Controlador fuzzy programado em processador NIOS II embarcado em FPGA

LEANDRO ELIAS BASMAGE PINHEIRO MACHADO

CAMPO GRANDE

2011

Controlador fuzzy programado em processador NIOS II embarcado em FPGA

LEANDRO ELIAS BASMAGE PINHEIRO MACHADO

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

_____

Edson Antonio Batista, Dr.
Orientador

_____

Luciana Cambraia Leite, Dra.
Co-Orientadora

_____

Luciana Cambraia Leite, Dra.
Coordenadora do Programa de Pós-Graduação em Engenharia Elétrica

Banca Examinadora:

_____

Edson Antonio Batista, Dr.
Presidente

_____

Mauro Conti Pereira, Dr.

_____

Nicolau Pereira Filho, Dr.

À Deus, minha família e minha namorada que sempre estiveram ao meu lado nessa etapa.

AGRADECIMENTOS

Aos professores Dr. Edson Batista e Dra. Luciana Cambraia Leite, meu orientador e co-orientadora, respectivamente, pelo apoio e pelas contribuições que foram fundamentais para o desenvolvimento deste trabalho.

Agradecimento especial à UCDB por ceder seu laboratório e kit de elétrica no decorrer do desenvolvimento do projeto e à FUNDECT pelos auxílios financeiros concedidos. Não podemos deixar de agradecer também os professores Dr. Mauro Conti Pereira e Dr. Nicolau Pereira Filho, membros da qualificação, por suas valiosas sugestões e recomendações.

À equipe de estudos da UCDB e alunos do PPGEE/UFMS, especialmente aos meus grandes amigos e incentivadores João Carlos Siqueira, Luis Henrique Corbelino e Carlos Vinícius Vanti e Flávio Palmiro.

Aos amigos que tive a honra de compartilhar esta etapa de mestrado, Marcelo Cristiano e Guilherme Alencar.

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

CONTROLADOR FUZZY PROGRAMADO EM PROCESSADOR NIOS II EMBARCADO EM FPGA

LEANDRO ELIAS BASMAGE PINHEIRO MACHADO

Maio/2011

Orientador: Dr. Edson Antonio Batista.
Co-orientadora: Dra. Luciana Cambraia Leite.
Área de Concentração: Energia.
Palavras-chave: Lógica Fuzzy, Inteligência Artificial, NIOS II, FPGA.
Número de Páginas: 67.

RESUMO: Neste trabalho foi realizada a inserção de técnicas de controle fuzzy no processador NIOS II embarcado em FPGA. A aplicabilidade deste hardware de controle está no posicionamento angular de uma válvula para manter a estabilidade da corrente de um motor de indução trifásico 220 V e 1/3 CV. Trata-se de um protótipo que simula uma centrifuga presente na etapa de fabricação de açúcar, em que o magma deve ser centrifugado para dar origem ao açúcar. Este processo eventualmente sofre paradas pelo fato do motor ficar em estado de sobre-corrente, fazendo-o desarmar e ocasionando paradas indesejáveis. O monitoramento da corrente de alimentação de uma centrífuga em uma usina de produção de açúcar é necessário para que seja possível fazer o controle do ângulo de abertura da válvula que controla o fluxo de entrada do material a ser centrifugado. Outro ponto de destaque é o desenvolvimento de um *template* específico para controladores fuzzy, gerado em linguagem C, que pode ser embarcado em FPGA. Neste *template*, denominado controlador fuzzy, os projetistas podem ajustar as funções de pertinência ou regras conforme a aplicação e utilização em outros hardwares.

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

FUZZY CONTROLLER PROGRAMMED INTO NIOS II PROCESSOR EMBEDDED IN FPGA

Leandro Elias Basmage Pinheiro Machado

Maio / 2011

Advisor: Dr. Edson Antonio Batista.
Co-advisor: Dra. Luciana Cambraia Leite.
Area of Concentration: Energy.
Keywords: Fuzzy Logic, Artificial Intelligence, NIOS II, FPGA.
Number of Pages: 67.

