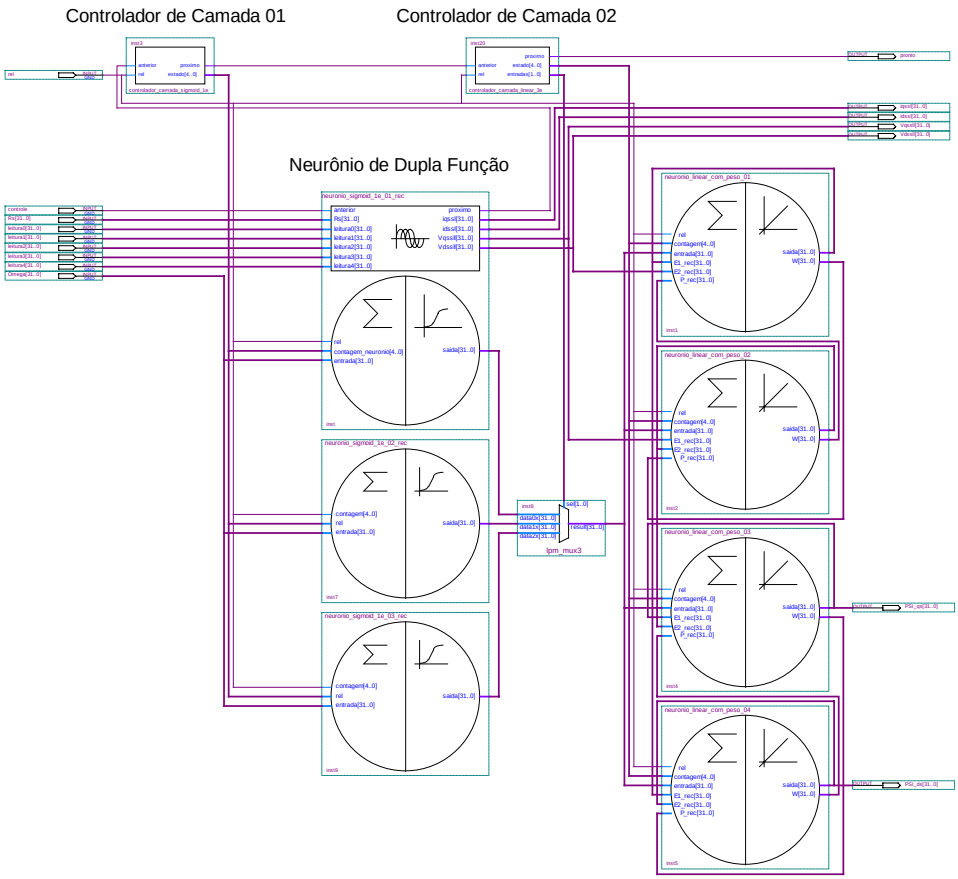


Figura 4. 6 - Rede híbrida "RNN-MLP" no FPGA.

Ainda em realce na figura 4.6, aparece o neurônio chamado “Neurônio de Dupla Função”. Este neurônio está localizado mais acima na camada 01 da rede RNN-MLP. A cada ciclo de trabalho da rede RNN-MLP ele primeiro faz a função do bloco “Transformação Trifásica para DQ” (capítulo 3, seção 3.3) e depois age realmente como o neurônio de primeira camada da estrutura da rede RNN-MLP.

Conforme já mencionado, a estrutura da rede RNN-MLP dentro do FPGA, mostrada na figura 4.6, é complexa. Ela foi somente apresentada de forma geral até este momento. Para um melhor entendimento sobre seu funcionamento é necessária uma análise mais profunda da arquitetura interna dos neurônios digitais e dos módulos controladores de camada utilizados. As seções a seguir descrevem um exemplo de controlador de camada de rede neural, um exemplo de neurônio digital com função de ativação linear, e por fim um neurônio digital com função de ativação sigmóide.

4.5.2.1 Controlador de Camada de Rede Neural

Conforme já mencionado, o módulo controlador de camada tem como função garantir o paralelismo de processamento entre os neurônios da camada de rede que ele comanda. Para atingir este objetivo, o controlador de camada funciona como um módulo de "lógica de próximo estado", compondo com os neurônios da camada controlada uma máquina de estados finitos síncrona. A figura 4.7 apresenta um diagrama mostrando um controlador comandando uma camada de neurônios com três entradas e função de ativação sigmóide.

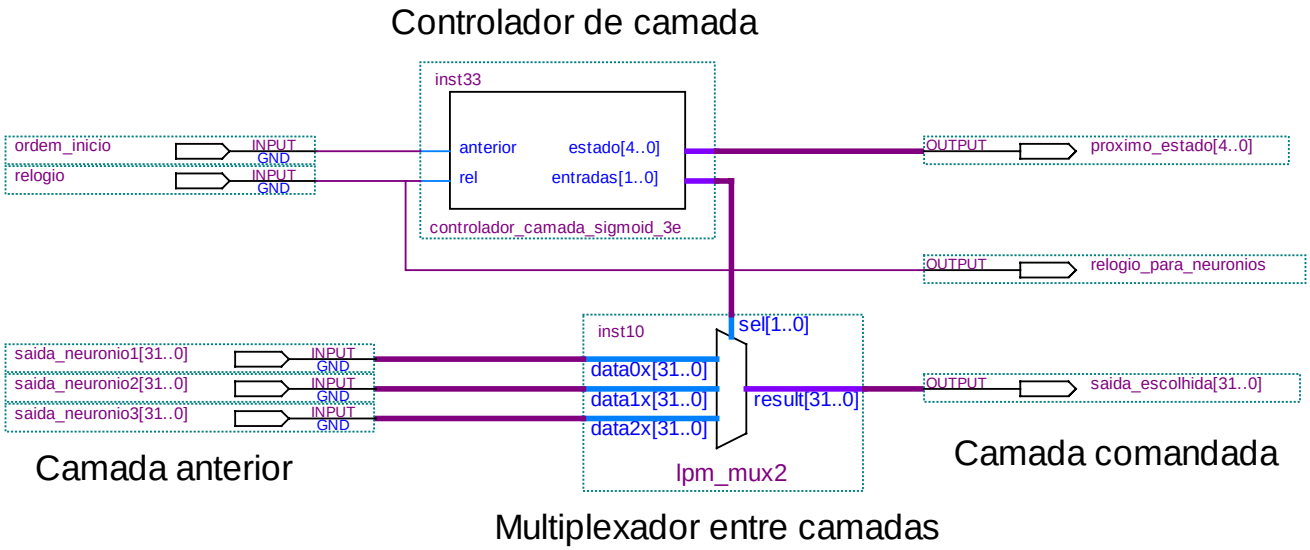


Figura 4. 7 - Interconexões do controlador de camada em uma rede neural.

Na figura 4.7 o módulo controlador de camada aparece representado como uma entidade tendo as entradas "anterior" e "rel", e as saídas "estado[4..0]" e "entradas[1..0]". Como elementos adicionais na figura 4.7, aparecem o "Multiplexador entre camadas", o conjunto de conectores de entrada "Camada anterior", e o conjunto de conectores de saída "Camada comandada".

Basicamente, o controlador de camada funciona como um contador, ele incrementa a saída "estado[4..0]" (de 5 bits) a cada borda de descida do sinal "rel" (ligado a entrada "clock"). Esta contagem representa exatamente a saída "próximo estado[4..0]" enviada para todos os neurônios da camada controlada para definir qual operação cada neurônio fará no estado atual. Paralelamente a contagem emitida durante cada estado pela saída "estado[4..0]", o controlador de camada emite um valor de seleção em "entradas[1..0]". Este valor de seleção permite ao "Multiplexador entre

camadas" definir qual saída de neurônio da "Camada anterior" deverá ser processada por todos os neurônios da "Camada comandada" no estado atual.

Alguns controladores de camada ainda apresentam uma saída adicional chamada "próximo". A partir desta saída um "controlador 01" informa a um "controlador 02" adjacente sobre o término dos cálculos da camada sob responsabilidade do próprio "controlador 01".

O incremento de contagem efetuado pelo controlador de camada vai de 0 até um limite máximo. Este limite máximo depende do número de estados exigidos para que os neurônios da "Camada comandada" realizem todos os seus cálculos. No caso da figura 4.7, os neurônios da camada comandada passam por 20 estados de processamento, o que exige a utilização de pelo menos 5 bits na representação destes estados.

Além dos sinais "próximo estado[4..0]" e "saída_escolhida[31..0]" (selecionado com auxílio do "Multiplexador entre camadas"), os neurônios de uma mesma camada recebem o mesmo sinal de clock utilizado pelo controlador da camada. Isto permite que os neurônios estejam sempre sincronizados com as trocas de estado efetuadas pelo controlador de camada a cada borda deste sinal de clock.

A idéia do controlador de camada descrita define uma interação entre controlador e neurônio replicada para todos os neurônios de uma mesma camada de rede neural. A utilização de controladores de camada em cascata permite que camadas adjacentes possam ser interligadas mesmo que uma camada tenha um tipo de neurônio diferente dos neurônios da outra. Geralmente, um tipo de neurônio de uma camada se diferencia do tipo de neurônio de outra camada, devido à função de ativação utilizada por cada tipo. A seção a seguir descreve a arquitetura e o funcionamento do tipo de neurônio digital mais simples implementado neste trabalho, o neurônio com função de ativação linear.

4.5.2.2 Neurônio digital com função de ativação linear

O neurônio com função de ativação linear é uma máquina de estados composta por duas unidades ou circuitos principais a "Unidade de Decisão" e a "Unidade de Execução". A "Unidade de Execução" realiza cálculos internamente através de um somador e um multiplicador. Estas duas unidades de cálculo usam duas entradas codificadas de acordo com o padrão de ponto flutuante de 32 bits IEEE 754. Tanto o somador quanto o multiplicador são unidades monolíticas (não são divididas em

estágios de *pipeline*). A "Unidade de Decisão" comanda a "Unidade de Execução" executando um programa que representa a "Lógica de saída" de uma máquina de estados. A "Lógica de próximo estado" e a "Memória de estados" dessa máquina de estados estão fora do neurônio. Ao se construir uma rede neural com muitos neurônios trabalhando em paralelo, é importante replicar em cada neurônio o mínimo necessário de circuitos. Em uma mesma camada de rede neural se todos os neurônios fossem máquinas de estados completas, haveria excessiva repetição das unidades de "Lógica de próximo estado" e "Memória de estados". Uma maior economia de recursos é conseguida colocando-se somente a "Lógica de saída" em cada neurônio. Tanto a "Lógica de próximo estado" quanto à "Memória de estados" podem ser colocadas em um "Controlador de Camada" que envia o valor dos estados para todos os neurônios da camada, a cada transição de um sinal de sincronismo "rel". Terminados todos os cálculos dos neurônios em uma camada, o "Controlador de Camada" destes neurônios pode enviar um aviso ativando o "Controlador de Camada" dos neurônios da camada seguinte na rede. Como já mencionado, esta comunicação entre os controladores de camadas possibilita um fluxo organizado de informações entre camadas.

Externamente, este neurônio possui 4 entradas. A entrada "rel", de 1 bit, serve como entrada de sincronismo da máquina de estados na "Unidade de Decisão". Pela entrada "contagem", de 4 bits, entra o valor do estado atual a ser executado enviado pelo "Controlador de Camada". Em "entrada" (de 32 bits) chega o valor em ponto flutuante que representa a entrada de dados do neurônio. A entrada de 32 bits "atual" recebe valores em ponto flutuante, calculados pela "Unidade de Execução" durante a operação do neurônio.

Para entregar a resposta dos cálculos internos realizados pelo neurônio, existe o terminal de 32 bits chamado "saída". Esta resposta é válida somente depois de um período conhecido como "latência". A latência é o atraso de resposta, ou o tempo que o neurônio gasta para produzir um resultado em "saída" correspondente ao valor atual em "entrada". A latência depende de fatores como freqüência da entrada "rel", do número de estados por que passa o neurônio, e da velocidade das unidades de cálculo na "Unidade de Decisão". Um diagrama geral do neurônio linear com as entradas e a saída descritas acima pode ser visto na figura 4.8.

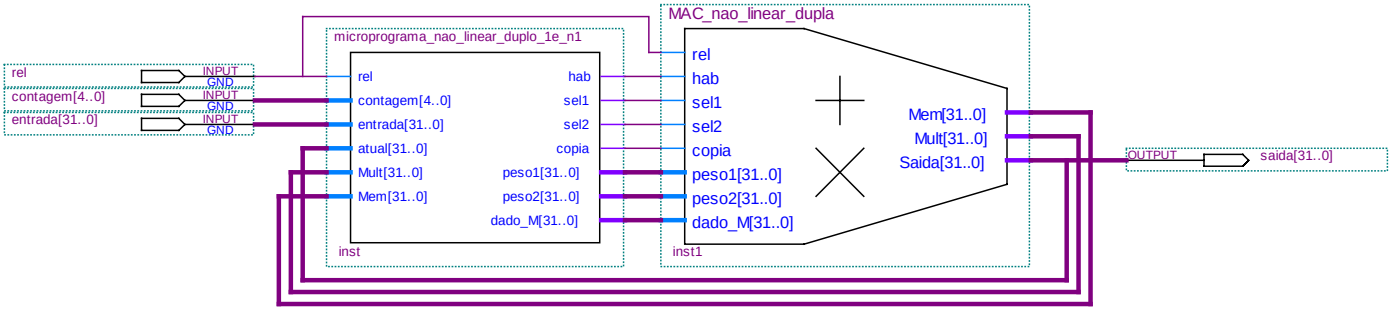


Figura 4. 8 - Diagrama do neurônio de 1 entrada e com função de ativação linear.

A "Unidade de Decisão" é o bloco à esquerda (ligado ao terminal "entrada") na figura 4.8 e a "Unidade de Execução" é o bloco à direita (ligado ao terminal "saída"). As ligações entre os dois blocos ("hab", "sel1", "sel2", "peso1" e "peso2") são sinais de controle que a "Unidade de Decisão" usa para manipular o somador e o multiplicador na "Unidade de Execução".

Esta é uma máquina que executa um programa de controle baseado em 10 estados. Uma descrição destes estados aparece na tabela 4.2.

Tabela 4. 2 - Microprograma da "Unidade de Decisão" do neurônio com função de ativação linear.

Rel	Estado atual	Próximo estado	Hab,sel1,sel2	Peso 1	Peso 2
↑	0	1	000	-	-
↑	1	2	100	XS1	1.0
↑	2	3	101	XM1	Entrada
↑	3	4	111	-	-
↑	4	5	100	P1	Atual
↑	5	6	101	BIAS	1.0
↑	6	7	111	-	-
↑	7	8	100	C1	Atual
↑	8	9	101	C0	1.0
↑	9	0	111	-	-

Na tabela 4.2 a entrada externa do neurônio (ou o terminal "entrada" da Unidade de Decisão) está representada pela palavra "Entrada", e a entrada "atual" (também da Unidade de Decisão) pela palavra "Atual".

Para facilitar o entendimento do que é feito em cada estado, na figura 4.9 é apresentado um diagrama simplificado do que está dentro da "Unidade de Execução" deste neurônio.

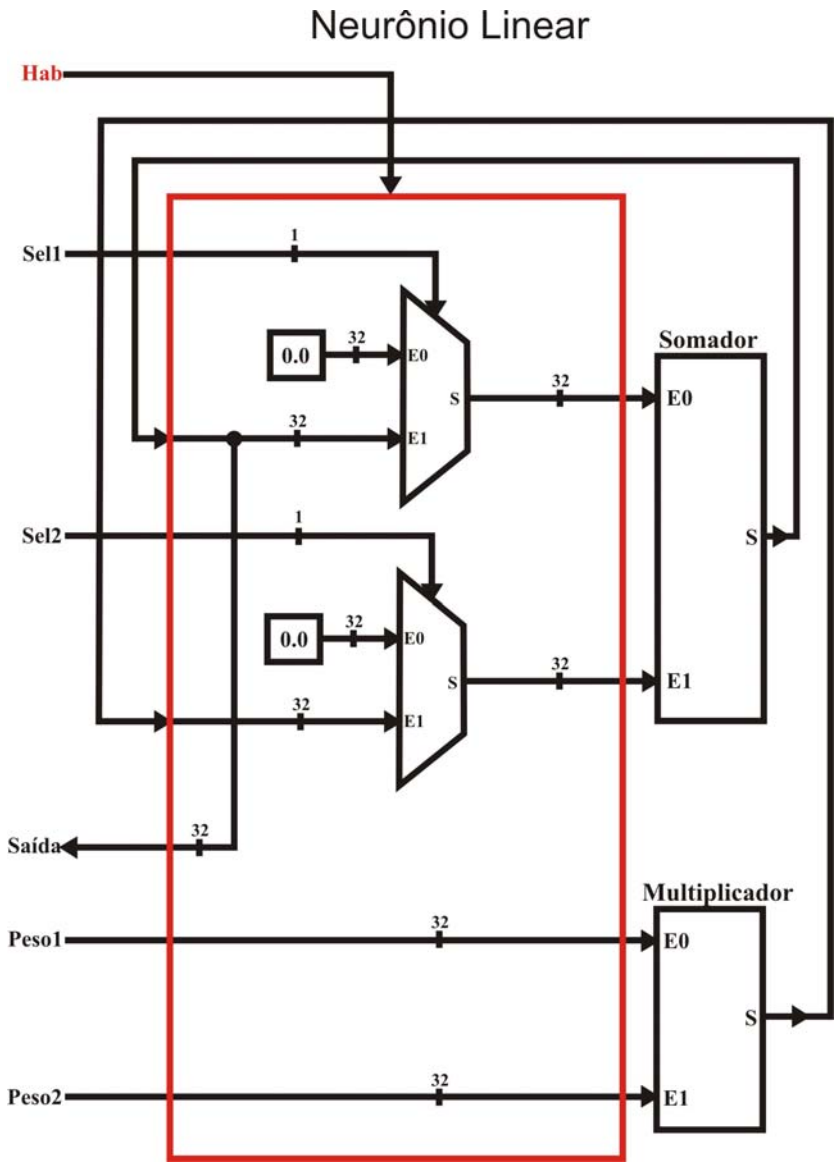


Figura 4. 9 - Interior da "Unidade de Execução" do neurônio com função de ativação linear.

Na figura 4.9, à esquerda, estão entradas de comando e de dados ("hab", "sel1", "sel2", "peso1" e "peso2"), além do terminal "saída" por onde sai o resultado final dos cálculos do neurônio. Estas entradas saem da "Unidade de Decisão" chegam na "Unidade de Execução" e representam os nomes das três últimas colunas na tabela 4.2. O terminal "saída" vai da "Unidade de Execução" e chega na entrada "atual" da "Unidade de Decisão". À direita na figura 4.9, estão as unidades de somador e de multiplicador em ponto flutuante. Internamente, ainda existem dois multiplexadores de duas entradas de 32 bits. Cada multiplexador age como uma chave seletora, enviando para sua saída "S" sua entrada "E0", caso sua entrada seletora "Sel" esteja no nível 0, e enviando a entrada "E1" se "Sel" for 1. As entradas "peso1" e "peso2" (operandos do multiplicador) entram direto nas entradas "E0" e "E1" do multiplicador. A entrada "hab"

representa uma porta de entrada de um sinal de controle geral que libera a execução da "Unidade de Execução" quando estiver no nível 1, e desabilita estando em nível 0. A entrada "sel1" é a entrada de seleção do multiplexador superior (cuja saída "S" é ligada a entrada "E0" do somador) e "sel2" é a entrada de seleção do multiplexador inferior (cuja saída "S" é ligada a entrada "E1" do somador).

Inicialmente, a máquina de estados do neurônio está no estado inicial, ou estado 0 na tabela 4.2. A "Lógica de próximo estado" no "Controlador de Camadas" da camada em que o neurônio está, encontra-se a espera de um sinal de início. Quando se dá este sinal de início, o "Controlador de camadas" envia o próximo estado (ou seja, o estado "1") para a entrada "contagem" do neurônio (figura 4.8). Os estados 1, 2 e 3 representam o processo de pré-normalização da entrada. Na pré-normalização a entrada do neurônio é multiplicada pela constante "XM1" e depois somada com a constante "XS1". Tais constantes de pré-normalização são valores específicos de aplicação.

No estado "1", o neurônio está na linha 2 da tabela 4.2, e envia o comando "100", ou "hab"=1, "sel1"=0, e "sel2"=0, para a "Unidade de Execução". Além do comando "100", são enviados valores de entrada para o multiplicador, "peso1"="XS1" e "peso2"="1,0". Como resultado final deste estado, "XS1" é multiplicado por "1,0" e aparece na entrada "E1" do multiplexador inferior.

Seguindo a execução da máquina de estados, começa o estado 2 (linha 3 da tabela 3.2). Neste estado, é enviado o comando "101" ou "hab"=1, "sel1"=0 e "sel2"=1. A entrada "peso1" recebe a constante "XM1" e em "peso2" entra o valor atual da entrada externa do neurônio. Duas operações ocorrem neste estado. Primeiro, o valor na entrada "E1" do multiplexador inferior (ou "XS1") é somado com "0,0", e o resultado dessa adição vai para a entrada "E1" do multiplexador superior. Como segunda operação, a entrada externa do neurônio é multiplicada pela constante "XM1" e fica disponível na entrada "E1" do multiplexador inferior.

No estado 3 o comando "111" é enviado e as entradas "peso1" e "peso2" não são utilizadas. Através do comando "111", a atribuição "sel1"=1 e "sel2"=1 seleciona a entrada "E1" do multiplexador superior (ou "XS1") conduzindo-o a entrada "E0" do somador. Ao mesmo tempo é selecionada a entrada "E1" do multiplexador inferior (ou "XM1*entrada") conduzindo-o a entrada "E1" do somador. O resultado final é o aparecimento, na entrada "E1" do multiplexador superior, do valor "XM1*entrada+XS1", ou seja, o valor pré-normalizado da entrada externa do neurônio.

Este valor pré-normalizado está também está na entrada "atual" da "Unidade de Decisão" (figura 4.8).

Nos estados 4, 5, 6, a entrada externa do neurônio é multiplicada por um peso "P1" e é adicionada a uma constante "BIAS". No estado 4 é enviado o comando "100", ou "hab"=1, "sel1"=0 e "sel2"=0. O operando "peso1" recebe a constante "P1" e "peso2" recebe o valor da entrada "atual" da "Unidade de Decisão". O comando "100" simplesmente habilita a "Unidade de Execução" e envia os valores "P1" e "Atual" para as entradas "E0" e "E1" do multiplicador, respectivamente. A saída do multiplicador no fim do estado 4 (ou o valor "P1*(XM1*entrada+XS1)") torna-se disponível na entrada "E1" do multiplexador inferior.

No estado 5 é enviado o comando "101", enquanto "peso1" recebe a constante "BIAS" e "peso2" recebe a constante "1,0". Os valores em "peso1" e "peso2" fazem com que surja na saída do multiplicador o valor "BIAS*1,0=BIAS". Esta saída do multiplicador também fica disponível na entrada "E1" do multiplexador inferior. Ao mesmo tempo, o comando "101" faz com que a entrada anterior "E1" do multiplicador de baixo (ou o valor "P1*(XM1*entrada+XS1)") seja adicionada com "0,0" e apareça na entrada "E1" do multiplexador superior.

O estado 6 simplesmente adiciona a entrada "E1" do multiplexador superior (ou o valor "P1*(XM1*entrada+XS1)") com a entrada "E1" do multiplexador inferior (ou o valor "BIAS"). O resultado final do estado 6 é o valor "P1*(XM1*entrada+XS1)+BIAS" na saída do somador (ou no terminal chamado "saída"). Este resultado final representa exatamente o valor "X", ou a entrada para a função de ativação linear do neurônio.

Nos estado 7, 8, 9, é feito o cálculo da função de ativação linear usando como entrada desta função, o valor "X" produzido pelo estado 6. O programa na tabela 4.2 considera "C1" o valor da inclinação da função linear de ativação, e C2 o coeficiente linear dessa função. No estado 7, volta a ser enviado o comando "100" para a "Unidade de Execução". O operando "peso1" recebe o valor "C1" enquanto "peso2" recebe o valor de "atual" (ou seja, "X"). Os valores em "peso1" e "peso2" geram na saída do multiplicador o valor "X*C1", que também fica disponível na entrada "E1" do multiplexador inferior.

O estado 8 envia o comando "101" e como operandos desse comando "peso1"="C0" e "peso2"="1,0". Os operandos "peso1" e "peso2" são multiplicados e geram o próprio valor "C0" na entrada "E1" do multiplexador inferior. O comando

"101" ainda faz com que o valor anterior da entrada "E1" do multiplexador de baixo (ou "X*C1") apareça na entrada "E1" do multiplexador superior.

Como finalização dos cálculos realizados pelo neurônio, no estado 9 é enviado o comando "111" para a "Unidade de Execução". Não são usados os operandos "peso1" e "peso2" neste estado. O comando "111" simplesmente adiciona as entradas "E1" do multiplexador inferior e superior, gerando na saída do somador o valor final "X*C1+C2". Este valor final, presente no terminal "saída" da "Unidade de Execução", representa a resposta final dos cálculos do neurônio para o ciclo realizado pela máquina de estados. Tal valor pode ser enviado para neurônios de camadas intermediárias ou ser considerado a saída final de uma rede neural.

O hardware descrito para um neurônio com função de ativação linear é adequado para operações realizadas por este tipo de neurônio, ou seja operações de adição e multiplicação. Porém, em redes neurais artificiais reais, as funções de ativação podem não ser tão simples, como é o caso da função sigmóide logística. Para neurônios com função de ativação deste tipo, um hardware composto por unidades básicas como somadores e multiplicadores precisam aplicar técnicas especiais de representação de funções ao realizarem seus cálculos. Os neurônios analisados nos próximos tópicos usam função de ativação sigmóide logística. Como será apresentado, suas arquiteturas procuram manter a simplicidade encontrada no neurônio de função linear, de forma que o que as diferencia são suas abordagens de representação da função sigmóide.

4.5.2.3 Neurônio digital com função de ativação sigmóide

O neurônio descrito nesta seção possui função de ativação sigmoidal. Trata-se neurônio mais complexo que o neurônio com função de ativação linear, por trabalhar com polinômios de maior grau para implementar em hardware a função sigmóide. Neste trabalho, a sigmóide é representada através do uso da técnica de interpolação "*spline*". A figura 4.10 mostra a função de ativação sigmoidal utilizada neste trabalho e representada por meio da interpolação "*spline*".

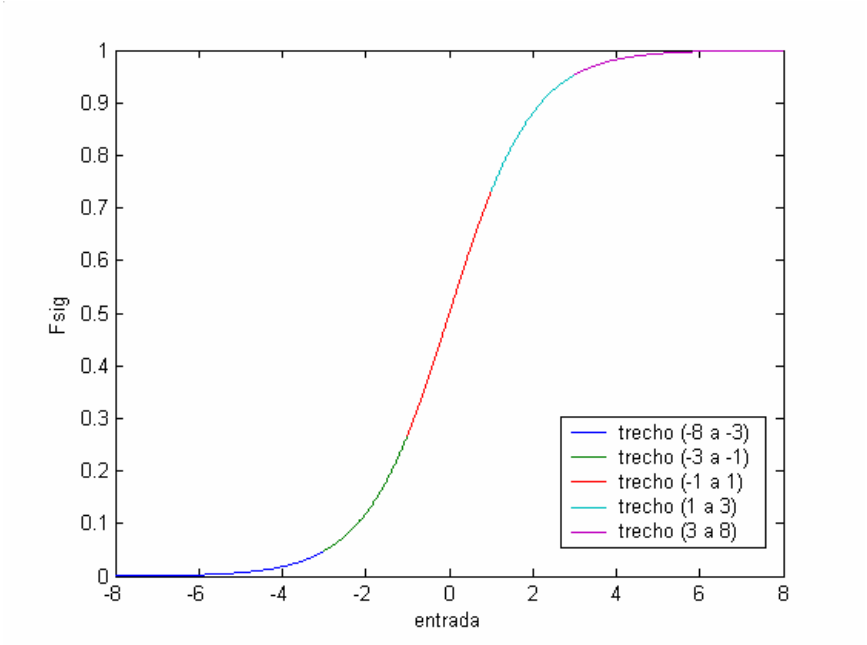


Figura 4. 10 - Representação da função sigmóide por "*Spline*".

Na figura 4.10, a função sigmóide foi dividida em 7 subintervalos, atribuindo-se a cada um deles um polinômio interpolador de grau igual 5. Cada trecho (ou intervalo de interpolação) é identificado por um segmento de curva de cor diferente. Na figura 4.10, são apresentados somente cinco trechos, já que os dois omitidos representam o comportamento da função em $-\infty$ e $+\infty$, ou seja, são exatamente as funções constante igual a 0 e 1, respectivamente.

A arquitetura do neurônio apresentado nesta seção também foi construída contendo duas unidades principais, uma "Unidade de Decisão" e uma "Unidade de Execução". Um diagrama de blocos geral deste neurônio pode ser visto na figura 4.11.

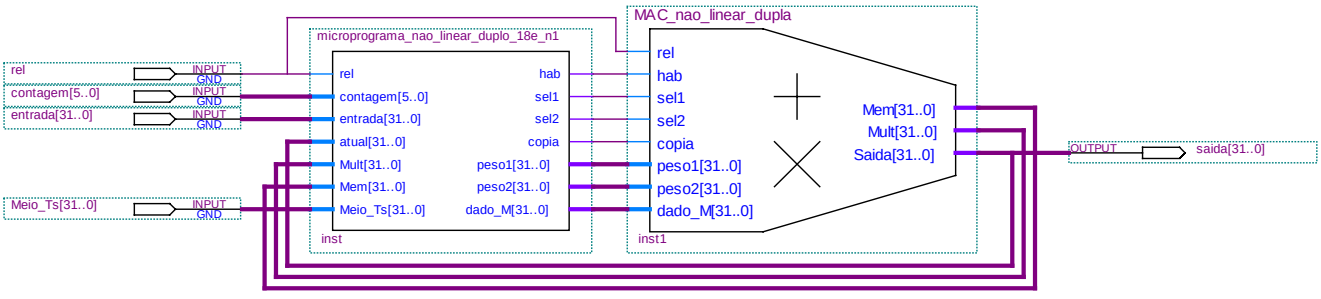


Figura 4. 11 - Neurônio digital com função de ativação sigmóide.

Os terminais de entrada e de saída deste neurônio são os mesmos encontrados no neurônio com função de ativação linear, "rel", "contagem", "entrada" e "saída". Na figura 4.11, a "Unidade de Decisão" é o bloco da esquerda e a "Unidade de Execução" o

da direita. Entre as duas unidades estão os mesmos sinais de controle enviados para a "Unidade de Execução" já mencionados para o neurônio com função linear, com a diferença dos sinais "cópia" e "Dado_M". O sinal "cópia" serve como sinal de gatilho para gravação, em registrador na "Unidade de Execução", de um valor de 32 bits enviado pelo sinal "Dado_M". A "Unidade de Execução" deste neurônio ainda apresenta outra diferença, ela possui três terminais de saída de 32 bits "Mem", "Mult" e "Saída". O terminal de saída "Mem" representa a saída do registrador interno da "Unidade de Execução" e o terminal "Mult" vem da saída do multiplicador interno desta unidade. Detalhes da "Unidade de Execução" podem ser vistos na figura 4.12.

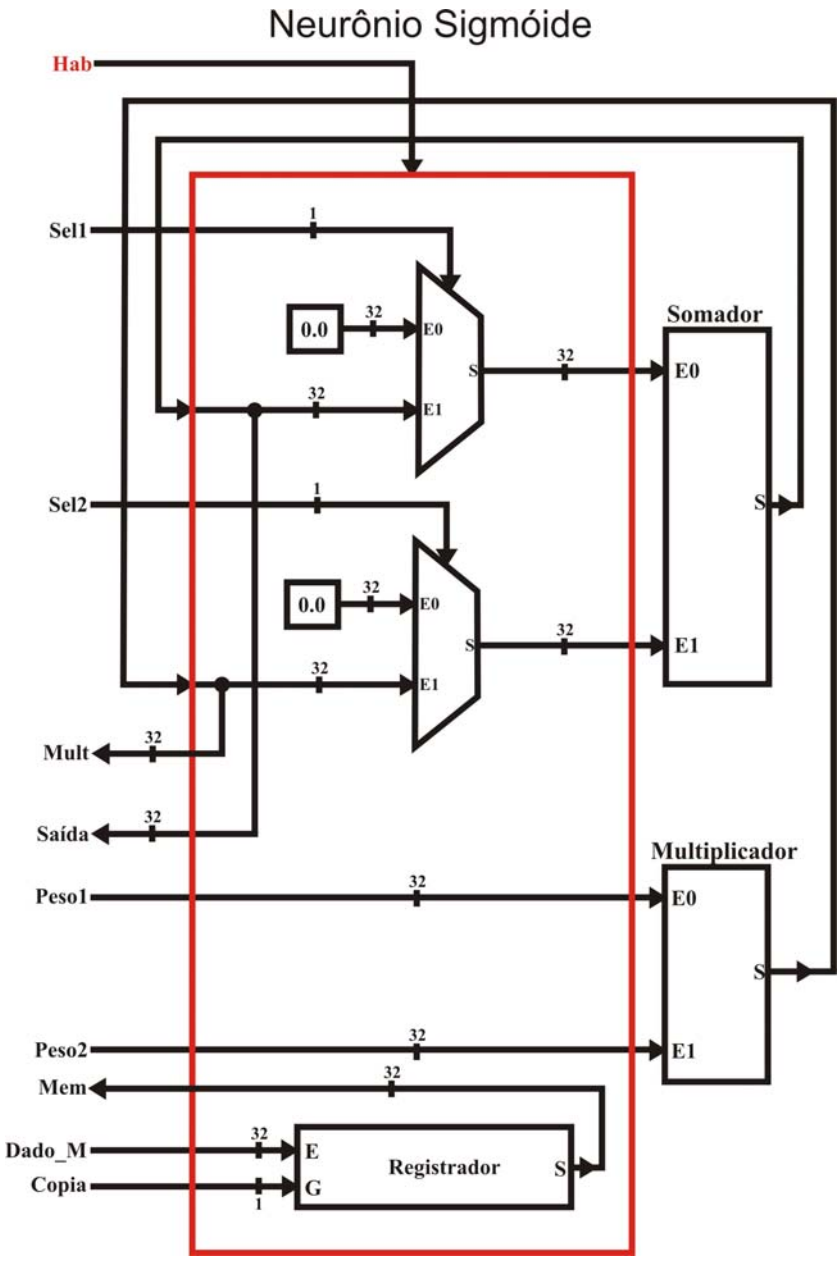


Figura 4. 12 - Diagrama da "Unidade de Execução" do neurônio com função sigmóide.

O terminal de saída "Mult" serve para que a "Unidade de Decisão" tenha mais facilidade para obter resultados de multiplicação gerados pela "Unidade de Decisão", quando se está trabalhando com operações como potências e cálculos polinomiais. O registrador interno na "Unidade de Execução" representa uma unidade para armazenamento de dados temporários gerados pelos cálculos durante execução. Dessa forma este registrador pode ser imaginado com uma unidade periférica de memória local.

Semelhante ao neurônio com função de ativação linear, existe aqui um programa de controle baseado em uma máquina de estados dentro da "Unidade de Decisão". Ou seja, a "Unidade de Decisão" ainda representa um bloco de "Lógica de Saída" de uma máquina de estados. Portanto, ainda aqui deve existir uma unidade de "Controlador de Camada" com a "Lógica de Próximo estado" e a "Memória de estado" embutidas. Continua sendo este "Controlador de Camada" que envia cada valor de estado atual para a entrada "contagem" do neurônio durante as transições entre estados. Um melhor entendimento do funcionamento dessa máquina de estados é possível através do diagrama representado pela tabela 4.3.

Tabela 4. 3 - Microprograma da "Unidade de Decisão" do neurônio com função sigmóide.

rel	Estado atual	Próximo estado	Hab,sel1,sel2,copia	Dado_M	Peso 1	Peso 2	X
↑	0	1	0000	-	-	-	-
↑	1	2	1000	-	XS1	1.0	-
↑	2	3	1010	-	XM1	Entrada	-
↑	3	4	1110	-	-	-	-
↑	4	5	1000	-	P1	Atual	-
↑	5	6	1010	-	BIAS	1.0	-
↑	6	7	1110	-	-	-	Atual
↑	7	8	0000	-	-	-	-
↑	8	9	1001	Atual	C0	1.0	-
↑	9	10	1010	-	C1	Mem	-
↑	10	11	1110	-	X	Mem	-
↑	11	12	1101	Mult	C2	Mult	-
↑	13	14	1110	-	X	Mem	-
↑	13	14	1101	Mult	C3	Mult	-
↑	14	15	1110	-	X	Mem	-
↑	15	16	1101	Mult	C4	Mult	-
↑	16	17	1110	-	X	Mem	-
↑	17	18	1101	Mult	C5	Mult	-
↑	18	0	1110	-	-	-	-

Na tabela 4.3, nas colunas 2 e 3 (segunda e terceira da esquerda para direita) estão os valores de estado atual e próximo estado respectivamente. Na coluna 4 estão os bits representando a operação a ser executada pela "Unidade de Execução", e nas colunas 5, 6, 7, estão os operandos dessa operação ("Dado_M", "peso1" e "peso2"). Na coluna 8 está representada a variável de 32 bits "X", interna a "Unidade de Decisão". As palavras na tabela 4.3, "Entrada", "Atual", "Mult", "Mem" representam os terminais de entrada com igual nome da "Unidade de Decisão" (figura 4.11). Já as palavras "XS1", "XM1", "P1", "BIAS", "C0 a C5", representam valores constantes usados nos cálculos do neurônio.

Os estados 0 a 6, realizam o mesmo tipo de função que os estados de 0 a 6 do neurônio com função de ativação linear, ou seja, pré-normalização da entrada, multiplicação desse valor pré-normalizado por "P1", e posterior soma com "BIAS". O resultado final dessa computação fica disponível em "atual" para ser armazenado na variável interna X no próximo estado (estado 7).

O estado 7 representa um estado de verificação de "rumo" a ser tomado. Como no momento já está na variável "X" um valor que representa a entrada para o cálculo da função de ativação sigmóide, é preciso escolher qual trecho de *spline* usar para este cálculo. Esta escolha de trecho (ou de rumo) é feita por uma unidade interna na "Unidade de Decisão" que realiza seu trabalho durante o transcorrer do estado 7. Definir um rumo pode ser entendido como determinar qual conjunto de coeficientes polinomiais (C0,C1,C2,C3,C4,C5) deve ser utilizado nos cálculos de interpolação que se seguem. Como foram escolhidos seis nós de interpolação, existem 7 trechos de *spline* representando a sigmóide, ou seja, 7 conjuntos possíveis de coeficiente polinomiais.

Definidos os valores dos coeficientes "C0" a "C5", ocorre o cálculo da função sigmóide nos estados 8 a 18. O resultado da interpolação realizada nos estados 8 a 18 pode ser representado pela equação 4.3:

$$F_{sigmóide}(X) = C0 + C1 \cdot X + C2 \cdot X^2 + C3 \cdot X^3 + C4 \cdot X^4 + C5 \cdot X^5 \qquad (4.3)$$

Finalizado o estado 18, a máquina de estados volta para o estado inicial 0, ficando a espera de um novo comando de início. Para valores de "X" maiores que 8 a função sigmóide aproxima-se da função constante 1 e, para valores "X" menores que -8 da função aproxima-se da constante 0. Este fato permite igualar o polinômio interpolador a 0 para "X" menor que -8, e igualar a 1 para "X" maior que 8.

Consequentemente o cálculo da função sigmóide é mais simples fora do intervalo [-8,8], podendo ser feito mais rapidamente.

As arquiteturas lineares e sigmoidais para neurônios digitais apresentadas até aqui representam os modelos básicos utilizados na criação dos neurônios da versão final das redes neurais do controlador do motor de indução. Nesta versão final das redes existem também expansões dos modelos básicos, criadas para permitir reutilização de capacidade de processamento em momentos ociosos de alguns neurônios.

Uma destas expansões já foi mencionada, anteriormente, ao ser apresentado o "Neurônio com dupla função", um dos neurônios da primeira camada da rede RNN-MLP. Este neurônio especial possui uma arquitetura muito parecida com a de um neurônio básico com função de ativação sigmóide. A diferença está em sua "Unidade de Decisão" ampliada, composta por dois blocos de microprograma diferentes compartilhando o uso de uma mesma "Unidade de Execução". Cada um destes blocos de microprograma executa em um momento diferente, evitando-se, assim, conflitos de hardware (*hazards*). Um deles realiza cálculos de transformação trifásica para eixos dq (equações 2.1 a 2.3), enquanto o outro realiza os cálculos normais de um neurônio de entrada de rede (acumulação de entradas ponderadas e aplicação desta acumulação em uma função de ativação).

Um outro exemplo de expansão dos modelos básicos de neurônios apresentados até o momento, surge na estrutura interna da rede RNN (rede recorrente pertencente ao bloco "Estimador de Fluxo"). Para permitir recorrência, ou seja, que esta rede utilize os próprios resultados produzidos na execução anterior para produzir novas saídas, a rede RNN deve memorizar um conjunto de variáveis a cada uma de suas iterações de execução.

Ainda, outros exemplos de expansão de arquitetura de neurônios serão apresentados nas próximas seções. Este é o momento de iniciar a apresentação de uma plataforma digital que também está envolvida no processamento de sinais do controlador do motor de indução deste trabalho. Esta plataforma, o DSP TMS320C6711, é apresentada em detalhes no apêndice C deste trabalho. A seção a seguir apresenta o primeiro bloco construído sob esta plataforma, ou seja, o bloco "Estimador de Sinais".

4.6 Hardware do bloco Estimador de Sinais

Conforme já apresentado no capítulo 2, o bloco "Estimador de Sinais" compreende um conjunto de equações envolvendo funções trigonométricas e de radiciação. Neste trabalho, o "Estimador de Sinais" não foi modelado baseado em redes neurais. Na prática este módulo está descrito como um código de programação em linguagem C para execução sobre a plataforma de hardware do processador de sinais TMS320C6711 da *Texas Instruments*. A seção a seguir descreve mais detalhadamente este código de programação.

4.6.1 Código em C do "Estimador de Sinais"

O DSP do protótipo do controlador do motor de indução age como um ponto de convergência de informações durante o funcionamento do sistema como um todo. O DSP gerencia a "Interface com Usuário", ativa e troca informações com as redes neurais dentro do FPGA e ainda realiza os cálculos dos blocos "Estimador de Sinais" e "Controlador" (seção a seguir). Todas estas operações são resultantes da execução de rotinas em C apresentadas nesta seção e na seção a seguir.

Um único arquivo, chamado "programa_usuario.c", contém linhas de código representando os cálculos efetuados pelo bloco "Estimador de Sinais" e pelo bloco "Controlador". Durante o funcionamento normal do protótipo do controlador do motor, um micro sistema operacional da *Texas Instruments*, chamado "DSP/BIOS", executa no DSP rotinas próprias de gerenciamento e as rotinas no arquivo "programa_usuario.c". Neste arquivo "programa_usuario.c" os cálculos dos blocos "Estimador de Sinais" e "Controlador" fazem parte da função "Caminho_retroacao", uma função de atendimento de interrupções do contador "*Timer 0*" do DSP. A cada 100μs (valor de período de amostragem T_S programado no contador "*Timer 0*") a função "Caminho_retroacao" executa uma iteração. No início desta iteração a função "Caminho_retroacao", recebe entradas da "Interface com Usuário" e realiza os cálculos do "Estimador de Sinais" a partir de dados gerados anteriormente pela rede RNN-MLP no FPGA. Finalizando a iteração, ocorrem os cálculos do bloco "Controlador" utilizando os resultados do "Estimador de Sinais" para gerar novos comandos para o bloco "Controle PWM Space-Vector Baseado em Redes Neurais" (bloco no FPGA descrito no capítulo 3, seção 3.2).

Para execução mais rápida dos cálculos dentro da função “Caminho_retroacao” foram utilizadas funções matemáticas da biblioteca “*FastRTS67x.lib*” disponibilizada pela *Texas Instruments*. O código de programação representando especificamente o bloco “Estimador de Sinais” é apresentado na figura 4.13. Declarações das variáveis e constantes utilizadas no código estão no apêndice B deste trabalho.

```
//-------------------------------------------------------------------------------------------------------------------------
//  Fase de cálculos de estimação dos sinais recebidos da rede recorrente
//-------------------------------------------------------------------------------------------------------------------------
1       //Lendo valores produzidos pelas redes neurais no FPGA
2       Yqss_int=Yqss;//Recebe do FPGA o fluxo Yqss
3       Ydss_int=Ydss;//Recebe do FPGA o fluxo Ydss
4       iqss_linha=iqssl;//Recebe do FPGA a corrente iqssl
5       idss_linha=idssl;//Recebe do FPGA a corrente idssl
6       Vqss_int=Vqssll;//Recebe do FPGA a tensão Vqssll (defasada)
7       Vdss_int=Vdssll;//Recebe do FPGA a tensão Vdssll (defasada)
8       //Realizando calculos de Estimacao
9       Ys2=Yqss_int*Yqss_int+Ydss_int*Ydss_int;
10      Omega_int=_divf(Vqss_int*Ydss_int-Vdss_int*Yqss_int,Ys2);
11      Omega=Omega_int*xm3+xs3;//Pré-normalização da entrada da rede recorrente.
12      controle_recorrente=1;//Ativa rede recorrente.
13      Ys_calc=sqrtf(Ys2);
14      senTETAe=_divf(Yqss_int,Ys_calc);
15      cosTETAe=_divf(Ydss_int,Ys_calc);
16      ids=iqss_linha*cosTETAe-idss_linha*senTETAe;
17      iqs=iqss_linha*senTETAe+idss_linha*cosTETAe;
18      idq=_divf((sigma*Ls*iqs*iqs),(Ydss_int-(sigma*Ls*ids)));
19      Yds=Yqss_int*cosTETAe-Ydss_int*senTETAe;
20      Yqs=Yqss_int*senTETAe+Ydss_int*cosTETAe;
21      Te_calc=0.75*P*(Yds*iqs-Yqs*ids);
```

Figura 4. 13 - Linhas de código representando o bloco “Estimador de Sinais”.