ABSTRACT: This work aims the inclusion of fuzzy control techniques in the NIOS II processor embedded into a FPGA. The applicability of this control hardware is the angular position of a valve to maintain the stability of a current induction motor of 220 V and 1/3 CV. This is a prototype that simulates a spin at this stage of sugar production, where the magma must be centrifuged to give rise to sugar. This process eventually may suffer because the engine stops in the state of overcurrent, causing unwanted downtime, monitoring the supply current of a centrifuge plant in a production of sugar is necessary to be able to control the opening angle of the valve controlling the inflow of material to be centrifuged. Another contribution is the development of a specific template for fuzzy controllers, generated in C language, which can be embedded into FPGA. In this template, called fuzzy controller, designers can adjust the membership functions or rules according to the application and can also be used in other hardwares. Another aspect explored in this work is the use of FPGA in industrial environment.

SUMÁRIO

Lista de Figuras

Lista de Símbolos

ART: Açúcares redutores totais.

CA: Corrente alternada.

CC: Corrente contínua.

CLB: *Configurable Logical Blocks.*

DSP: *Digital Signal Processor.*

E/S: Entrada/Saída.

FPGA: *Field Programmable Gate Array.*

I_n: Corrente nominal.

MIT: Motor de Indução Trifásico.

PLD: *Programmable Logic Device.*

PLL: *Phase Locked Loop*.

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

RMS: *Root Mean Square* (Valor eficaz)

RTC: Rendimento total da cana.

SOPC: *System on a Programmable Chip.*

SDRAM*: Synchronous Dynamic Random Access Memory*

UART: *Universal Asynchronous Receiver/Transmitter.*

VHDL: *VHSIC Hardware Description Language.*

VHSIC: *Very High Speed Integrated Circuit*.

CAPÍTULO 1 – INTRODUÇÃO

1.1. Introdução

Este capítulo tem como intuito fornecer os conceitos que envolvem o setor de açúcar e álcool, o qual foi escolhido para testar as funcionalidades do módulo embarcado e os princípios de funcionamento de uma das etapas do processo, chamado centrifugação contínua.

Nesta etapa a matéria prima envolvida já sofreu muitas alterações, tendo agregado um valor econômico considerado, sendo assim, perdas nesta etapa são de uma importância relevante para a usina. Inicialmente, são apresentados alguns conceitos sobre o setor sucro-alcooleiro, seguido dos objetivos do trabalho, os trabalhos correlatos e por último a descrição dos capítulos.

1.2. Setor Sucro-Alcooleiro

Atualmente as empresas do setor sucroalcooleiro trabalham em função de índices de: disponibilidade industrial, ART da cana (açucares redutores totais, que seriam glicose, frutose e sacarose) e perdas de açúcares durante o processo [1]. Segundo informações de usinas do setor em Mato Grosso do Sul, há um boletim que mensura todos esses índices, e com a contribuição deles é gerado o RTC (rendimento total da cana). Para uma planta com boa eficiência, este índice tem que estar entre 93% – 95% pois há perdas indeterminadas durante o processo. Em média as quatro maiores perdas nas usinas são:

→ Bagaço (2% – 4 %)

→ Destilaria (1%-1,5%)

→ Torta (0,8%-1,5%)

→ Indeterminadas (0,5% - 2%)

Para reduzir as perdas e melhorar a eficiência da planta, muitos processos tiveram que ser automatizado como a evaporação e o tratamento de caldo, processos que antecedem a centrifugação contínua.

Este trabalho descreve sobre a automação de um protótipo que simula as funções de uma centrífuga contínua, a qual trabalha a massa cristalizada que provém dos cozedores e a

transforma em cristais de açúcar que são misturados com xarope dando origem ao magma. O magma tem como objetivo fazer o pé para o próximo cozimento, ou seja, qualquer perda neste processo tem um significado relevante, pois a matéria que está no processo já sofreu muitas alterações com auxílio de insumos e vapor, tendo assim agregado um valor econômico alto e, conseqüentemente, as perdas são mais consideráveis.

O controle do processo visando uma melhor eficiência foi feito através da corrente do motor da centrífuga, utilizada como variável de entrada, e com o auxílio do conhecimento do operador as funções de pertinências que representam esta variável foram configuradas.

Este controle tem como principal objetivo tornar o processo de centrifugação totalmente automatizado minimizando as perdas, tendo como atuador uma válvula de controle que é responsável pelo controle de vazão de magma a ser centrifugado. Como foi utilizada a técnica fuzzy, conhecimentos técnicos de operadores do setor sucro-alcooleiro foram requeridos.