A rotina apresentada na figura 4.13 é iniciada pelas linhas 2 a 7, recebendo-se os valores produzidos pela última execução da rede RNN-MLP no FPGA (bloco “Estimador de Fluxo”). As linhas restantes (9 a 21) simplesmente utilizam estes valores e executam os cálculos referentes ao bloco “Estimador de Sinais” (equações 2.8 a 2.16). Na linha 12, o comando “controle_recorrente=1” não atribui um valor a uma variável, na realidade ele ativa a entrada do “Controlador de Camada 01” da rede RNN-MLP, iniciando um novo ciclo de trabalho do bloco “Estimador de Fluxo”. Não está previsto no código apresentado na figura 4.13 qualquer método de leitura de sinal de resposta

vindo do FPGA que indique o momento em que a rede RNN-MLP termina seus cálculos. Porém, uma iteração do código da figura 4.13 não envia uma nova ordem de início para a rede RNN-MLP antes que ela tenha terminado seus cálculos. Isto ocorre, pois, esta rede termina seus cálculos em um tempo menor que 100μs (período entre iterações do código da figura 4.13, definido em [25] e [26]).

Após cada iteração, as respostas produzidas pelo DSP executando os cálculos do bloco "Estimador de Sinais" são repassadas para o bloco descrito na seção a seguir, o "Controlador".

4.7 Hardware do bloco "Controlador"

Como já descrito, o bloco "Controlador" representa um local do sistema em que estimações são comparadas com valores desejados, sendo os erros destas comparações inseridos em controladores PI (proporcional e integral) para geração de saídas de atuação. O hardware utilizado na implementação dos nós de comparação e dos PIs do bloco "Controlador", continua sendo a arquitetura de processador de sinais apresentada na seção 4.6.1 (descrição do DSP TMS320C6711).

Conforme já mencionado, o bloco "Controlador" está representado no protótipo de controlador do motor deste trabalho como um código de programação escrito em linguagem C. A seção a seguir tem mais informações a respeito deste código.

4.7.1 Código em C do bloco "Controlador"

Conforme já mencionado, as linhas de comando representando o bloco "Controlador" estão dentro do arquivo "programa_usuario.c", juntamente com o código representando o bloco "Estimador de Sinais". A estrutura interna do bloco "Controlador", mostrada no capítulo 2, sugere que este módulo do sistema deve ter uma interação direta tanto com a "Interface com Usuário" (fora do DSP) quanto com o bloco "Estimador de Sinais". A "Interface com Usuário" fornece ao bloco "Controlador" valores estabelecidos pelo usuário referentes a torque (T_e^*) e fluxo de estator ($\hat{\psi}_s^*$). O bloco "Estimador de Sinais" fornece valores estimados de corrente (i_{qs}, i_{ds}, i_{dq}), de

torque (T_e) e de fluxo de estator ($\hat{\psi}_s$). Esta interação do bloco "Controlador" com os dois módulos mencionados está explícita na figura 4.14.

A figura 4.14 apresenta o código de programação para realização dos cálculos do bloco "Controlador". As linhas 1 a 7 representam a recepção de valores atualizados vindos das variáveis torque "Te" e fluxo "Ys", variáveis que representam endereços de memória dentro da faixa representando o hardware da "Interface com Usuário".

```
//-------------------------------------------------------------------------------------------------------------------------------
//  Fase de cálculos da malha de controladores PI em série
//-------------------------------------------------------------------------------------------------------------------------------
1       if(atualizar==1)//Verifica se existem atualizações para as variáveis de entrada
2       {
3         //Associando valores as variáveis
4         Te_star=Te;
5         Ys_star=Ys;
6         Meio_Ts=Ts*0.5;//Envia um novo valor de período para o FPGA
7       }
8       //Definindo saidas dos PIs em série
9       Vqs_star=Te_star+Te_calc+KI1+iqs+KI2;
10      Vds_star=Ys_star+Ys_calc+KI3+idq+ids+KI4;
11      //Vector Rotator. Definindo entradas da rede MLP Espace Vector
12      V_star=sqrtf(Vqs_star*Vqs_star+Vds_star*Vds_star);//Envia entrada de módulo
13      modulo=V_star*xm1+xs1;//Pré-normalização da entrada de módulo da rede SV.
14      TETAe=asinf(senTETAe);
15      TETAe_star=TETAe+atanf(_divf(Vqs_star,Vds_star));
16      TETAe_linha=TETAe_star-floorf(0.1591549*TETAe)*6.2831853;//Envia entrada de ângulo
17      ângulo=TETAe_linha*xm2+xs2;//Pré-normalização da entrada de módulo da rede SV.
18      //Ativando a rede PWM Space-Vector
19      controle_SV=1;
```

Figura 4. 14 - Código de programação do bloco "Controlador".

As linhas 9 e 10 representam os cálculos realizados pelos quatro controladores PI, ajustados através de suas constantes predefinidas "KI1", "KI2", "KI3", "KI4". As linhas 12 a 17 representam os cálculos do subsistema "Vector-Rotator", pertencente ao bloco "Controlador". Este subsistema produz como resultado, na linha 12, um valor de módulo ($\hat{V}^*$) e na linha 16 um valor de ângulo ($\theta_e^{*'}$), juntos eles representam o vetor de referência de modulação PWM. Tanto o módulo quanto o ângulo são pré-normalizados antes de serem aplicados (linhas 13 e 17) ao hardware do bloco "Controle PWM Space-vector Baseado em Redes Neurais" (descrito na seção a seguir). A ordem de início de

operação do bloco “Controle PWM Space-vector Baseado em Redes Neurais” somente é emitida pelo bloco “Controlador” durante a execução do comando “controle_SV=1” (linha 19 do código da figura 4.14).

Todas as estimações e comparações com valores desejados pelo usuário, culminam na geração de sinais de comando aplicados ao bloco “Controle PWM Space-vector Baseado em Redes Neurais” no FPGA. Este bloco é descrito em maiores detalhes na seção a seguir.

4.8 Hardware do bloco "Controle PWM Space-Vector Baseado em Redes Neurais"

O hardware do bloco “Controle PWM Space-Vector Baseado em Redes Neurais” é o último módulo do “Subsistema de Processamento de Sinais” a atuar durante um ciclo de trabalho do protótipo de controlador do motor de indução.

Conforme já discutido nos capítulo 2 e 3, as redes neurais no FPGA pertencentes ao “Controle PWM Space-Vector Baseado em Redes Neurais” recebem em suas entradas as saídas normalizadas de módulo ($\hat{V}^*$) e ângulo ($\theta_e^{*'}$), geradas pelo módulo “Controlador” (seção 4.7). Como resposta, estas redes produzem seis sinais de modulação PWM, para controle das chaves do “Inversor Baseado em IGBT” (Subsistema de Circuito de Potência).

No FPGA, o hardware do bloco "Controle PWM Space-Vector Baseado em Redes Neurais" está implementado como três subsistemas interligados: a “Rede de Módulo”, a “Rede de ângulo e de Cálculo de Ton” e o “Contador Trifásico”. As seções a seguir detalham estes três módulos, iniciando pela “Rede de módulo”.

4.8.1 Rede de módulo

A implementação em hardware da “Rede de módulo” conta com quatro processadores digitais (neurônios) interligados, formando duas camadas de rede neural. Todos estes processadores seguem o mesmo modelo básico de arquitetura mostrado para os neurônios com função de ativação sigmoidal na seção 4.5.2.3. A primeira diferença entre os neurônios da “Rede de módulo” e os neurônios com a arquitetura básica mencionada, está no fato dos neurônios da camada de entrada da “Rede de

módulo" não pré-normalizarem a entrada de módulo ($\hat{V}^*$) recebida do bloco "Controlador" no DSP. A segunda diferença é devida ao fato do neurônio da segunda camada possuir três entradas e produzir como saída final um valor que já sofreu pós-normalização. Um diagrama mostrando a estrutura de hardware da "Rede de módulo" é mostrado na figura 4.15.

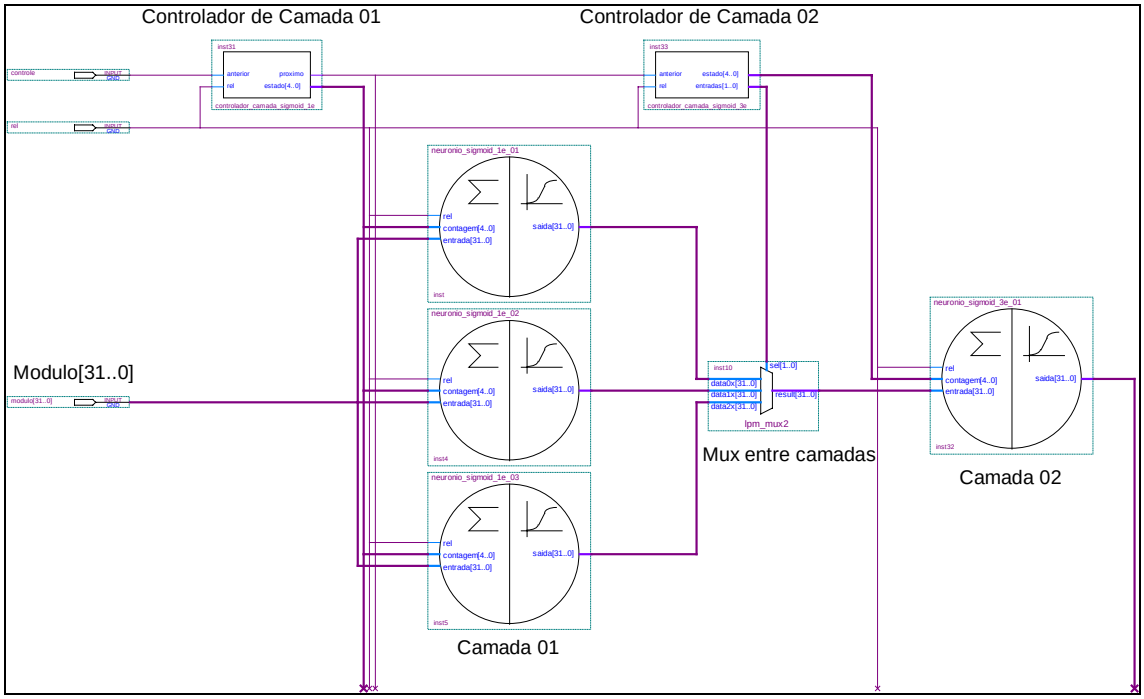


Figura 4. 15 - Estrutura da "Rede de módulo" no FPGA.

De forma análoga a estrutura da rede RNN-MLP do bloco "Estimador de Fluxo", existem dois controladores de camada compondo a "Rede de módulo", indicados na figura como: "Controlador de Camada 01" e "Controlador de Camada 02". Num ciclo completo de cálculos a camada de entrada da rede passa por 15 estados, enquanto a camada de saída passa por 20 estados. Para direcionar as saídas dos neurônios da camada de entrada para o neurônio da camada de saída existe o bloco "Mux entre camadas", gerenciado pelo "Controlador de Camada 02". Como saída final a "Rede de módulo" gera o valor $f(V^*)$.

A saída $f(V^*)$ é utilizada, posteriormente, nos cálculos dos tempos "T_{A-ON}", "T_{B-ON}" e "T_{C-ON}", realizados pela "Rede de ângulo e de Cálculo de Ton", descrita na seção a seguir.

4.8.2 Módulo “Rede de ângulo e de Cálculo de Ton”

A “Rede de ângulo e de Cálculo de Ton” possui uma estrutura mais complexa que a descrita para a “Rede de módulo”. Na “Rede de ângulo e de Cálculo de Ton” até 21 processadores independentes (neurônios) trabalham na geração dos seis sinais de modulação de saída da rede. A estrutura inteira da rede conta com um “Controlador de Camada 01” comandando 18 neurônios de entrada, um “Controlador de Camada 02” comandando 3 neurônios da camada de saída, e um bloco “Limitador de Ton entre 0 e Ts/2”. A figura 4.16 mostra parte da “Rede de ângulo e de Cálculo de Ton” no FPGA.

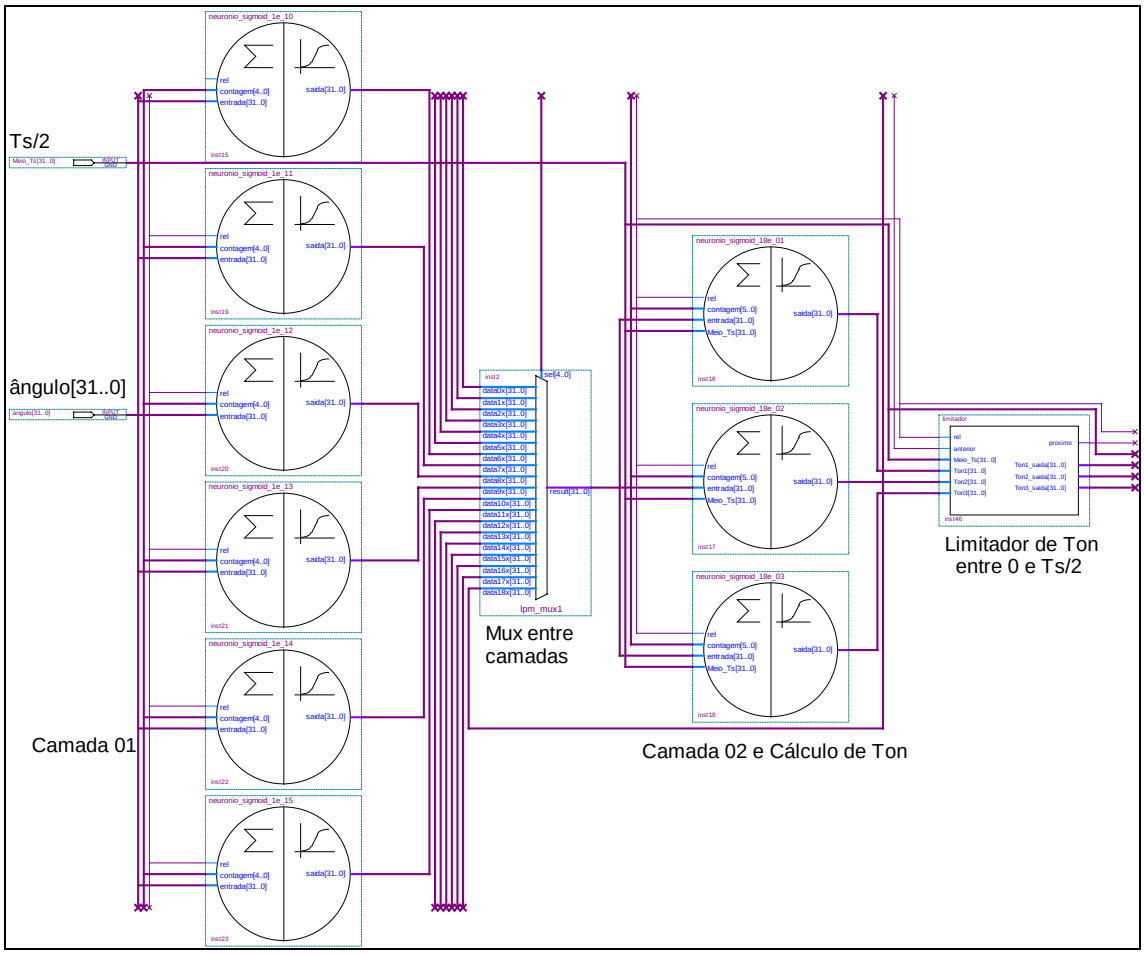


Figura 4. 16 - Hardware da “Rede de ângulo e de Cálculo de Ton”.

Para que se tenha maior resolução de detalhes na figura 4.16, são mostrados apenas 6 dos 18 neurônios da camada de entrada e não são apresentados o “Controlador de Camada 01” e o “Controlador de Camada 02”. Um “Mux entre camadas” extendido, formado por 18 entradas e uma saída, interliga a camada de entrada com de saída. Este

multiplexador recebe um sinal de seleção de entradas vindo do "Controlador de Camada 02". Durante um ciclo de trabalho completo da "Rede de ângulo e de Cálculo de Ton", os neurônios da camada de entrada passam por 15 estados, enquanto os neurônios da camada de saída passam por 39 estados. O grande número de estados de cálculo para a camada de saída deve-se ao fato desta camada também realizar operações relacionadas com o "Cálculo de Ton".

Como o nome "Rede de ângulo e de Cálculo de Ton" indica, este bloco é constituído por duas partes interagindo diretamente, uma "Rede de ângulo" e um módulo de "Cálculo de Ton".

A "Rede de ângulo" sempre começa trabalhando. Inicialmente, ela recebe do bloco "Controlador" (seção 4.7) um novo valor para a entrada "ângulo[31..0]" (figura 4.17). Depois de realizar todos os cálculos necessários, a "Rede de ângulo" gera três resultados: "$g_A(\alpha^*)$", "$g_B(\alpha^*)$", e "$g_C(\alpha^*)$". A partir destes resultados, os três neurônios da camada de saída da própria "Rede de ângulo" iniciam um novo trabalho, os cálculos relacionados somente com a porção de "Cálculo de Ton" do bloco.

Para a realização das operações do "Cálculo de Ton", estes neurônios ainda utilizam duas entradas adicionais. A primeira, a entrada "Ts/2[31..0]" na figura 4.17, é fornecida pelo DSP e representa a metade do período de amostragem da modulação "Space-Vector". A segunda entrada consiste na saída da "Rede de módulo" (descrita na seção 4.8.1), ou seja, "$f(V^*)$".

Após alguns cálculos adicionais os três neurônios produzem três valores de tempo (em segundos): "T_{A-ON}" (equação 2.19), "T_{B-ON}" (equação 2.20) e "T_{C-ON}" (equação 2.21). Antes de serem aplicados em contadores para geração de sinais PWM, os valores de tempo "T_{A-ON}", "T_{B-ON}" e "T_{C-ON}" passam por um subsistema chamado "Limitador de Ton entre 0 e Ts/2". Conforme já indicado pelo nome, este subsistema impede que valores de tempo de "T_{ON}" fora da faixa 0 a "Ts/2" sejam aplicados aos contadores de geração de sinais PWM. Tempos negativos são truncados para 0 e tempos maiores que "Ts/2" são truncados para "Ts/2". Valores de tempo fora da faixa de 0 a "Ts/2" surgem no momento em que o controlador de motor de indução opera em regiões de sobremodulação conforme já mencionado no capítulo 2.

Estando devidamente limitados, os valores de tempo saem do subsistema "Limitador de Ton entre 0 e Ts/2" chegando ao bloco "Contador Trifásico". A operação do bloco "Contador Trifásico" é detalhada na seção a seguir.

4.8.3 Módulo “Contador Trifásico”

O hardware do módulo “Contador Trifásico” representa o circuito de tradução dos valores de tempo gerados pelo bloco “Rede de ângulo e de Cálculo de Ton” para comprimentos de pulso de seis ondas quadradas, a serem aplicadas nas chaves de potência do “Inversor Baseado em IGBT” (pertencente ao “Subsistema de Circuito de Potência”). Um diagrama de entradas e saídas do “Contador Trifásico” é mostrado na figura 4.17.

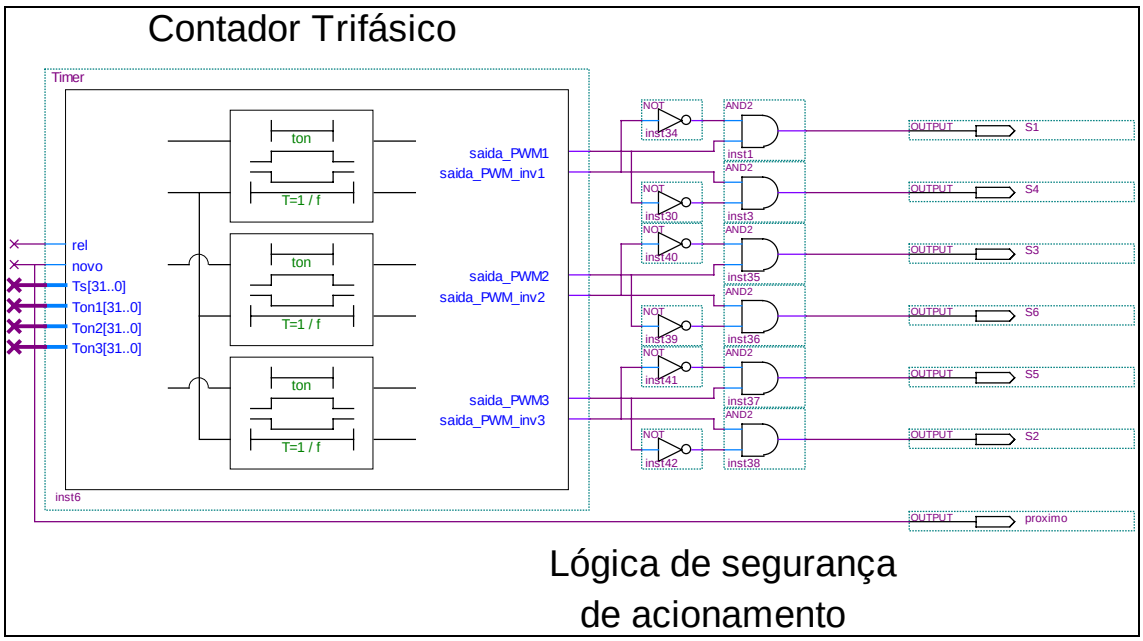


Figura 4. 17 - Diagrama do sistema “Contador Trifásico”.

A figura 4.17 mostra o “Contador Trifásico” ligado a um circuito de saída de “Lógica de segurança de acionamento”. Durante operação normal, o “Contador Trifásico” recebe valores atualizados de “T_{A-ON}”, “T_{B-ON}” e “T_{C-ON}” (vindos do bloco “Rede de ângulo e de Cálculo de Ton”, descrito na seção 4.8.2). Estes tempos são então comparados com uma onda triangular de referência gerada internamente. Esta onda triangular é gerada a partir de uma contagem “crescente/decrescente” produzida por um acumulador dentro do bloco “Contador Trifásico”. Este acumulador trabalha com uma resolução de contagem igual a um passo (incremento ou decremento) a cada 0,05µs. A onda triangular final é simétrica e possui uma freqüência igual a “1/Ts”. O circuito de “Lógica de segurança de acionamento” simplesmente assegura que cada saída PWM e a

saída representando seu par complementado não ativem (fiquem em nível 1) em instantes iguais, caso contrário, curto-circuitaria o ramo em questão.

Do "Contador Trifásico", os sinais PWM gerados saem do FPGA (na placa do "Subsistema Digital") e vão para circuitos opto-acopladores na placa de circuito do "Subsistema Analógico". Destes otpo-acopladores os sinais PWM chegam aos circuitos de comando dos IGBTs, pertencentes ao "Inversor Baseado em IGBT".

A cada ciclo de trabalho completo, o hardware do controlador do motor de indução atualiza os comprimentos de pulso dos sinais PWM enviados ao "Inversor Baseado em IGBT". Isto causa modificações de grandezas como torque e/ou velocidade rotacional do motor controlado.

4.9 Considerações finais

O principal motivo de se ter implementado o "Subsistema de Processamento de Sinais" utilizando também um DSP, e não somente um FPGA, é a limitação de espaço encontrada em um FPGA sem que haja um custo elevado. Além disso, a característica não redutível em simples multiplicações e acumulações das operações matemáticas executadas pelos blocos "Estimador de Sinais" e "Controlador" favorece a escolha de um DSP, em vez de um outro FPGA para interação com o FPGA "Stratix 2" já existente. Esta característica pode ser observada pela presença tanto de operações elementares (como adições, subtrações e multiplicações), como de operações transcendentais (como "seno", "arco-seno", raiz quadrada) nos cálculos do "Estimador de Sinais" e do "Controlador".

A escolha de um DSP permite o uso de bibliotecas matemáticas otimizadas para a arquitetura do próprio DSP. Isto economiza tempo de desenvolvimento comparado ao esforço que seria necessário para se criar um suporte dessas funções para uma arquitetura voltada para implementação de redes neurais em FPGA.

Diferentemente, uma rede neural possui modularidade por possuir um neurônio como bloco construtivo básico e, desta forma, a própria rede representa apenas uma replicação deste bloco construtivo. O maior esforço de desenvolvimento neste caso consiste na definição de uma estrutura de processamento representativa de um neurônio, esforço redutível ao uso de unidades MAC (multiplicador-acumulador). Esta modularidade unida à necessidade de rápido processamento paralelo, nortearam a escolha de um dispositivo FPGA para a implementação das redes neurais deste trabalho.

Tendo sido finalizada a descrição de hardware do protótipo do controlador de motor de indução, é chegado o momento de apresentar e discutir os resultados práticos obtidos. A seguir, o capítulo 5 apresenta estes resultados, além de fotos dos circuitos e da bancada utilizada durante os testes do protótipo.

CAPÍTULO 5 - PROTÓTIPO E RESULTADOS

5.1 Introdução

Como contribuição experimental deste trabalho, neste capítulo são apresentados resultados de simulações e de testes práticos realizados com o protótipo do controlador desenvolvido para o motor de indução. Além disso, estão também neste capítulo fotos das placas de circuito impresso do protótipo, do motor de indução e do inversor utilizados. Simulações do hardware no FPGA foram feitas com auxílio do gerador de formas de onda, presente no programa "Quartus 2 v6.0" da Altera. Formas de onda do protótipo representando variáveis de 32 bits do sistema foram obtidas com auxílio de ferramentas do programa "Code Composer Studio" para DSPs da família TMS320C6000, da *Texas Instruments*. Ondas PWM, tensões e correntes do sistema, foram obtidas utilizando-se um osciloscópio, as quais, foram armazenadas digitalmente como figuras para inclusão neste capítulo.

Este capítulo inicia apresentando especificações técnicas dos equipamentos utilizados, e então, são apresentadas fotos do sistema e as formas de onda de simulação e dos testes experimentais.

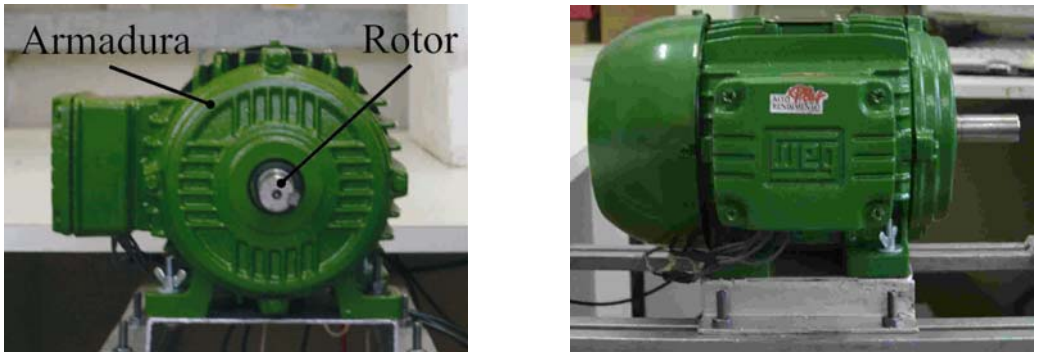
5.2 Especificações do sistema

Tabela 5. 1 - Especificações do sistema.

Motor de indução trifásico:	**Conversor:**
Fabricante: Weg, IP55	Fabricante: Semikron.
Número de pólos: 4, 1730 rpm	Modelo: SKS 27F B6U+B6I+E1IF.
220 V (Δ), I_{nom} = 2,98A, 60Hz	IGBT: SK 45GB 063.
P_{nom} = 1 HP (0,75 kW)	*Drivers*: SKHI 20opA.
η = 82,6 %, cosφ = 0,80	Potência: 3,7KW.
Parâmetros de ensaios:	*Link* CC: 310V.
$R_S = 7{,}65\Omega$, $R_R' = 2{,}76\Omega$ $X_{mag} = 66{,}69\Omega$, $X_{eq} = Xl_S + Xl_R' = 5{,}45\Omega$	Frequência de chaveamento: 10KHz.

5.3 O protótipo

Vários são os elementos compondo o protótipo final do controlador do motor de indução trifásico. A figura 5.1 inicia apresentando uma visão frontal (a) e uma lateral (b) do motor de indução trifásico de alto rendimento controlado pelo protótipo. Especificações deste motor foram apresentadas na seção 5.2.



(a) (b)

Figura 5. 1 - Motor de indução trifásico (MIT) utilizado.

A seguir, na figura 5.2, é mostrado o protótipo propriamente dito do controlador de motor de indução construído. Na figura aparecem placas de circuito impresso de todos os subsistemas mencionados nos capítulos anteriores: "Interface com Usuário", "Subsistema Digital", "Subsistema Analógico" e "Inversor Baseado em IGBT". O protótipo foi montado como uma estrutura em níveis, estando mais acima os circuitos digitais e de interfaceamento com o usuário e, mais abaixo, os circuitos analógicos e de potência.

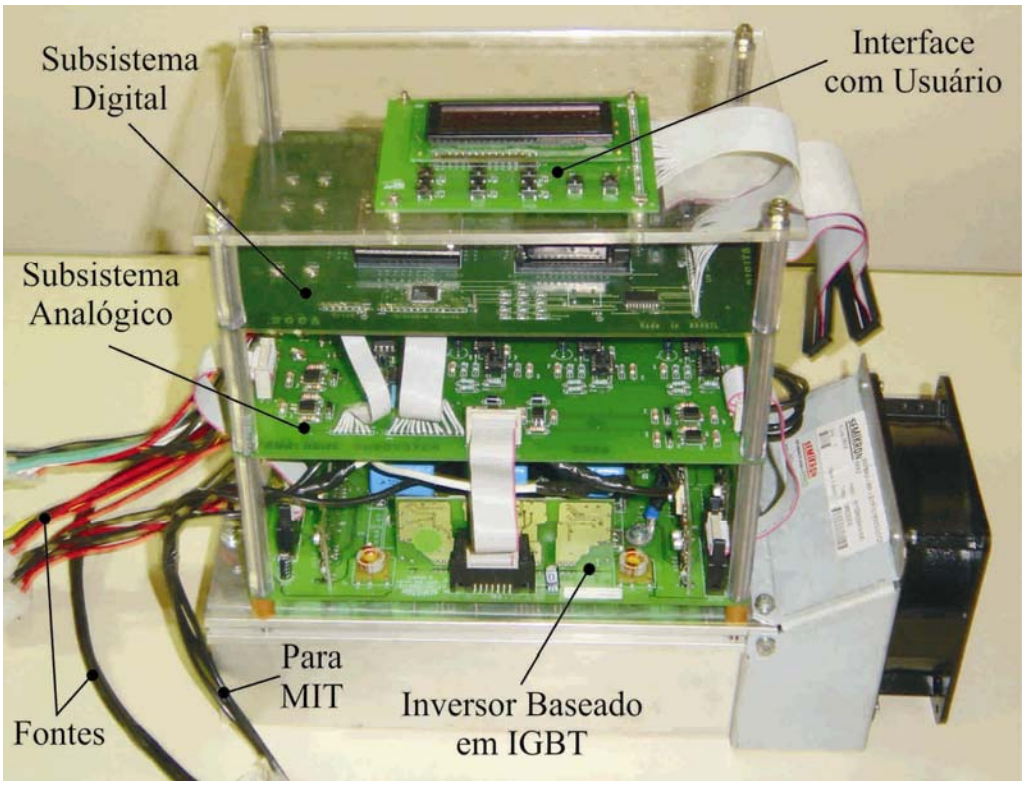


Figura 5. 2 - Visão do protótipo do controlador de MIT.

A figura 5.3 apresenta uma visão ampliada da placa de circuito impresso da "Interface com Usuário", composta por um módulo LCD (de 2 linhas e 16 colunas) e por um teclado com 8 botões.

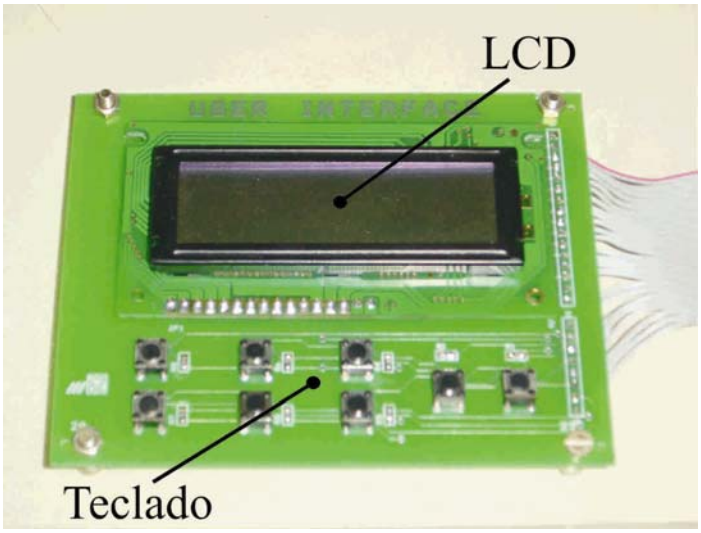


Figura 5. 3 - Placa de circuito da "Interface com Usuário".

A figura 5.4 mostra uma visão ampliada do circuito impresso do "Subsistema Digital". Este circuito foi desenvolvido utilizando o programa Orcad 9.2 e representa

uma das contribuições deste trabalho. Nesta figura, estão destacados elementos importantes como: o FPGA, o DSP, o conversor ADS8364 e os terminais ligados às fontes de alimentação. Esta placa de circuito impresso é composta por 6 camadas: "*Top*", "*Bottom*", "Roteamento interno", "GND", "VCC 5V e 3,3V", "VCC 1,2V e 1,2VPLL". Conectores laterais na placa permitem gravação da memória de dados do FPGA (dispositivo "EPCS64"), além de interligação com a placa de demonstração para DSPs "TMS320C6711" (ou seja, placa "TMS320C6711-DSK").

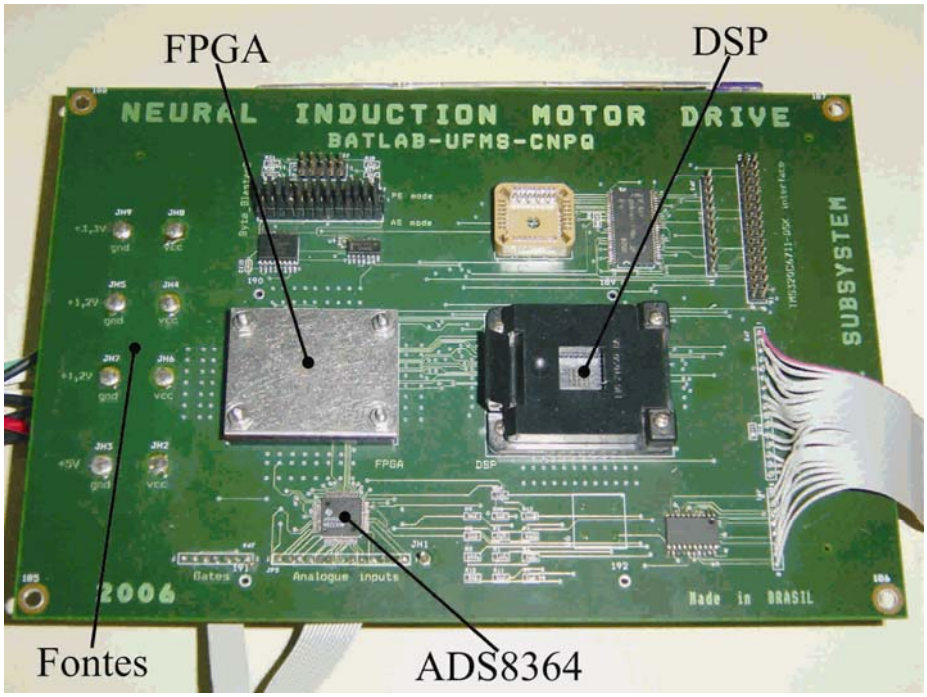


Figura 5. 4 - Placa do "Subsistema Digital".

Representando os circuitos de condicionamento de sinais do protótipo, na figura 5.5 está uma visão ampliada da placa de circuito impresso do "Subsistem Analógico". Nesta figura estão os três sensores de tensão (transformadores), e os locais contendo os dois sensores de corrente (os sensores estão na parte inferior da placa). Também são mostrados três pares de circuitos opto-acopladores, para comando das chaves de potência do "Inversor Baseado em IGBT", a partir de sinais vindos do "Subsistema Digital". A placa de circuito do "Subsistema Analógico" possui duas camadas de roteamento e é alimentada através dos 10 terminais localizados à direita na figura 5.5.

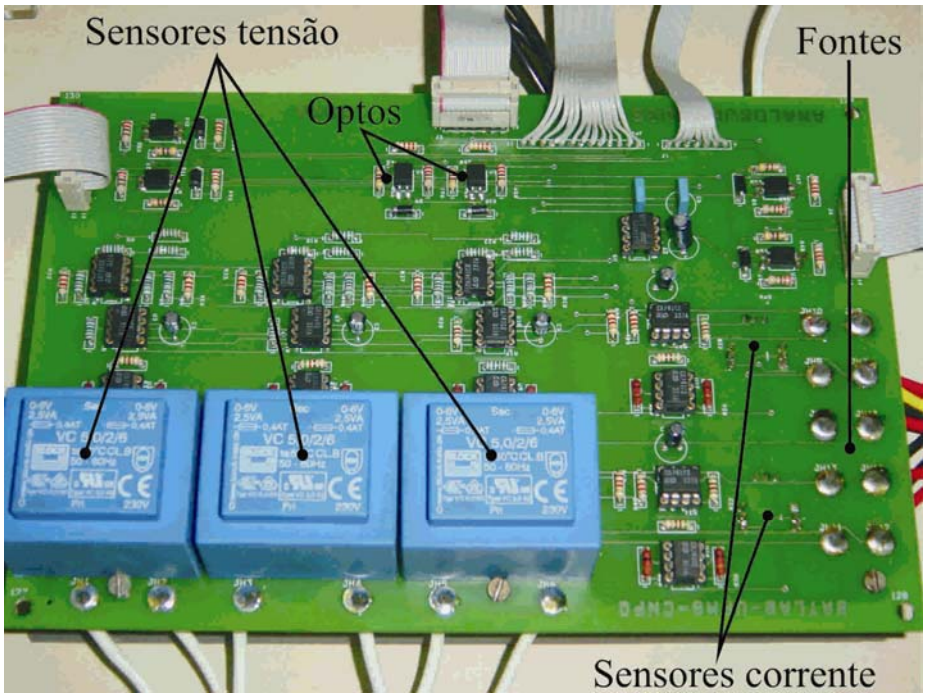


Figura 5. 5 - Placa do "Subsistema Analógico".

Para a tarefa de comutação de altas tensões e correntes, o protótipo final conta com o módulo "Inversor Baseado em IGBT", mostrado na figura 5.6. Este é um módulo de conversor trifásico de potência, adquirido da empresa "*Semikron*". Através de uma alimentação trifásica, um retificador não controlado gera a tensão de "*Link* CC" (terminais à direita na figura 5.6). Por meio de sinais de controle recebidos pelos "*Drivers(DRV)*", a ponte de "IGBTs" transforma a tensão de "*Link* CC" nas formas de ondas transmitidas para as saídas "OUT1", "OUT2" e "OUT3" do inversor. Estas saídas estão interligadas com os terminais de entrada do motor controlado e representam, na figura 5.6, os terminais indicados como "Saídas para MIT". Este é um inversor com potência máxima de 3,7 KW e que alcança a freqüência máxima de chaveamento de 20 KHz.

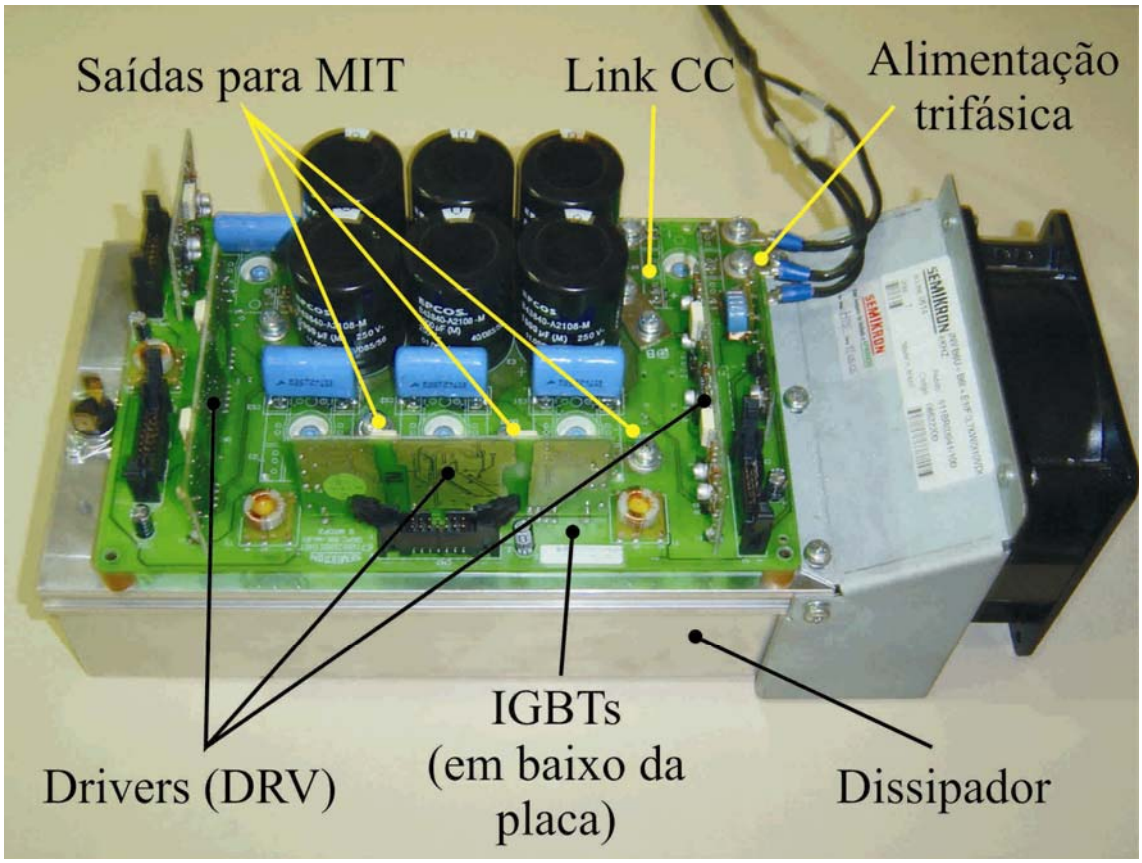


Figura 5. 6 - Módulo "Inversor Baseado em IGBT".

Alimentando todos os circuitos do protótipo, foi utilizada uma combinação de cinco fontes lineares, conforme mostrado na figura 5.7. A “Fonte 5” alimenta com +15V as entradas de cinco reguladores de tensão “LM338”. Estes reguladores geram as tensões: 1,2V (para os núcleos do FPGA e do DSP), 1,2V (alimentando circuitos “PLL” no FPGA), 3,3V (para portas de E/S digitais do sistema) e 5V (para os sensores de corrente e para o conversor ADS8364). Já as fontes de alimentação "Fonte1", "Fonte2", "Fonte3" são isoladas entre si. Cada uma alimenta com +15V uma das três "pernas" do módulo "Inversor Baseado em IGBT" (figura 5.6). Para suprir os circuitos de filtragem baseados em amplificadores operacionais na placa de circuito do “Subsistema Analógico”, a “Fonte 4” disponibiliza a tensão de ±15V. No lado esquerdo da figura 5.7, todas as tensões mencionadas estão acessíveis através dos conectores indicados na figura como “Saídas”.

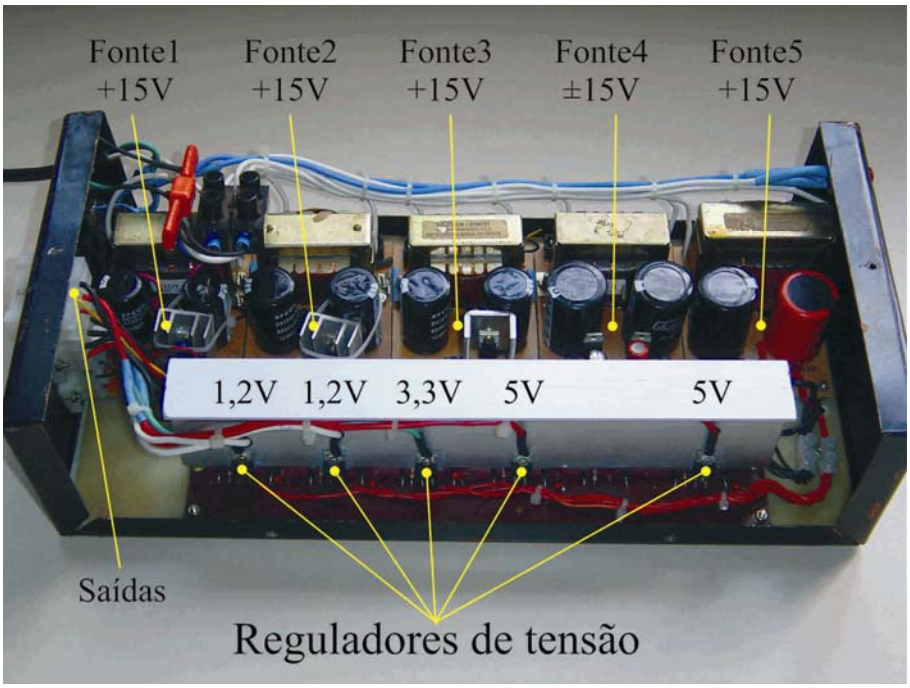


Figura 5. 7 - Fonte de alimentação do protótipo.