1.3. Processo de Centrifugação dos Cristais de Açúcar

A separação dos cristais do licor mãe, até por volta da metade do século XIX, era realizada somente por gravidade, em moldes cônicos perfurados, em que, após a drenagem do mel era obtido o açúcar. Porém, o tempo gasto nesta operação era maior. Assim, com o objetivo de diminuir o tempo gasto no processo e aumentar sua eficiência, surgiu daí a idéia de se utilizar a força centrífuga para a remoção do mel envolvente dos cristais [1].

O princípio de funcionamento das centrífugas está baseado no emprego da força centrífuga e da gravidade entre si. A máquina em movimento faz com que, através da força gerada, a massa cozida que atinge o fundo da centrífuga procure as paredes laterais do cesto, atravessando os orifícios da tela que o reveste, retendo, assim, os cristais no cesto [1].

1.3.1 Centrifugação Contínua

Seu funcionamento inicia-se colocando a turbina em movimento de trabalho, abre-se a válvula de registro deixando a massa fluir para o centro da câmara de aceleração. Neste ponto a aceleração é praticamente zero e vai, gradativamente, subindo à medida que foge do centro. A partir do centro do fundo da câmara de aceleração, a massa move-se tomando as

paredes por onde sobe em camada fina, até passar para a área perfurada, onde começa a separar o mel. Daí, o mel passa pela tela, é recolhido pelo cone interno, passa para a tubulação externa pela parte superior e, posteriormente, é descarregado para o tanque de mel. O açúcar livre do licor envolvente sobe pelas paredes perfuradas até ser descarregado na câmara de açúcar em um fluxo contínuo [1]. O fluxograma do processo de fabricação do açúcar pode ser visualizado na Figura 1.

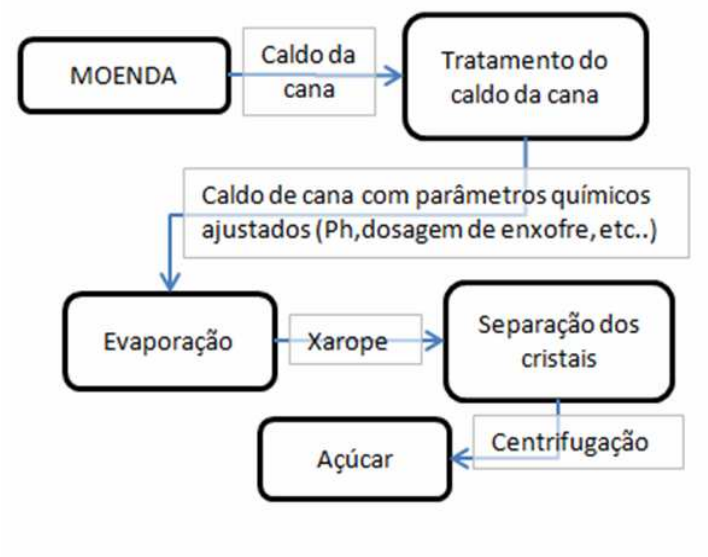


Figura 1 - Etapas da fabricação do açúcar.

1.4. Objetivos Gerais

Dentre os objetivos gerais e específicos do trabalho, destacam-se:

Inserir técnicas de controle fuzzy em FPGA, desenvolvendo um protótipo de centrifugação contínua, o qual representa um dos processos de uma fábrica de açúcar;
Desenvolver circuitos de condicionamento de sinais;
Utilizar a instrumentação (sensores e atuadores) com valores operacionais semelhantes aos utilizados nas indústrias de açúcar;
Gerar uma biblioteca que possa ser utilizada em outras aplicações que necessitam de controle fuzzy;
Utilizar FPGA em ambientes indústrias;

O objetivo final é apresentar um controlador fuzzy genérico (baseado em regras) programado em ambiente NIOS II validado pelo controle de posição de um servo motor através da corrente da um motor de indução.

1.5. Trabalhos Correlatos

Alguns trabalhos apresentam a inserção do sistema fuzzy integral totalmente embarcado, visando os ajustes dos parâmetros de um controlador PI de um controle direto de torque (DTC) de motores de indução (MI), como em [2].

Destaca-se também o trabalho em que foi realizado o desenvolvimento também integral de um controlador, nesse caso neuro-fuzzy, auto-ajustável em um DSP, utilizando-se a ferramenta Simulink do *software* Matlab para implentar o algoritmo juntamente com o toolbox *real–time*; o *software* central-*desk* gerou automaticamente o código do programa para o DSP [3].