Finalizada a exposição do protótipo desenvolvido do controlador do motor de indução, chega o momento de se apresentar resultados obtidos. A seção a seguir apresenta resultados e discussões de testes e simulações do controlador do motor trifásico de corrente alternada proposto neste trabalho.

5.4 Resultados experimentais e de simulação

Esta seção de apresentação de resultados inicia com simulações em VHDL da execução de um neurônio com função de ativação sigmoidal e uma entrada. Inicialmente, a figura 5.8 trata de um exemplo apresentando como são codificados os valores recebidos pela entrada do neurônio. Todo valor está codificado em ponto flutuante de 32 bits (padrão IEEE 754). A figura 5.8 realça os campos de sinal, expoente e mantissa desta representação, além de suas respectivas conversões para decimal.

Bit 31	Bits 30 - 23	Bits 22 - 0
Bit de sinal: 0	Campo de Expoente: 01111110	Magnitude: 1.10011101101000101001010
Representações para sinal: 0 = +, 1 = –	Valor decimal do campo do expoente e expoente: 126 – 127 = -1	Valor decimal da mantissa: 1.6157620

Figura 5. 8 - Valor 0.8078 aproximado em formato IEEE 754.

As figuras 5.9 à 5.11 apresentam formas de onda representando a execução deste neurônio em três situações distintas. Matematicamente os cálculos realizados pelos circuitos do neurônio podem ser representados pelas equações 5.1 e 5.2.

$$f_{sig}(x) = sigmoid\left(\sum\left(entrada_{normalizada} \cdot peso_{entrada}\right) + bias\right) = saida\_neuronio \qquad (5.1)$$

$$f_{sig}(x) = sigmoid(0.00418848 \cdot x + 0.1) \cdot 1.4276 + 0.1657) \qquad (5.2)$$

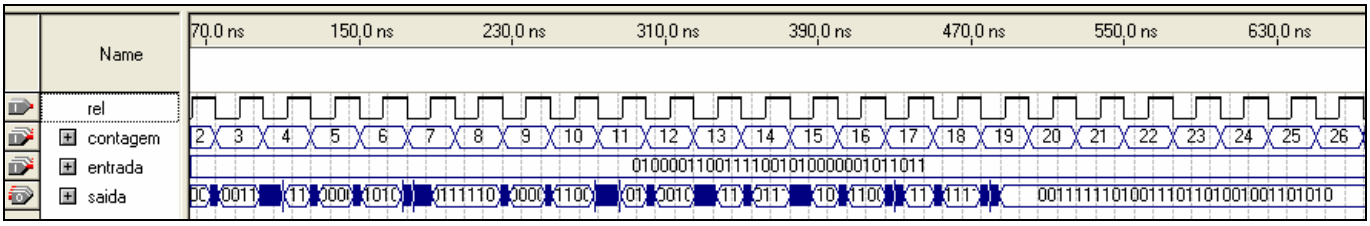


Figura 5. 9 - Saída = 0.8078, para uma entrada sem normalização x =188.62638. Latência do neurônio próxima de 500ns.

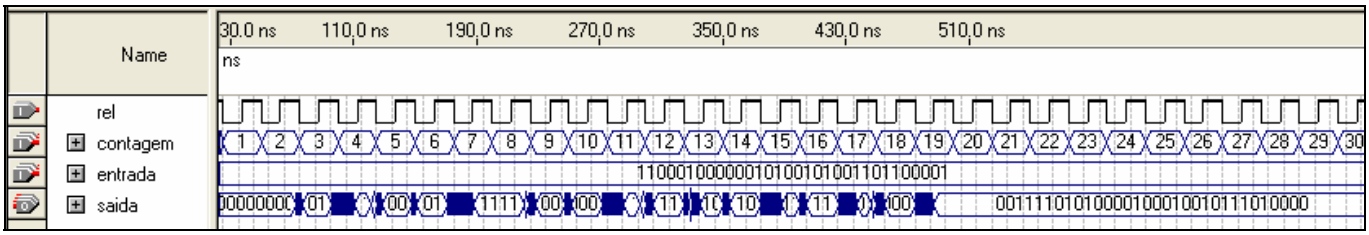


Figura 5. 10 - Saída = 0.0474, para uma entrada sem normalização x = −553.30284. Latência do neurônio próxima de 500ns.

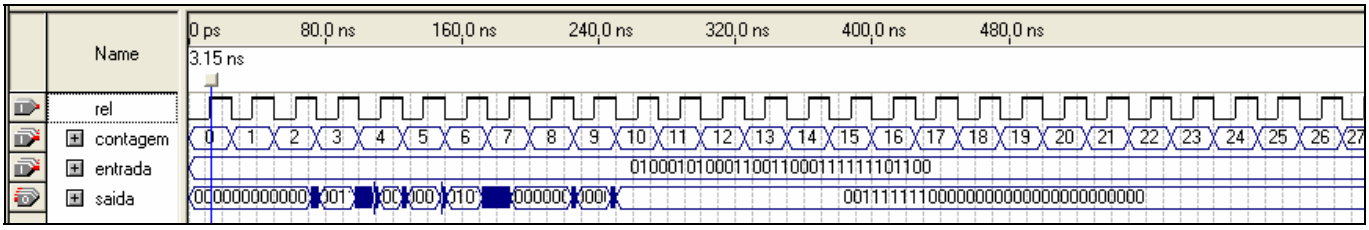


Figura 5. 11 - Saída = 1.0 (comportamento no mais infinito), resultado de uma entrada não normalizada x =2456.99536. Latência do neurônio próxima de 250ns.

Os resultados de simulação apresentados nas figuras 5.9 à 5.11 utilizam um sinal ideal "rel" (*clock*) com período igual a 0.03µs (33.333 MHz). No protótipo final, a freqüência deste sinal de *clock* teve de ser reduzida para 20 MHz. Esta redução foi necessária porque os circuitos reais das redes neurais apresentam atrasos adicionais de cálculo não presentes em formas de onda de simulação. Um maior período de *clock* permite que os circuitos dos neurônios tenham tempo suficiente para realizar seus cálculos, mesmo na presença de variações momentâneas deste *clock*.

A entrada "contagem" (figuras 5.9 à 5.11) representa valores de estado para o controle de fluxo do microprograma do neurônio (conforme já mencionado no capítulo

4). Cada uma das figuras representa uma situação de operação em que o neurônio "escolhe" um diferente intervalo de "*spline*" para calcular o resultado da sua função de ativação sigmoidal. Esta "escolha" é feita pelo neurônio, ao ser definido em qual trecho de "*spline*" está contida a entrada "x" após ter sofrido uma pré-normalização.

Nas figuras 5.9 e 5.10 obteve-se um atraso de resposta do neurônio próximo a 500ns. Este atraso é devido principalmente aos cálculos polinomiais necessários para representar a função sigmóide nos trechos de "*spline*" escolhidos. Já na figura 5.11, forçosamente, foi escolhido um valor de entrada que, após ser pré-normalizado, pertence a um trecho de saturação da função sigmóide (comportamento no $+\infty$), resultando em uma resposta do neurônio igual a 1.0. Neste caso, por não serem realizados todos os cálculos polinomiais, como ocorrido nas situações das figuras 5.9 e 5.10, o atraso de resposta do neurônio foi de apenas 250 ns.

Prosseguindo com a apresentação dos resultados obtidos, nas figuras 5.12 à 5.14 são apresentadas curvas medidas em três das saídas PWM do protótipo. Conforme já mencionado nos capítulos 2 e 3, as saídas PWM são produzidas pelo bloco "Controle PWM Space-Vector Baseado em Redes Neurais", dentro do FPGA (Subsistema Digital). Como pode ser notado pelas figuras, as três ondas PWM apresentam período igual a 100µs, ou seja, o próprio valor do período de amostragem utilizado pelo sistema para geração das tensões e correntes aplicadas ao motor de indução.

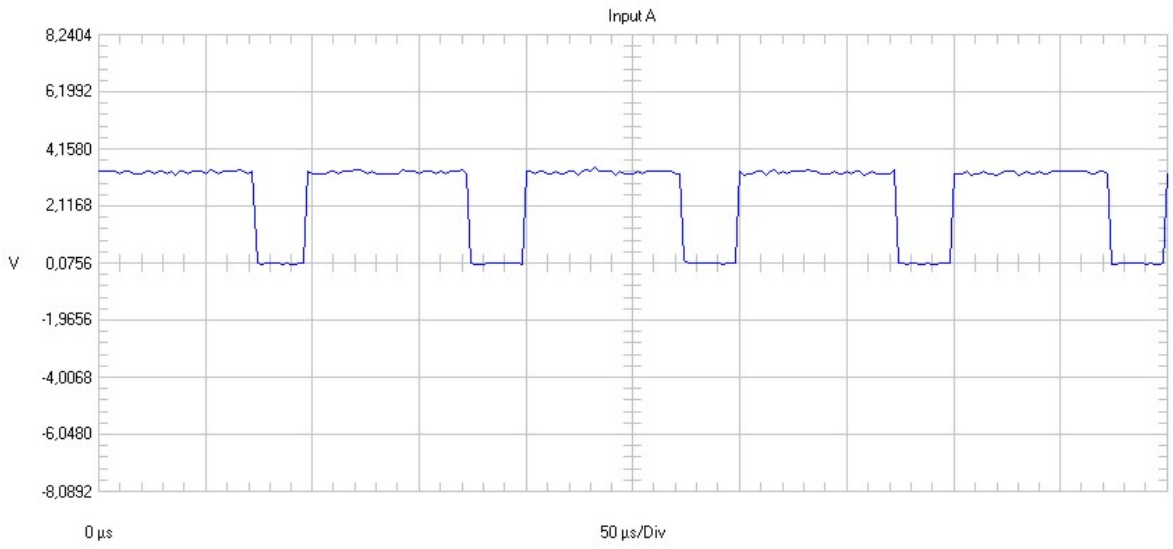


Figura 5. 12 - Sinal PWM na fase A do inversor.

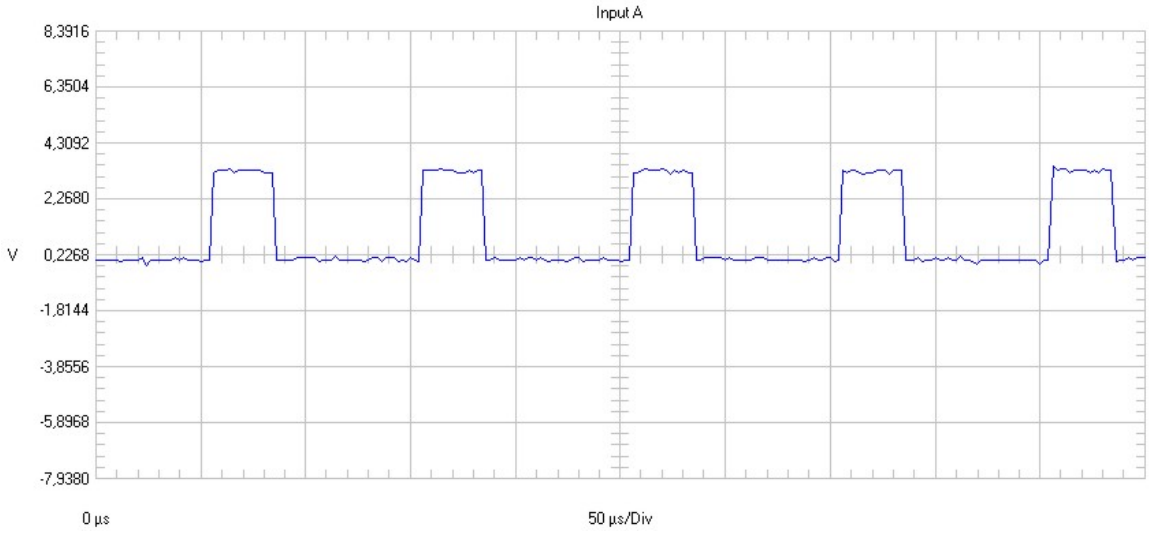


Figura 5. 13 - Sinal PWM na fase B do inversor.

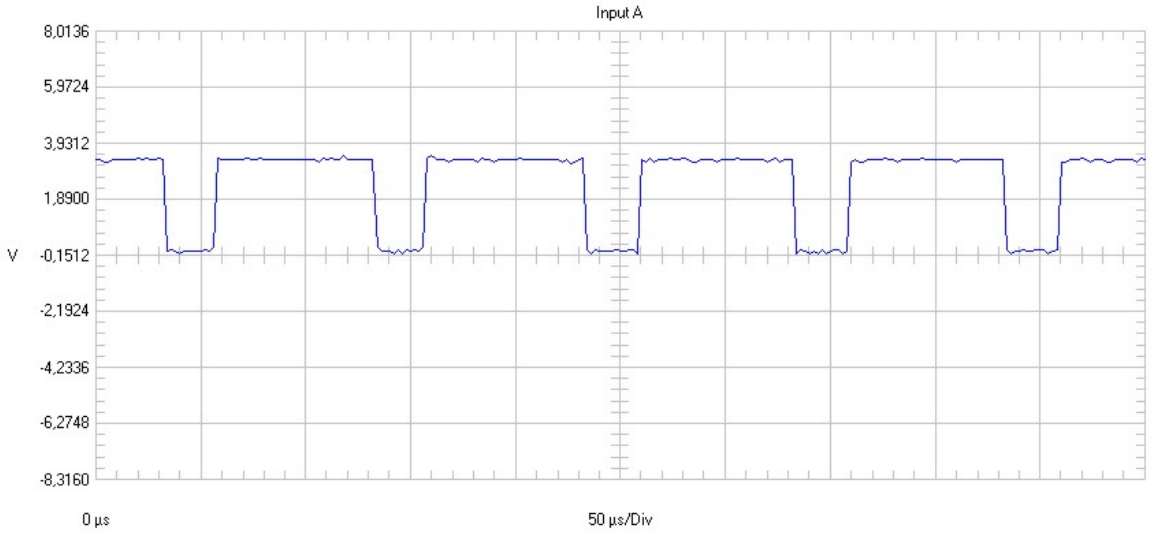


Figura 5. 14 - Sinal PWM na fase C do inversor.

Nas figuras 5.12 à 5.14 os sinais PWM apresentados estão com largura de pulso fixa. Porém para girar o motor, a cada 100µs (período de amostragem) a largura dos pulsos nas três fases deve ser modificada. Quando estes pulsos PWM de largura variável são filtrados, os resultados são curvas representando tempos de T_{on}, já discutidos no capítulo 3 deste trabalho. A figura 5.15 representa uma curva de T_{on} com 30 Hz, sendo suas características típicas de curvas pertencentes à região de submodulação. Já a figura 5.16 (com 57,2 Hz) representa o comportamento do sistema na região de sobremodulação-modo1, enquanto a curva de T_{on} com 60 Hz (na figura 5.17) representa operação na região de sobremodulação-modo2.

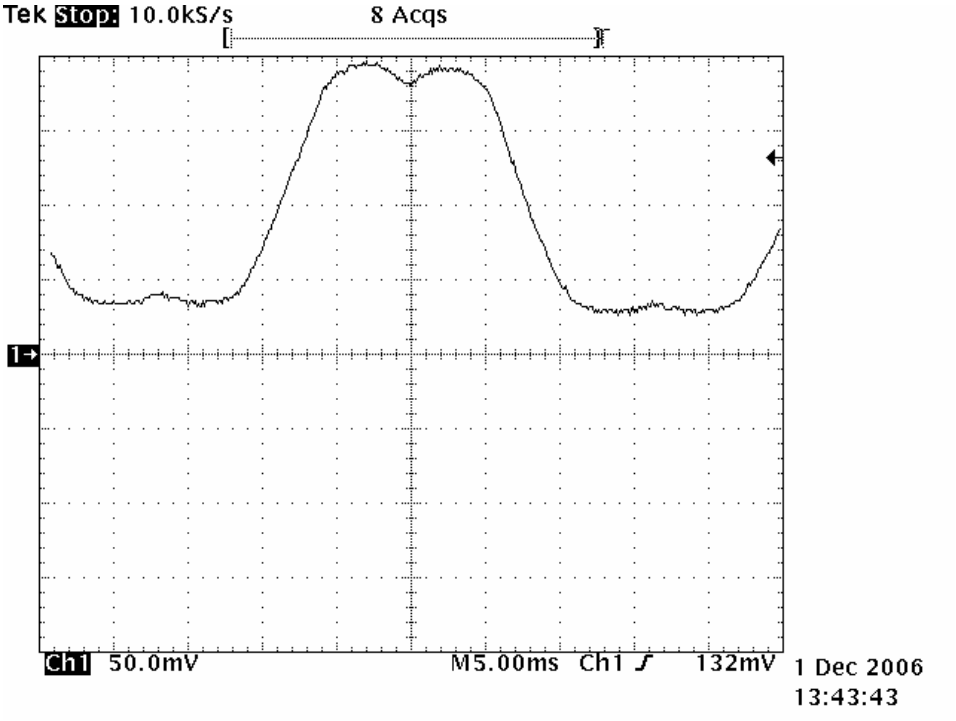


Figura 5. 15 - Exemplo de formas de onda de tempo de T_{on} para região de submodulação.

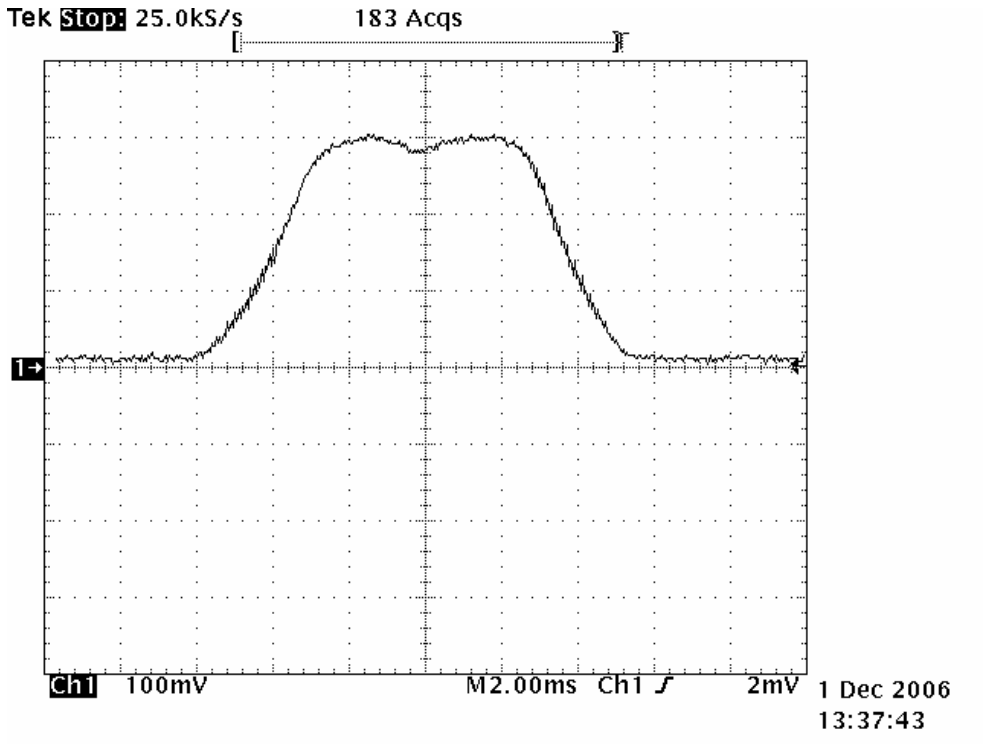


Figura 5. 16 - Exemplo de formas de onda de tempo de T_{on} para região de sobremodulação-modo 1.

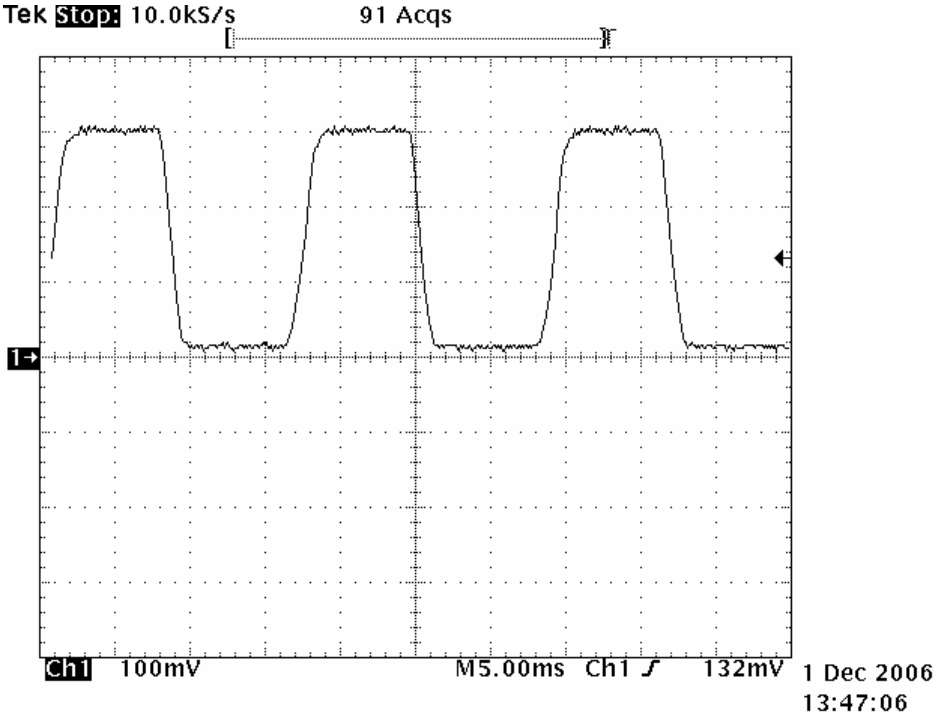


Figura 5. 17 - Exemplo de formas de onda de tempo de T_{on} para região de sobremodulação-modo 2.

O funcionamento do motor de indução a ser controlado permite a medição das formas de onda de tensão e corrente trifásicas aplicadas em seus terminais de entrada. A figura 5.18 apresenta formas de onda de tensão de duas fases do motor. Estas medições foram feitas após os filtros passa-baixa de 300 Hz no "Subsistema Analógico". Neste caso, o motor de indução foi operado com uma freqüência de tensão no estator igual a 30 Hz.

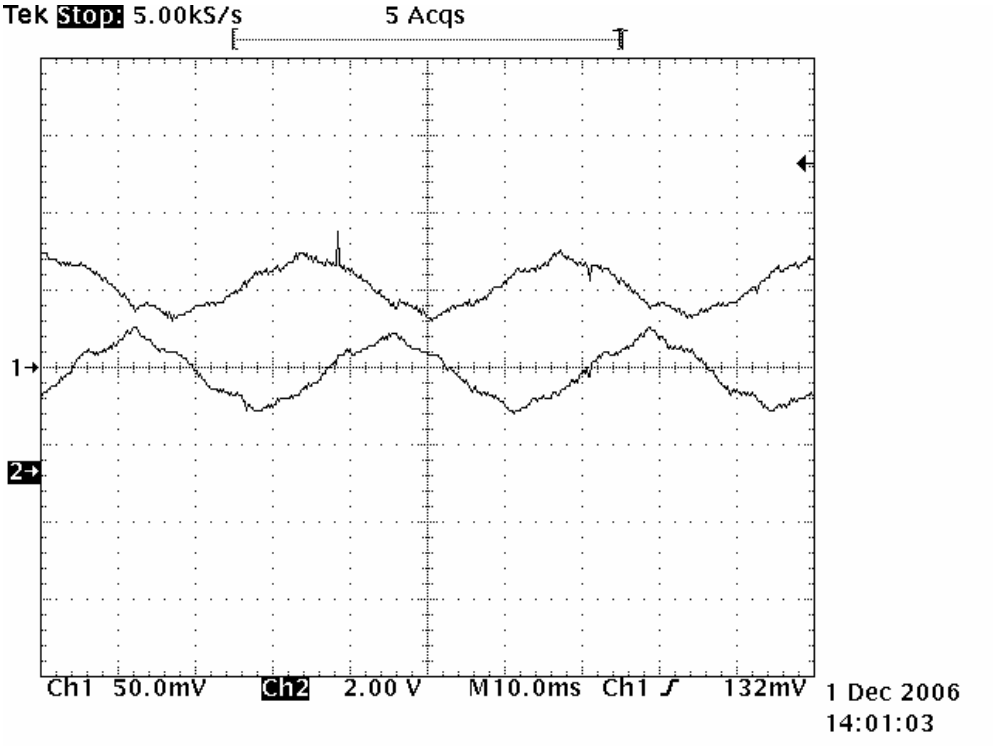


Figura 5. 18 - Tensões em duas fases do motor de indução.

As figuras 5.19 e 5.20 apresentam medições de corrente feitas diretamente em dois terminais de entrada do motor de indução. No momento da medição, o motor funcionava com uma freqüência de tensão no estator igual a 30 Hz. O osciloscópio utilizado realiza medições de corrente por meio de ponteiras por efeito Hall. Desta maneira, as formas de onda nas figuras apresentam unidades em volts (V) no eixo vertical, podendo, porém, estes valores ser interpretados diretamente em ampéres.

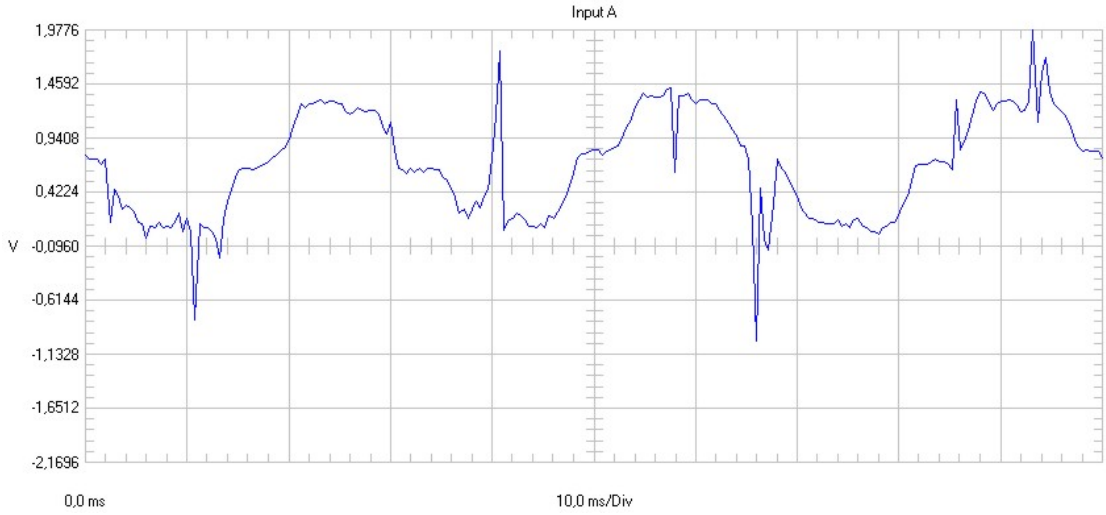


Figura 5. 19 - Corrente da fase B do motor de indução.

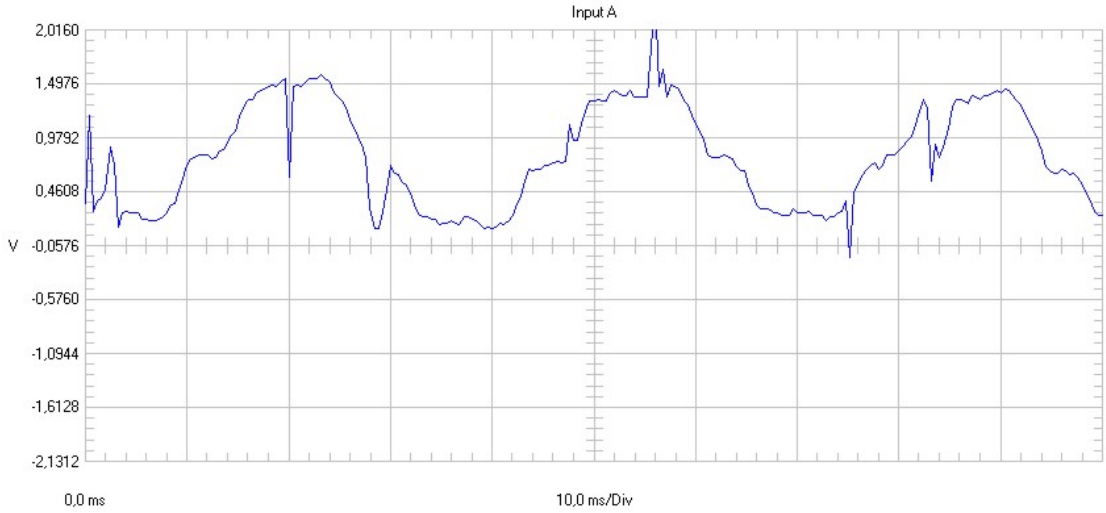


Figura 5. 20 - Corrente da fase C do motor de indução.

Através de medições filtradas de tensões e correntes do motor de indução controlado, os sistemas do caminho de retroação do controlador do motor podem iniciar seu trabalho. A figura 5.21 apresenta um exemplo de simulação (realizada no programa "Quartus 2") do funcionamento do módulo "Estimador de Fluxo" no FPGA (pontos em vermelho). O bloco recebeu em suas entradas formas de onda de tensão, cujas freqüências destas ondas foram variadas uma vez no tempo. O resultado é uma forma de onda de fluxo de estator que acompanha a freqüência das tensões de entrada. Na mesma figura há uma simulação no programa "Simulink" (em azul), realizada também nestas condições de entradas.

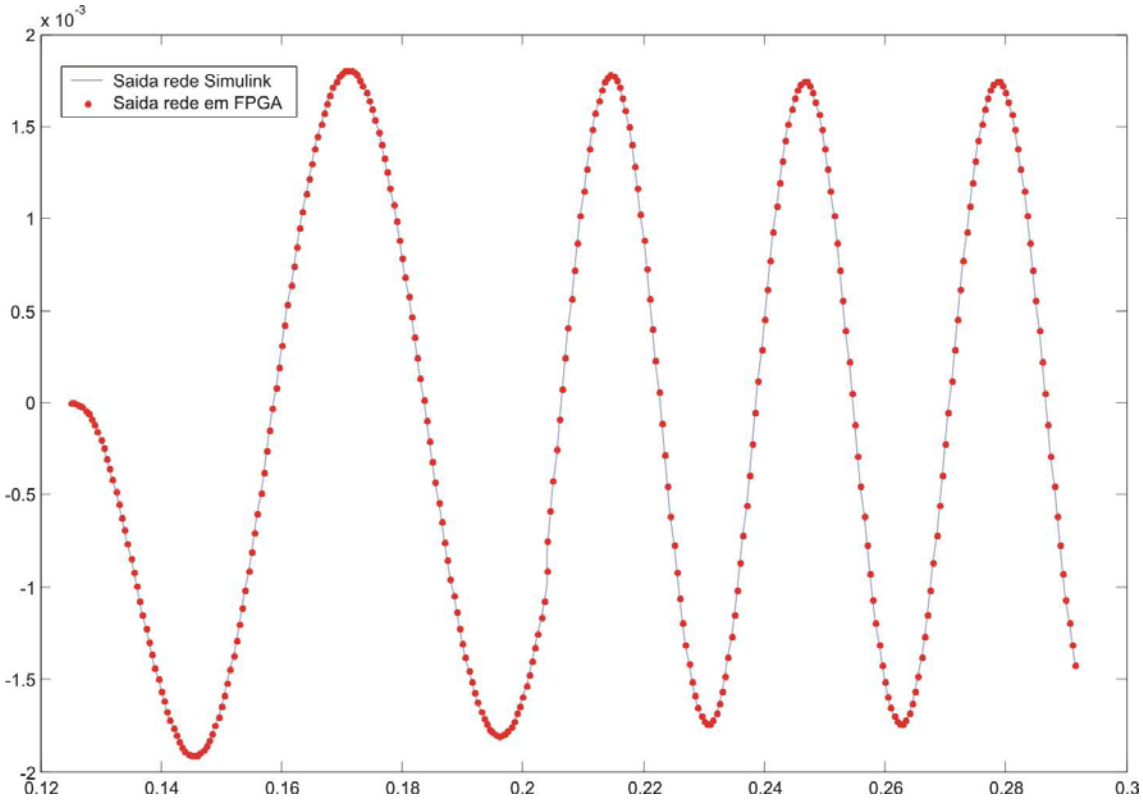


Figura 5. 21 - Simulação de estimação de fluxo do MIT.

5.5 Considerações finais

O capítulo buscou apresentar resultados experimentais e de simulação que comprovem o funcionamento dos módulos mais importantes do controlador do motor de indução construído. O protótipo final foi capaz de controlar a velocidade de um motor de indução trifásico convencional, porém, somente a partir de um controle V/Hz (malha aberta), não havendo tempo hábil para ser alcançado um controle total em malha fechada. As redes neurais no FPGA realizaram cálculos, na prática, com latências aceitáveis para a aplicação, de forma que o sistema possui um tempo de resposta total próximo a 20µs (menor que o período de amostragem de 100µs utilizado).

Assim, finaliza-se a apresentação e discussão de vários resultados experimentais e de simulação, realizados com os sistemas implementados. O capítulo 6 apresentará conclusões finais a respeito do trabalho como um todo.

CAPÍTULO 6 - CONCLUSÕES

6.1 Conclusões

O presente trabalho mostrou a possibilidade da construção em hardware de arquiteturas de redes neurais capazes de grande precisão numérica e alta capacidade de processamento paralelo. Para isto, foi seguida uma estratégia de construção de neurônios como processadores independentes reunidos, formando redes neurais completas inseridas dentro de um FPGA. O problema da representação de funções de ativação sigmoidais pôde ser resolvido através da representação desta função no hardware dos neurônios utilizando interpolação polinomial. O trabalho utilizou como base modelagens de redes definidas em trabalhos anteriores. As versões em hardware destas redes neurais mostraram-se capazes de realizar, satisfatoriamente, tanto a função de estimação de fluxo estatórico, quanto à função de modulação por vetores espaciais "SVM".

Em contrapartida aos sucessos alcançados, o protótipo mostrou-se excessivamente sensível aos ruídos de chaveamento gerados pelo funcionamento dos IGBTs do inversor de potência. O fato dos sistemas digitais do protótipo utilizar baixas tensões de alimentação (3,3V e 1,2V) cria uma exigência de alta relação sinal ruído, somente alcançada com uso de blindagens adequadas e filtragens em pontos de alimentação.

Os resultados finais deste trabalho representam uma porta de entrada para muitos desenvolvimentos futuros na área de implementação de redes neurais em hardware. A seção a seguir apresenta algumas propostas para trabalhos futuros.

6.2 Propostas para trabalhos futuros

Várias são as melhorias possíveis para trabalhos futuros, tais como utilização de precisões com menor número de bits de representação nos neurônios implementados dentro do FPGA. Esta melhoria poderia permitir a construção de redes com mais neurônios dentro da mesma área de silício do FPGA.

Outra possibilidade de alteração seria a implementação em dispositivo único de todo o sistema, ou seja, a implementação de todos os blocos do sistema em VHDL e

dentro de um único FPGA de maior capacidade. Isto permitiria maior imunidade a ruídos externos devido a maior proximidade dos componentes do sistema.

Uma terceira proposta de trabalho futuro seria a construção de arquiteturas de redes neurais com pesos não fixos e interligação entre neurônios reconfigurável. Esta abordagem representaria uma generalização do trabalho atual, permitindo a implementação em hardware de diferentes tipos existentes de redes neurais artificiais.

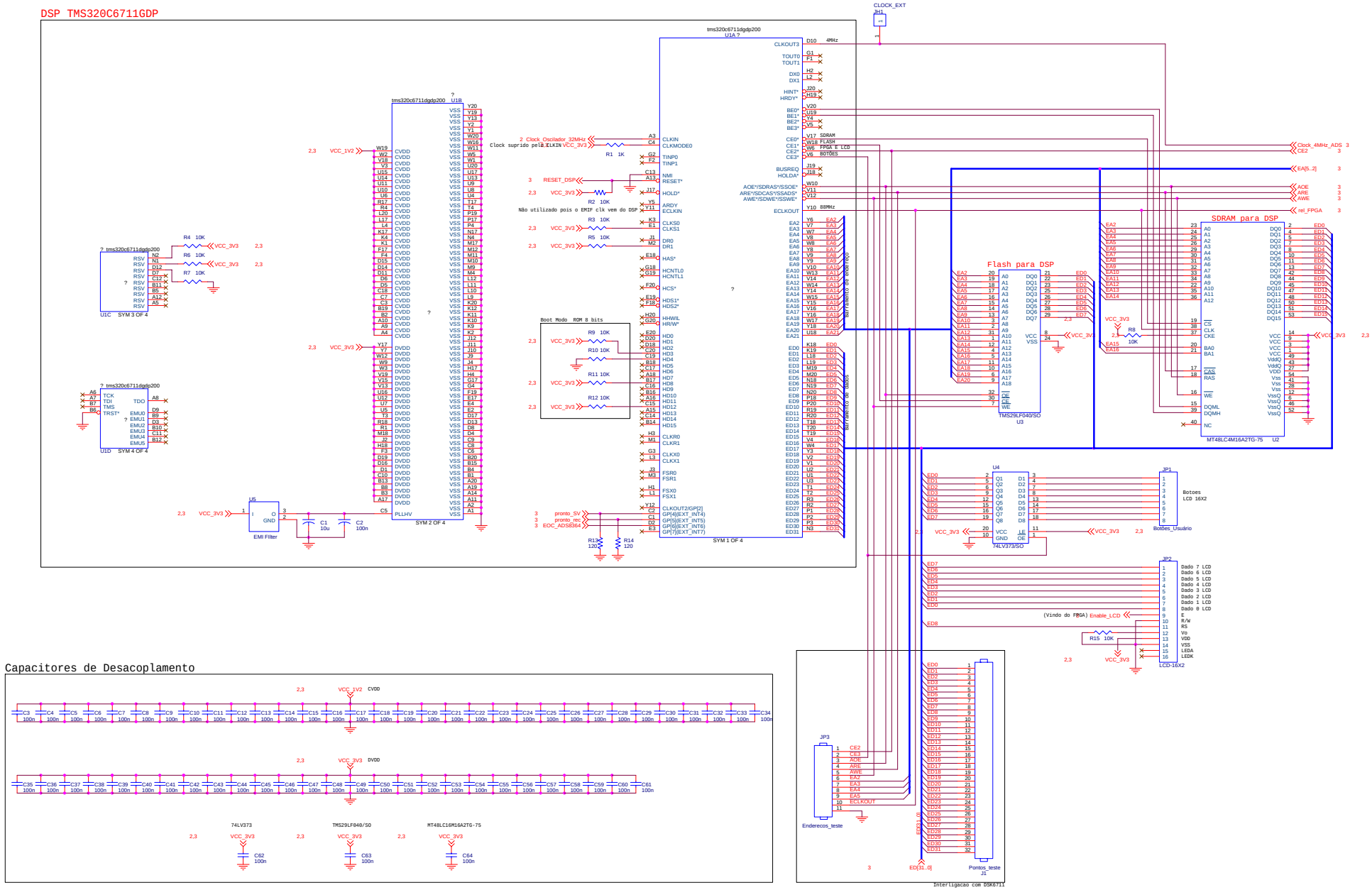
Novos trabalhos na área devem procurar sempre alcançar melhores compromissos entre área de silício ocupada, usar precisão numérica adequada para aplicação e alcançar maior velocidade de processamento na arquitetura final proposta. A figura 6.1 apresenta um trecho do arquivo de sumário de análise e síntese gerado ao ser compilado, no software Quartus 2, os projetos em VHDL das redes neurais deste trabalho. Nesta figura, estão dados de ocupação de recursos de hardware do FPGA Stratix 2, tais como: número de ALUTs (tabelas de *look-up* adaptativas), ALMs (módulos lógicos adaptativos, semelhantes aos blocos lógicos CLBs), número de blocos de DSP, etc..

Analysis & Synthesis Resource Usage Summary	
Resource	Usage
Estimated Total ALUTs	49562
Total combinational functions	38672
ALUT usage by number of inputs	
-- 7 input functions	165
-- 6 input functions	5132
-- 5 input functions	13768
-- 4 input functions	5114
-- <=3 input functions	14493
-- Combinational cells for routing	0
ALUTs by mode	
-- normal mode	35162
-- extended LUT mode	165
-- arithmetic mode	3345
-- shared arithmetic mode	0
Total registers	7309
Estimated ALMs: partially or completely used	24,781
I/O pins	72
DSP block 9-bit elements	256
Maximum fan-out node	gerador_clock_redes:inst|inst3
Maximum fan-out	6992
Total fan-out	187377
Average fan-out	4.05

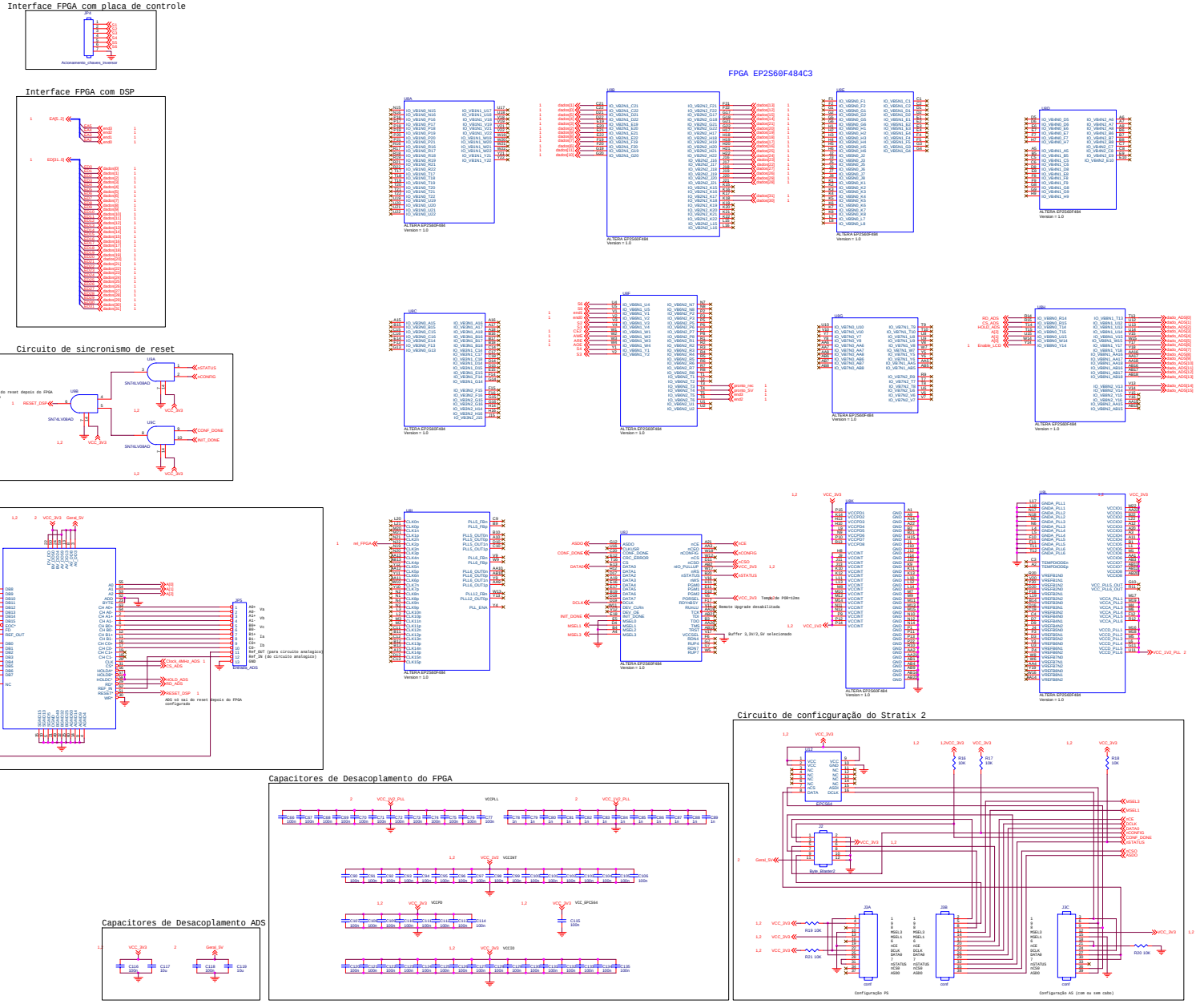
Figura 6. 1. Dados de utilização global do FPGA Stratix 2.