Um exemplo de processamento realizado por um chip dedicado ao processo de inferência *fuzzy* utilizado em uma aplicação exclusiva pode ser encontrado no trabalho proposto por [4].

Outro trabalho interessante em que foi desenvolvida uma metodologia de implementação de algoritmos com estratégias fuzzy para sistemas embarcados em processadores digitais de sinais (DSP) pode ser encontrado em [5].

Neste contexto, o presente trabalho tem como característica o desenvolvimento de um protótipo que emule o processo de centrifugação contínua utilizando técnica de controle fuzzy e tecnologia FPGA. O protótipo consiste de um módulo embarcado contendo um hardware gerado a partir do SOPC Builder e de um *firmware* (conjunto de instruções lógicas programadas diretamente no hardware) programado no ambiente NIOS II IDE, onde foi armazenado o algoritmo do controlador fuzzy.

1.6. Descrição dos Capítulos:

Esta dissertação possui seis capítulos adicionais e a descrição de cada um deles segue abaixo:

No capítulo 2 é feita uma revisão teórica dos aspectos abordados pelo autor, conceitos de lógica fuzzy e do controlador fuzzy.

Os conceitos e definições sobre a tecnologia FPGA são feitas no capítulo 3. Este capítulo trata sobre os principais recursos utilizados para esse dispositivo, como exemplo, a ferramenta SOPC *Builder* e o processador NIOS II, e ainda descreve-se, brevemente, sobre condicionamento de sinais.

O desenvolvimento do módulo embarcado e a parte de condicionamento de sinais são descrita no capítulo 4, no qual é demonstrada a elaboração tanto do *firmware* quanto do *hardware*, além do desenvolvimento da placa de condicionamento de sinal utilizada.

As simulações do circuito proposto, incluindo um motor que representa o funcionamento da centrífuga e com ajustes em seus ganhos, foram realizadas no Simulink/Matlab e são apresentadas no capítulo 5.

Para finalizar, os resultados experimentais, comparações e conclusões finais do trabalho são demonstrados nos dois últimos capítulos, 6 e 7, respectivamente.

CAPÍTULO 2 - LÓGICA FUZZY

2.1 Introdução

Existem diversas técnicas inteligentes que são utilizadas no controle de processos industriais, como por exemplo, redes neurais, lógica fuzzy, Ziegler-Nichols, Takagi-Sugeno, entre outros. Neste trabalho utilizou-se a lógica fuzzy para desenvolver o controle das atividades do protótipo. Inicialmente serão abordados alguns conceitos da lógica fuzzy e do controlador fuzzy.

A lógica fuzzy (lógica nebulosa) permitiu o desenvolvimento de controladores fuzzy, no qual funções de pertinência convertem estratégias de controle baseadas no conhecimento do especialista ou do operador, comportando-se de forma similar a um raciocínio dedutivo, possibilitando estratégias de tomadas de decisão em problemas complexos ou que exijam tempo de solução reduzido, apresentando a opção de implementar um controlador baseado no conhecimento do especialista [6].

Em um sistema de automação industrial, as técnicas de controle são executadas em computadores e/ou em controladores específicos.

2.2 Lógica Fuzzy

Em 1965, o professor Loft Zadeh introduziu a lógica fuzzy no contexto científico com sua publicação do artigo "*Fuzzy Sets*" no *Journal Information and Control*. Seu pensamento era que a teoria usual de conjuntos era demasiadamente rígida para contemplar fenômenos cotidianos do dia a dia [7].

A lógica fuzzy é diferenciada por incorporar forma humana de pensar em um sistema de controle, podendo assim ser usado nos controles de processos variados, principalmente aqueles com características não lineares, pois os mesmos não possuem uma teoria geral para solução analítica ou, em alguns casos, até impossíveis fazer a modelagem pelos métodos clássicos que adotam o valor falso (0) ou o valor verdadeiro (1) [8].

A lógica fuzzy caracteriza-se por ser multivalente, ou seja, não existe apenas o verdadeiro (1) e falso (0), mas uma infinidade de valores, assegurando que a verdade é uma

questão de ponto de vista ou de graduação, definindo o grau de veracidade em um intervalo numérico [0,1] [9].

Os controladores fuzzy utilizam das experiências do especialista da área para estabelecer um conjunto de regras condicionais pelo método heurístico que serão utilizadas na implementação do referido controle.