Anexo A:
Esquemáticos dos circuitos do protótipo

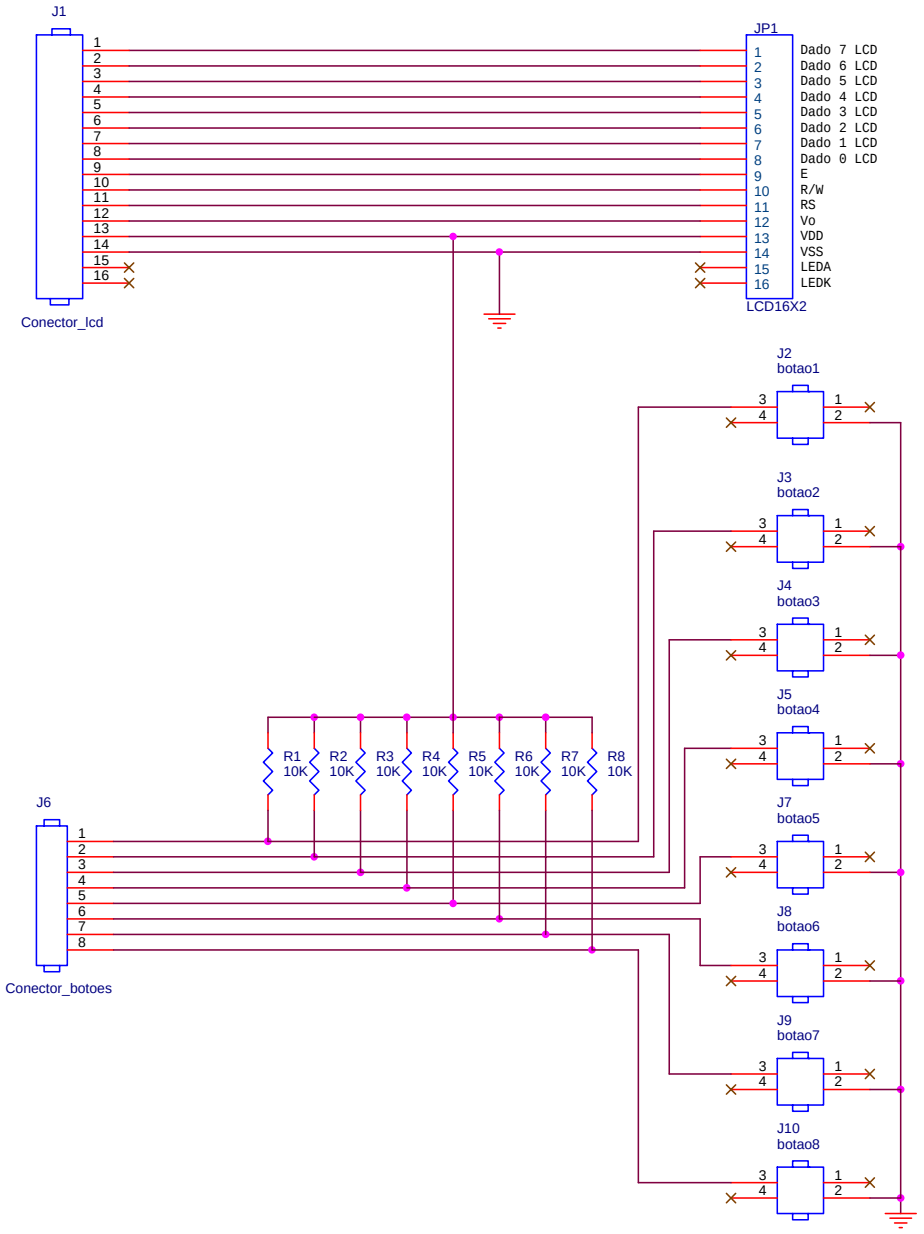
Esquemático do DSP TMS320C6711 e periféricos (Subsistema Digital)



Esquemático do FPGA Stratix 2 e periféricos (Subsistema Digital)

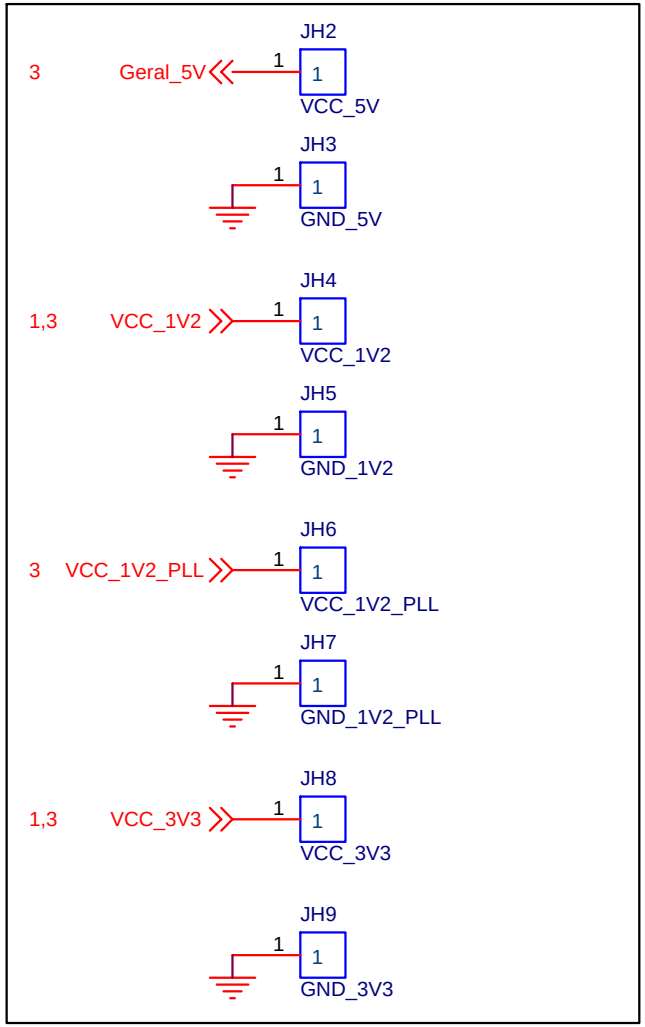


Esquemático do módulo "Interface com Usuário"

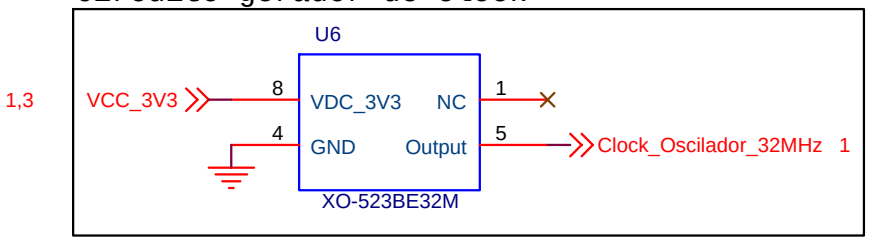


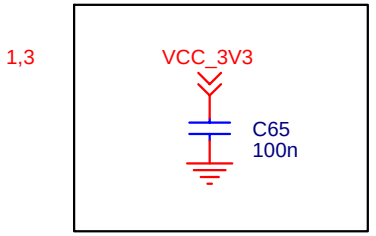
Esquemático dos circuitos de alimentação e geração de *clock* (Subsistema Digital)

Circuito gerador de alimentação



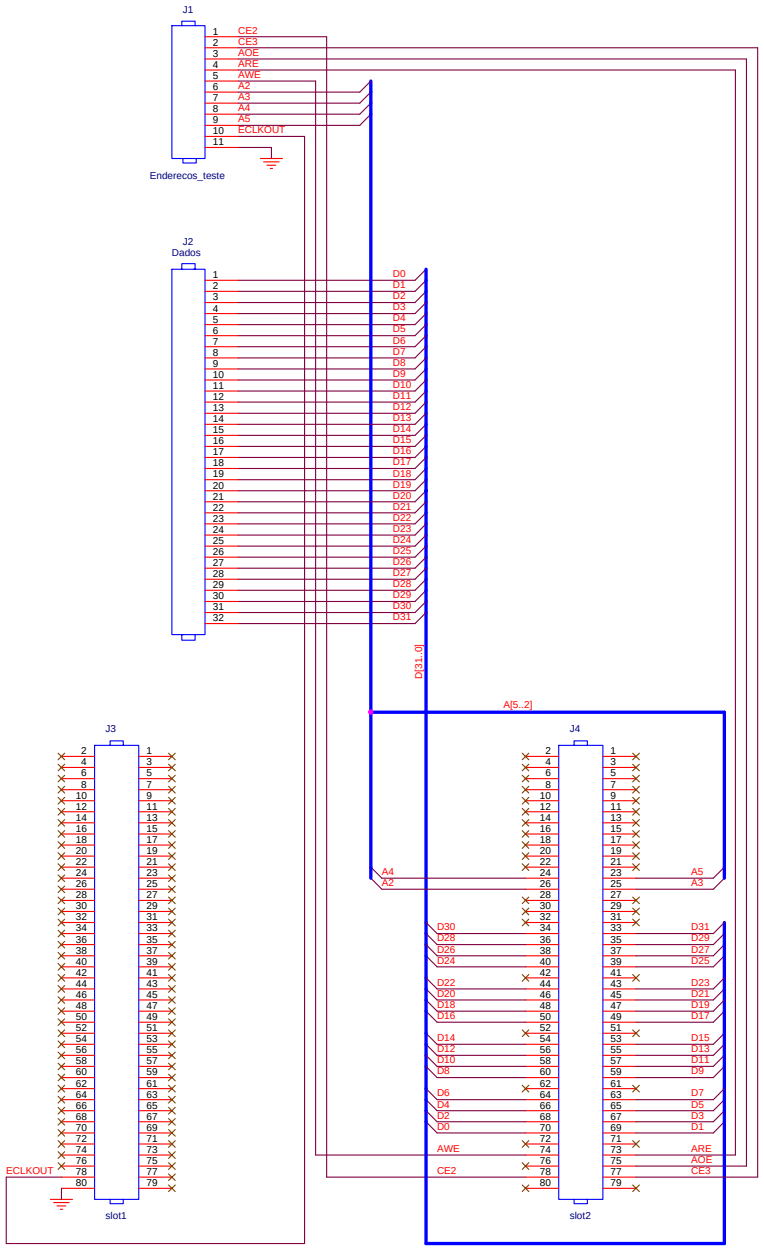
Circuito gerador de clock



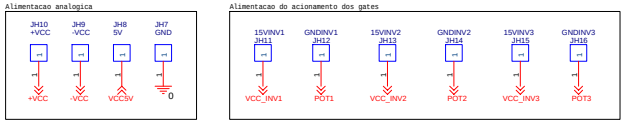


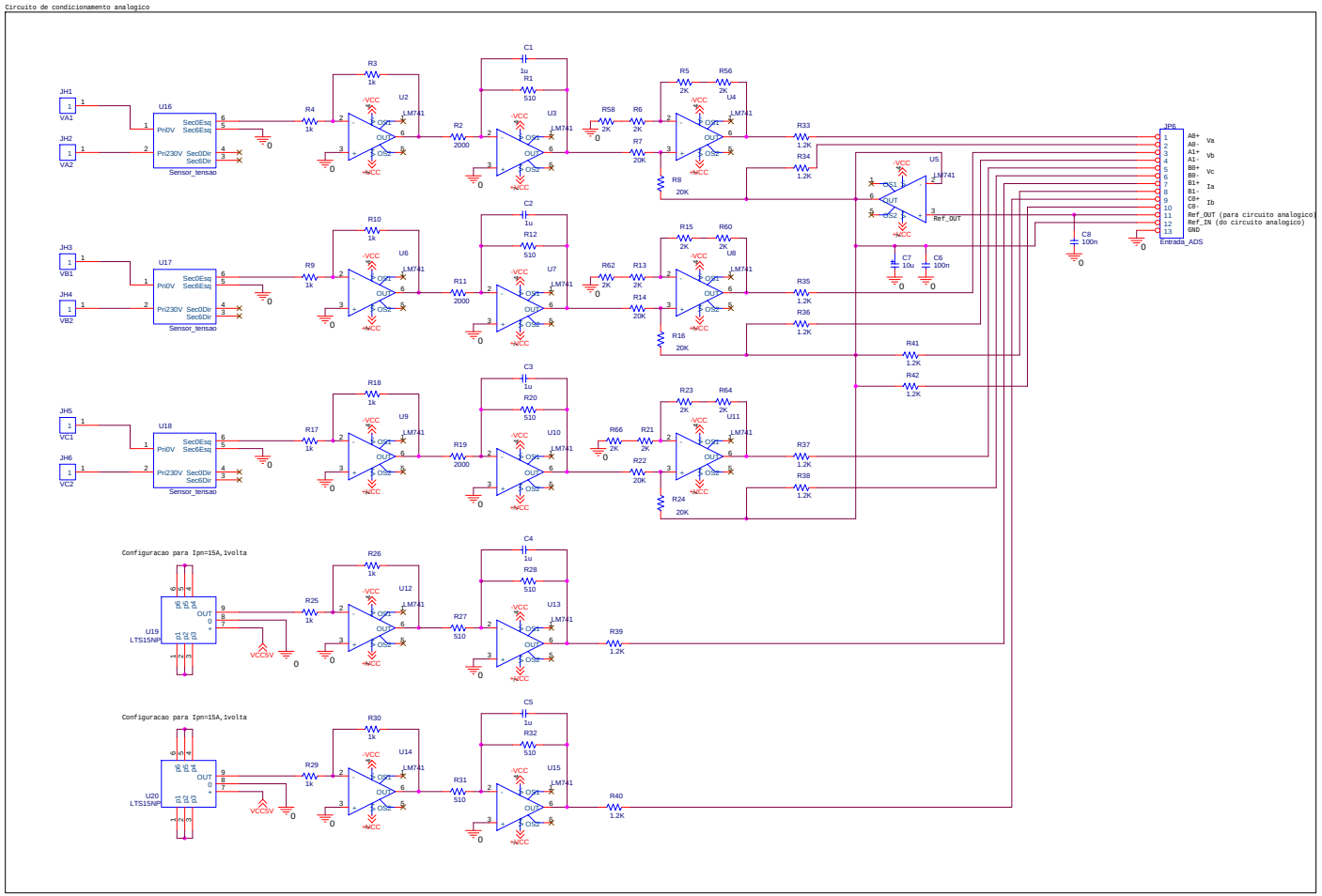
Capacitores de Desacoplamento

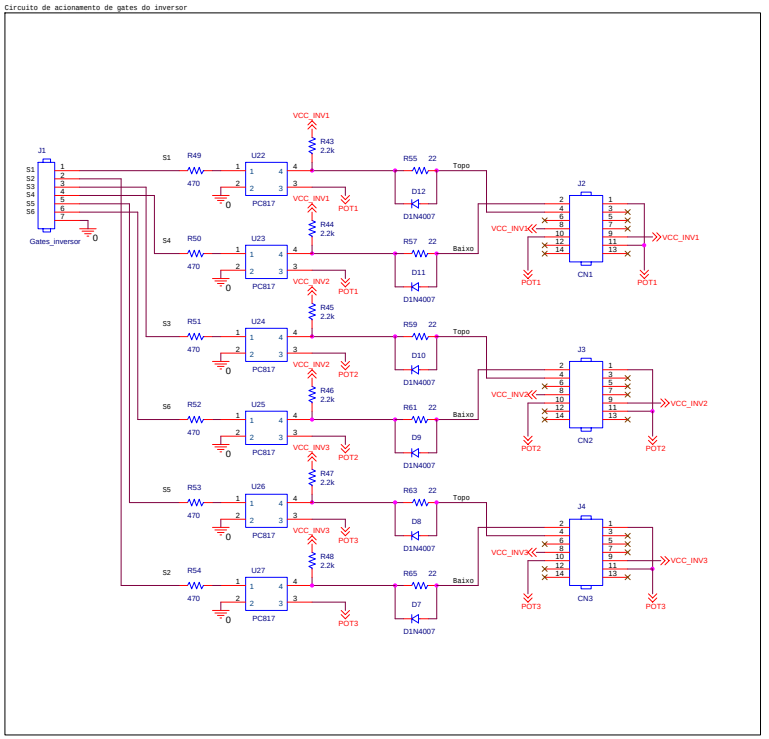
Esquemático da placa de interligação do "Subsistema Digital" com a placa de demonstração "TMS320C6711-DSK"



Esquemático dos circuitos de condicionamento de sinal e acionamento de "*drivers*" de potência (Subsistema Analógico)







Anexo B:
Códigos em VHDL e em C

Exemplo de código VHDL de um controlador de camada de rede neural:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;

--  Entity Declaration

ENTITY controlador_camada_sigmoid_18e IS

        -- {{ALTERA_IO_BEGIN}} DO NOT REMOVE THIS LINE!
        PORT
        (rel,anterior:in std_logic;
         entradas:out std_logic_vector(4 downto 0);
         estado:out std_logic_vector(5 downto 0);
         proximo:out std_logic
        );
        -- {{ALTERA_IO_END}} DO NOT REMOVE THIS LINE!

END controlador_camada_sigmoid_18e;

--  Architecture Body

ARCHITECTURE interior OF controlador_camada_sigmoid_18e IS
        signal fim,hab:std_logic;
        signal conta,ent:integer;
        constant limite:integer:=40;-- Igual ao numero de estados mais 1
        constant limite_entradas:integer:=18;

BEGIN

        captura:process(rel)
        begin
                if(rel'event and rel='1')then
                        if(anterior='1')then
                                hab<='1';
                        elsif(fim='1')then
                                hab<='0';
                        end if;
                end if;
        end process captura;
```

```
        contagem:process(rel)
        begin
                if(rel'event and rel='0')then
                        if(hab='1')then
                                if(conta<limite)then
                                --Causa atraso entre contagem de estado e de seleção de entradas
                                        if(conta>=1 and ent<limite_entradas)then
                                           ent<=ent+1;
                                        end if;
                                        conta<=conta+1;
                                        fim<='0';
                                else    conta<=0;
                                        ent<=0;
                                        fim<='1';
                                end if;
                         else   conta<=0;
                                 fim<='0';
                        end if;
                end if;
        end process contagem;

        entradas<=CONV_STD_LOGIC_VECTOR(ent,5);
        estado<=CONV_STD_LOGIC_VECTOR(conta,6);
        proximo<=fim;

END interior;
```

Exemplo de código VHDL de um neurônio digital de uma entrada e com função de ativação sigmoidal:

```
ENTITY microprograma_nao_linear_duplo_1e_n2_rec IS
        -- {{ALTERA_IO_BEGIN}} DO NOT REMOVE THIS LINE!
        PORT
        (rel:in std_logic;
         contagem:in UNSIGNED(4 downto 0);
         entrada,atual,Mult,Mem:in std_logic_vector(31 downto 0);
         hab,sel1,sel2,copia:out std_logic;
         peso1,peso2,dado_M:out std_logic_vector(31 downto 0)
        );
        -- {{ALTERA_IO_END}} DO NOT REMOVE THIS LINE!

END microprograma_nao_linear_duplo_1e_n2_rec;

ARCHITECTURE interior OF microprograma_nao_linear_duplo_1e_n2_rec IS
-- Declaração dos sinais omitida!
BEGIN
--------------------------------------------------------------------------------------------------------------------------
--Rotina de controle do neurônio sigmoid
palavra<=
```

-- Comando:	Dado_M:	Peso1:	Peso2:	Contagem:
--Fazendo somatório de entradas ponderadas				
\temp1=E*P;\	& sem	& P1	& entrada	when contagem=1 else
\temp2=temp1;temp1=BIAS;\	& sem	& BIAS	& um	when contagem=2 else
\X=temp1+temp2\	& sem	& sem	& sem	when contagem=3 else
--Esperando definição do intervalo de spline				
\espera;\	& sem	& sem	& sem	when contagem=4 else
--Comandos para geração de novas potências				
\Gera potencia\	& sem	& X	& Mem	when (rumo/="001" and rumo/="111") and (contagem=7 or contagem=9 or contagem=11 or contagem=13) else
--Avaliando sigmoid no intervalo menor que -7				
\7)Interpolando!;\	& sem	& I1Ca	& um	when contagem>=5 and contagem<=15 and rumo="001" else
--Avaliando sigmoid no intervalo entre -7 e -3				
\0)Interpolando!;\	& atual	& I2Ca	& um	when contagem=5 and rumo="010" else
\1)Interpolando!;\	& sem	& I2Pb	& Mem	when contagem=6 and rumo="010" else
\2)Interpolando!;\	& Mult	& I2Pc	& Mult	when contagem=8 and rumo="010" else

```
\3)Interpolando!;\      & Mult     & I2Pd     & Mult     when contagem=10 and rumo="010" else
\4)Interpolando!;\      & Mult     & I2Pe     & Mult     when contagem=12 and rumo="010" else
\5)Interpolando!;\      & Mult     & I2Pf     & Mult     when contagem=14 and rumo="010" else

--Avaliando sigmoid no intervalo entre -3 e -1
\0)Interpolando!;\      & atual    & I3Ca     & um       when contagem=5 and rumo="011" else
\1)Interpolando!;\      & sem      & I3Pb     & Mem      when contagem=6 and rumo="011" else
\2)Interpolando!;\      & Mult     & I3Pc     & Mult     when contagem=8 and rumo="011" else
\3)Interpolando!;\      & Mult     & I3Pd     & Mult     when contagem=10 and rumo="011" else
\4)Interpolando!;\      & Mult     & I3Pe     & Mult     when contagem=12 and rumo="011" else
\5)Interpolando!;\      & Mult     & I3Pf     & Mult     when contagem=14 and rumo="011" else

--Avaliando sigmoid no intervalo entre -1 e 1
\0)Interpolando!;\      & atual    & I4Ca     & um       when contagem=5 and rumo="100" else
\1)Interpolando!;\      & sem      & I4Pb     & Mem      when contagem=6 and rumo="100" else
\2)Interpolando!;\      & Mult     & I4Pc     & Mult     when contagem=8 and rumo="100" else
\3)Interpolando!;\      & Mult     & I4Pd     & Mult     when contagem=10 and rumo="100" else
\4)Interpolando!;\      & Mult     & I4Pe     & Mult     when contagem=12 and rumo="100" else
\5)Interpolando!;\      & Mult     & I4Pf     & Mult     when contagem=14 and rumo="100" else
```

```
--Avaliando sigmoid no intervalo entre 1 e 3
\0)Interpolando!;\          & atual     & I5Ca     & um      when contagem=5 and rumo="101" else
\1)Interpolando!;\          & sem       & I5Pb     & Mem     when contagem=6 and rumo="101" else
\2)Interpolando!;\          & Mult      & I5Pc     & Mult    when contagem=8 and rumo="101" else
\3)Interpolando!;\          & Mult      & I5Pd     & Mult    when contagem=10 and rumo="101" else
\4)Interpolando!;\          & Mult      & I5Pe     & Mult    when contagem=12 and rumo="101" else
\5)Interpolando!;\          & Mult      & I5Pf     & Mult    when contagem=14 and rumo="101" else

--Avaliando sigmoid no intervalo entre 3 e 7
\0)Interpolando!;\          & atual     & I6Ca     & um      when contagem=5 and rumo="110" else
\1)Interpolando!;\          & sem       & I6Pb     & Mem     when contagem=6 and rumo="110" else
\2)Interpolando!;\          & Mult      & I6Pc     & Mult    when contagem=8 and rumo="110" else
\3)Interpolando!;\          & Mult      & I6Pd     & Mult    when contagem=10 and rumo="110" else
\4)Interpolando!;\          & Mult      & I6Pe     & Mult    when contagem=12 and rumo="110" else
\5)Interpolando!;\          & Mult      & I6Pf     & Mult    when contagem=14 and rumo="110" else

--Avaliando sigmoid no intervalo maior que 7
\8)Interpolando!;\          & sem       & I7Ca     & um      when contagem>=5 and contagem<=15 and rumo="111" else

--Finalizando Interpolação
```

```
\6)Interpolando!;\          & sem      & sem      & sem      when contagem=15 else
--Esperando inicio
\espera;\                   & sem      & sem      & sem      ;

        --Avaliador de X para escolha do intervalo de spline
        define_rumo:process(rel)
        begin
                if(rel'event and rel='1')then
                        if(contagem=4)then
                          X<=atual;
                          if atual(31)>limite1(31) or (atual(31)=limite1(31) and atual(30 downto 0)>=limite1(30 downto 0))then
                                  rumo<="001";
                          elsif atual(31)>limite2(31) or (atual(31)=limite2(31) and atual(30 downto 0)>=limite2(30 downto 0))then
                              rumo<="010";
                        elsif atual(31)>limite3(31) or (atual(31)=limite3(31) and atual(30 downto 0)>=limite3(30 downto 0))then
                              rumo<="011";
                        elsif atual(31)>limite4(31) or (atual(31)=limite4(31) and atual(30 downto 0)<=limite4(30 downto 0))then
                              rumo<="100";
                        elsif atual(31)>limite5(31) or (atual(31)=limite5(31) and atual(30 downto 0)<=limite5(30 downto 0))then
                              rumo<="101";
```

```
                    elsif atual(31)>limite6(31) or (atual(31)=limite6(31) and atual(30 downto 0)<=limite6(30 downto 0))then
                        rumo<="110";
                    else rumo<="111";
                    end if;
                end if;
            end if;
        end process define_rumo;
-------------------------------------------------------------------------------------------------------------------------
        --Distribuindo bits da palavra para as saídas
        hab<=palavra(99);
        sel1<=palavra(98);
        sel2<=palavra(97);
        copia<=palavra(96);
        Dado_M<=palavra(95 downto 64);
        peso1<=palavra(63 downto 32);
        peso2<=palavra(31 downto 0);
-------------------------------------------------------------------------------------------------------------------------

END interior;
```

Declaração em C de variáveis importantes para o DSP TMS320C6711:

```
//---------------------------------------------------------------------------------------------
// Incluindo cabeçalhos necessários para a geração dos arquivos de baixo nível
//---------------------------------------------------------------------------------------------
#include <std.h>
#include <math.h>
#include <prd.h>
#include "configuracao_para_dskcfg.h"

//---------------------------------------------------------------------------------------------
// Denições e constantes para o controle opcional VHz, na função "Caminho_retroacao"
//---------------------------------------------------------------------------------------------
#define inic_cont 400//400*100us=0,04s. Período entre atualizações de freqüência do
motor.
#define Wstep 0.1//Valor de incremento (em Hz) da freqüência do motor.
#define Const2Pi 6.28318530718
#define cte 3.6//Para transformação de uma freqüência (Hz) em um valor de módulo
(volts)
#define per 100E-6;//Deve bater com a configuração do objeto "Iteracao_amostragem"
do DSP/BIOS

//---------------------------------------------------------------------------------------------
// Variáveis do controle opcional VHz, na função "Caminho_retroacao"
//---------------------------------------------------------------------------------------------
Float Watual=5;//Freqüência inicial desejada do motor
Float Wmeta=30;//Freqüência final desejada do motor
Float Theta=0,DeltaTheta,M,A;
int cont=inic_cont;//Contador para períodos de atualização de valores de freqüência
int inicio=0;//Inicia não permitindo o uso de algoritmo V/Hz

//---------------------------------------------------------------------------------------------
// Variáveis de cálculo da função "Caminho_retroacao"
//---------------------------------------------------------------------------------------------
```

```
Float iqss_linha,idss_linha,Vqss_int,Vdss_int;//Variaveis globais da função
"Transformação"
Float Te_star,Ys_star;//Variáveis de entrada por interface com usuário
Float Ys2,Ys_calc,idq,ids,iqs,senTETAe,cosTETAe,Yds,Yqs,Te_calc,Vqs_star,
Vds_star,TETAe_star;
Float TETAe_linha,TETAe,V_star;
Float Yqss_int,Ydss_int,Omega_int;
Float Vqss_linha,Vdss_linha;//Variaveis temporarias
int botoes_int,entra=0;

//-----------------------------------------------------------------------------------------
// Constantes para cálculos da função "Caminho_retroacao"
//-----------------------------------------------------------------------------------------
const Float Ls=40,sigma=2,P=4;
const Float KI1=50,KI2=10,KI3=10,KI4=5;
const Float xm1=0.00359,xs1=0.1,xm2=0.127323954473559,xs2=0.1;
const Float xm3=0.00402010050251,xs3=0.09597989949749;

//-----------------------------------------------------------------------------------------
// Objetos criados com a ferramenta de configuração do DSPBIOS
//-----------------------------------------------------------------------------------------
extern PRD_Obj Iteracao_amostragem,Atraso_trepidacao,Escreve_LCD;
extern LOG_Obj Monitora_variaveis;

//-----------------------------------------------------------------------------------------
// Definição de nomes para posições nas faixas de endereçamento CE2 e CE3
//-----------------------------------------------------------------------------------------
//Definição das variáveis espaço de endereçamento CE2
#define lcd *(int *) 0xA0000004 //Enable_LCD no FPGA
#define controle_SV *(float *) 0xA0000008
#define controle_recorrente *(float *) 0xA000000C
#define modulo *(float *) 0xA0000010
#define angulo *(float *) 0xA0000014
#define Meio_Ts *(float *) 0xA0000018
```

```
#define Omega *(float *) 0xA000001C
#define Rs *(float *) 0xA0000020
#define Yqss *(float *) 0xA0000024
#define Ydss *(float *) 0xA0000028
#define Vqssll *(float *) 0xA000002C
#define Vdssll *(float *) 0xA0000030
#define iqssl *(float *) 0xA0000034
#define idssl *(float *) 0xA0000038
#define senha *(float *) 0xA000003C
//Definição das variáveis espaço de endereçamento CE3
#define botoes *(int *) 0xB0000000

//-----------------------------------------------------------------------------------------
// Definições e constantes para manipulação da "Interface com Usuário
//-----------------------------------------------------------------------------------------
//Variáveis e definições para escrita de caracteres no LCD
#define dado 0x0100
#define torque 0
#define periodo 1
#define fluxo 2
```

Anexo C:
Especificações de Componentes

O dispositivo TMS320C6711 é um processador digital de sinais da família TMS320C6000 da *Texas Instruments*. Possui arquitetura superescalar conhecida como "*VelociTI*", baseada em VLWI (*very large word instruction*).

Sua UCP (unidade central de processamento) é capaz de executar um pacote de instruções (palavra de programa) com até oito instruções de 32 bits por ciclo de máquina. Para executar estas instruções, sua arquitetura conta com oito unidades funcionais independentes: dois multiplicadores e seis ULAs (unidades lógicas e aritméticas). Estas oito unidades dentro da UCP estão divididas em dois grupos (ou caminhos de dados A e B), com quatro unidades funcionais em cada caminho. A organização interna deste paralelismo de execução obedece a uma estratégia de *pipeline*. O *pipeline* é composto por três estágios: "*fetch*" (carregamento de instrução), "*decode*" (decodificação de instrução), e "*execute*" (execução de instrução em até 10 fases). O usuário conta com o auxílio de ferramentas de projeto (como compilador, montador, depurador) para otimizar seu código de programação, visando um melhor uso desta estrutura de *pipeline* do hardware. A UCP ainda conta com dois bancos (arquivos internos) de registrados, tendo cada um 16 registradores de 32 bits. Embora a arquitetura deste processador seja construída para operar com representação numérica em ponto flutuante de 32 bits, são predefinidas ainda instruções de máquina para dados com 40 bits (ponto fixo) ou 64 bits (ponto flutuante, ou precisão dupla). No que se refere à velocidade de processamento, esta UCP admite uma entrada de clock com até 250MHz (ciclo de instrução mínimo de 4*n*s), ou seja, admite a realização de até 1500 MFLOPS (milhões de operações em ponto flutuante por segundo) ou 500 MMACs (milhões de multiplicações e acumulações por segundo).

No que se refere à memória, este processador de sinais trabalha com uma hierarquia composta por dois níveis de cache interna (L1 com 8 Kbytes, L2 com 64 Kbytes) e um nível de memória externa com uma faixa de endereçamento de 32 bits. A figura C1 mostra mais detalhes sobre a periferia em torna da UCP dentro deste processador.

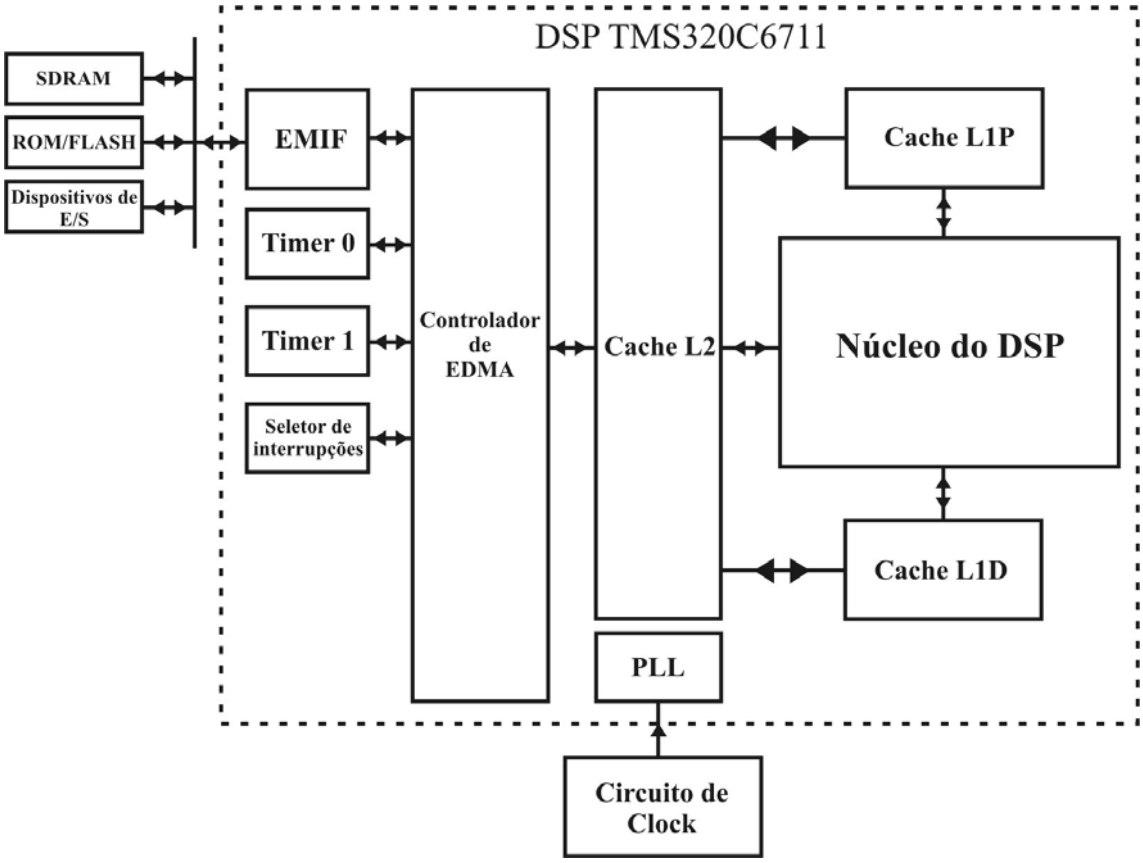


Figura C1. Arquitetura do DSP TMS320C6711.

A figura C1 mostra o encapsulamento (chip) do DSP representado pelo retângulo tracejado. Dentro deste encapsulamento estão a UCP (Núcleo do DSP) e vários periféricos controlados por ela (memórias cache, PLL, *timers*, além de outros). Como pode ser observado, a arquitetura possui memória cache L1 dividida em porção para programa (cache L1P) e porção para dados (cache L1D). No que se refere aos sinais de clock (*clock*), existe dentro do DSP um circuito PLL (*phase-locked-loop*) que supre a rede de sinais de clock interna do DSP através de uma referência fornecida por um "Circuito de *Clock*" externo. Para permitir contagens precisas estão integrados dentro do DSP dois contadores de 32 bits (*Timer 0* e *Timer 1*). Para detecção de eventos existe uma lógica de tratamento de interrupções (internas e externas) representada pelo módulo "Seletor de interrupções". Finalmente, para o interfaceamento do DSP com a hierarquia de memórias externas, existe a "EMIF" (*External Memory Interface*). A "EMIF" permite a subdivisão da faixa de endereçamento de memória externa utilizando diversos dispositivos diferentes como: memórias SDRAM, FLASH, ROM, além de outros dispositivos de E/S (entrada e saída). Neste trabalho, a faixa de endereçamento

de memória externa vista pelo DSP TMS320C6711 do protótipo foi subdividida conforme a tabela C1.

Tabela C1. Mapa de memória simplificado do DSP TMS320C6711 do protótipo.

Faixa de endereços	Nome do espaço	Dispositivos contidos no espaço
80000000h a 8FFFFFFFh	CE0	Memória SDRAM
90000000h a 9FFFFFFFh	CE1	Memória Flash (Boot do DSP)
A0000000h a AFFFFFFFh	CE2	FPGA e LCD (Interface com Usuário)
B0000000h a BFFFFFFFh	CE3	Teclado (Interface com Usuário)

Na tabela C1, o espaço CE0 (primeira linha da tabela) foi destinado a uma memória SDRAM (de 16Mbytes por 16 bits), necessária para conter as variáveis definidas pelo programa do usuário e variáveis do micro sistema operativo "DSP/BIOS" utilizado. No espaço CE1 está uma memória Flash de 4Kbytes por 8bits, contendo o programa do usuário a ser carregado para o DSP no processo de "Boot do DSP". Já no espaço de endereçamento CE2, está o FPGA Stratix 2, visto pelo DSP somente como um conjunto de variáveis de leitura e escrita modificadas também pela execução das redes neurais dentro do FPGA. Além do FPGA, encontra-se no espaço CE2 o módulo de LCD (da "Interface com Usuário"), tratado pelo DSP como um dispositivo assíncrono utilizado somente para escrita de palavras de 8 bits. No último espaço de endereçamento (o CE3), está o registrador de porta de entrada de 8 bits ligado ao teclado da "Interface com Usuário".

REFERÊNCIAS BIBLIOGRÁFICAS

[1] A. Ba - Razzouk, A. Cheriti, G. Olivier, *"Artificial neural networks rotor time constant adaptation indirect field oriented control drives"*, IEEE PESC, Baveno, pp. 701-707, June 1986.

[2] A. Bakhshai, J. Espinoza, G. Joos and H. Jin, *"A combined artificial neural network and DSP approach to the implementation of space vector modulation techniques"*, in Conf. Rec. IEEE-IAS Annu. Meeting, pp. 934-940, 1996.

[3] Felix Blaschke, *"The Principle of Field Orientation as Applied to the New Transvektor Closed-Loop Control System for Rotating-Field Machines. Adjustable Speed AC Drive Systems"*, IEEE Press, 1972.

[4] Cursino B. Jacobina and Antônio M.N. Lima, *"Estratégias de Controle para Sistemas de Acionamento com Máquina Assíncrona"*, Controle & Automação, Vol. 7(1), 1996, pp. 15–28.

[5] L. J**.** Garces, *"Parameter adaptation for the speed controlled static AC-drive with squirrel cage induction motor"*, in Proc. IEEE Ind. Appl. Ann. Conf., pp. 843, 1979.

[6] R. Ortega and D. Taoutaou, *"Indirect Field-Orientation Speed Regulation for Induction Motor is Globally Stable"*, IEEE Transactions on Automatic Control, vol. 35 (11), pp. 1238–1243, 1996.

[7] D.W. Novotny and R.D. Lorenz, *"Introduction to Field Orientation and High Performance AC Drives"*, IEEE Industry Applications Society, 2nd Edition, 1986.

[8] Paulo José C. Branco, *"Controle Vetorial de Motor de Indução por Microcomputador com Adaptação da Constante de Tempo Rotórica"*, Universidade Federal do Rio de Janeiro, Dissertação de Mestrado, 1991, Rio de Janeiro-RJ.

[9] José Andrés S. Larrea, *"Estudo Comparativo dos Métodos de Controle Vetorial para Motores de Indução"*, Universidade Federal do Rio de Janeiro, Dissertação de Mestrado, 1993, Rio de Janeiro-RJ.

[10] Takahashi, I., Noguchi, T., *"A new quick response and high efficiency control strategy of an induction motor"*, IEEE IAS Meeting, pp. 496-502, 1985.

[11] M. Depenbrock. *"Direct Self-Control (DSC) of Inverter-Fed Induction Machine"*, IEEE Trans. On Power Electronics, vol. 3, nº 4, October 1988, pp. 420-429.

[12] B. K. Bose, *"Sliding Mode Control of Induction Motor"*, Conf. Rec. IEEE Indust. Applicat. Annu. Meeting, 1985, pp. 479–486.

[13] E.Y.Y. Ho, P.C. Sen, *"Decoupling control of induction motor drives"*, IEEE Trans. on Ind. Electron., vol. 35**,** nº 2**,** pp. 253 - 262**,** May, 1988**.**

[14] Rogelio Soto and K. S. Yeung, *"Sliding-Mode Control of an Induction Motor Without Flux Measurement"*, IEEE Transactions on Industry Applications, Vol. 31(4), 1995, pp. 744–750.

[15] L. A. Zadeh, *"Fuzzy Sets"*, Information Control, vol. 8, pp. 338-353, 1965.

[16] E. H. Mamdani and S. Assilian, *"An experiment in linguistic synthesis with a fuzzy logic controller"*, Int. Journal of Man-Machine Studies, vol. 7, pp. 1-13, 1975.

[17] Goldberg, D. E., *"Genetic algorithms in search, optimization and machine learning"*, Addison-Wesley, reading, MA, 1989.

[18] G. Franceschini, A. Piazzi and C. Tassoni, *"A genetic algorithm approach to design flux observers for induction servomotors"*, IECOM, Bologna, 1994.

[19] Narendra K, Parthasarathy K, *"Identification and Control of Dynamical Systems using Neural Networks"*, IEEE Trans. on Neural Networks, March 1990, Vol. 1, No.1, pp 4-27.

[20] Filippetti, F., Tassoni, C., Franceschini, G., Vas, P., *"Integrated condition monitoring and diagnosis of electrical machines using minimum configuration artificial intelligence"*, EPE'97, Trondheim, pp. 2983-2988, 1997.

[21] J. J. Hopfield, *"Neurons with graded response have collective computational properties like those of two-state neurons"*, Proceedings, National Academic Sciences, USA, 81, 1984, pp.3088-3092.

[22] M. Sugeno and G. Kang, *"Structure identification of fuzzy model"*, Fuzzy Sets and Systems, vol. 28, pp. 15-33, 1988.

[23] J. Si and A. N. Michel, *"Analysis and synthesis of discrete-time neural networks with multi-level threshold functions"*, Proceedings of the IEEE International Symposium on Circuits and Systems, The Westin Stamford and Westin Plaza, Singapore, June 1991, pp. 1461-1464.

[24] J. Yuh and R. W. Newcomb, *"Circuits for multi-level nonlinealrities"*, Proceedings of the International Joint Conference on Neural Networks, Baltimore, MD, Vol.11, June 1992, pp.27-32.

[25] João, O. P. Pinto, Bimal K. Bose, e Luiz Eduardo Borges da Silva. (2001). *"A Stator-Flux-Oriented Vector-Controlled Induction Motor Drive With Space-Vector PWM and Flux-Vector Synthesis by Neural Networks"*. IEEE Trans. Ind. Aplications.

[26] João, O. P. Pinto, Bimal K. Bose, e Luiz Eduardo Borges da Silva, *"A Neural-Network-Based Space-Vector PWM Controller for Voltage-Fed Inverter Induction Motor Drive"*. IEEE Trans. on Ind. Applications, vol. 36, nº 6, nov/december 2000, pp. 1628-36.

[27] K. Shimane, S. Tanaka, and S. Tadakuma, *"Vector controlled IM using neural networks,"* Trans. IEEE, vol. 113-D, no. 10, pp. 1154–1161, 1993.

[28] Lazhar Ben-Brahim, Susumu Tadakuma and Alper Akdag, *"Speed Control of Induction Motor Without Rotational Transducers"*, IEEE Transactions on Industry Applications, Vol. 35, No. 4, July/August 1999, pp. 844-850.

[29] W. Leonhard, *"Field-Orientation for Controlling AC Machines-principle and Application"*. Power Electronics and Variable-Speed Drives, Third International Conference on, 13-14 Jul, 1988, pp. 277-282.

[30] W. Leonhard, *"Adjustable-speed AC Drives"*. Proceeding of the IEE, vol. 76, nº 4, April, 1988, pp. 455-471.

[31] Li Zhang, K M Hasan, *"Neural network aided energy efficiency control for a field-orientation induction machine drive"*, Ninth international Conference on Electrical Machines and Drives, Conference Publication No. 468, IEE, 1999, pp. 356-360.

[32] M. A. C.Maher, S. P. Deweerth, M. A. Mahowald, and C. A. Mead, *"Implementing neural architectures using analog VLSl circuits,"* IEEE Transactions on Circuits and Systems, Vo1.36, No.5, 1989, pp.643-652.

[33] Bimal Gisuthan, T. Srikanthan and K.V.Asari, *"A High Speed Flat CORDIC Based Neuron with Multi-Level Activation Function for Robust Pattern Recognition"*, IEEE, 2000, pp. 87-94.

[34] Chen F.C., *"Back-propogation neural network for nonlinear self-tuning adaptive control"*, Proc. IEEE Int. Symposium on Intelligent Control, Sept. 1989, Albany, New York, pp. 274-279.

[35] F. Filippetti, G. Franceschini, C. Tassoni, *"Neural network aided on - line diagnostics of induction motor rotor faults"*, IEEE Trans. Ind. Applicat, vol 3, nº 4, pp. 892-899, June 1995.

[36] M. Furman and A. Abidi, *"CMOS analog IC implementing the back propagation algorithm,"* Neural Networks, I , Sup. I , 1988, pp. 381.

[37] Nguyen D, Widrow B, *"The truck-backer-upper: an example of self-learning in neural networks"*, Proc. Int. Joint Conf. on Neural Network, Washington DC, Vol. 2, June 1989, pp. 11357-11363.

[38] T. Fukuda and T. Shibata, *"Theory and applications for neural networks for industrial control systems"*, IEEE, Trans. Ind. Electron., vol. 39, pp. 472–489, Dec. 1992.

[39] Y. Takahashi, *"Adaptive control via neural networks"*, J. SICE, vol. 29, no. 8, pp. 729–733, Aug. 1990.