Dentre as vantagens da Lógica fuzzy pode-se citar [10]:

- Simplificação do modelo do processo;
- Melhor tratamento das imprecisões inerentes aos sensores utilizados;
- Facilidade na especificação das regras de controle, em linguagem próxima à natural;
- Satisfação de múltiplos objetivos de controle e;
- Facilidade de incorporação do conhecimento de especialistas humanos.

2.2.1 Método Heurístico

Consiste em ter uma experiência e com ela realizar uma tarefa com regras práticas e estratégias já utilizadas. Trata-se de uma implicação lógica de forma que [9]:

Se condição Então conseqüência
Ou
Se condição Então ação

2.2.1.1 Base de Regras

A base de regras tem por objetivo representar, de forma sistemática, a maneira como o controlador gerenciará o sistema [11].

As condições, também chamadas de antecedentes, são associadas com as entradas do controlador fuzzy e formam a parte das regras representada à esquerda, enquanto as conseqüências, também conhecidas como ações, estão associadas às saídas dos controladores [9].

Para exemplificar a utilização da base de regras, toma-se como exemplo o controle de trânsito, em que se estabelece uma relação da circulação de carros com o semáforo, produzindo as seguintes regras [9]:

SE o trânsito está CONGESTIONADO ENTÃO mantenha semáforo verde por MAIS TEMPO.
SE o trânsito está LIVRE ENTÃO mantenha o semáforo verde por MENOS TEMPO
SE o trânsito está NORMAL ENTÃO mantenha o semáforo verde em tempo NORMAL.

Em que os termos CONGESTIONADO e MAIS TEMPO representam conjuntos fuzzy, CONGESTIONADO é uma função que define o grau de densidade do trânsito, enquanto MAIS TEMPO é uma função que define o grau de duração do tempo de operação do semáforo.

Aparentemente simples, o exemplo dado é capaz de melhorar ou até manter estável o trânsito em determinado local, desde que os conjuntos fuzzy envolvidos tenham uma definição harmônica ou combinada com a realidade. Para cada estado do processo é relacionada uma ação de controle, demonstrando que a base de regras fuzzy é bastante intuitiva.

Neste caso, não há uma formulação matemática para a solução do problema e sim o conhecimento do operador de trânsito. Portanto é este conhecimento, o da experiência, que as regras fuzzy representam [12].

2.3 Controlador Fuzzy

Um controlador fuzzy é constituído dos seguintes blocos funcionais [7]:

- Interface de fuzzificação;
- Base de conhecimento;
- Lógica de tomada de decisões;
- Interface de defuzzificação.

Esta estrutura de controlador representa a transformação que ocorre dos números reais para o domínio dos números fuzzy. Para tal transformação utiliza-se um conjunto de

inferências fuzzy para as tomadas de decisões, e por fim, uma transformação inversa do domínio fuzzy para o domínio real, a defuzzificação.

2.3.1 Interface de fuzzificação

Os valores discretos (não fuzzy) das variáveis de entrada geralmente são provenientes de sensores (grandezas físicas) ou de dispositivos de entrada computadorizados. A fuzzificação faz com que o valor "real" da variável de entrada converta ao seu valor correspondente fuzzy, ou seja, faz a "translação" da variável medida do domínio real para o domínio fuzzy. Isto é feito para que o mesmo expresse o valor das incertezas existentes na medida realizada [13].

A interface de fuzzificação utiliza funções de pertinência contidas na base de conhecimento, convertendo os sinais de entrada em um intervalo [0,1], podendo estar associado a rótulos lingüísticos como, por exemplo, "MT" (MAIS TEMPO) descrito no tópico base de regras.

2.3.2 Base de Conhecimento

A base de conhecimento representa o modelo do sistema a ser controlado consistindo de:

- Base de dados ou funções de pertinências → fornece as definições numéricas necessárias às funções de pertinência usadas no conjunto de regras fuzzy;
- Base de regras fuzzy lingüísticas → caracteriza os objetos de controle e a estratégia de controle utilizada por especialistas na área, por meio de um conjunto de regras de controle, em geral lingüísticas.

> *Funções de pertinência representam os aspectos fundamentais de todas as ações teóricas e práticas de sistemas fuzzy. Uma função de pertinência é uma função numérica gráfica ou tabulada que atribui valores de pertinência fuzzy para valores discretos de uma variável, em seu universo de discurso. O*

> *universo de discurso de uma variável representa o intervalo numérico de todos os possíveis valores reais que uma variável específica pode assumir [9].*

O tipo e a quantidade de funções de pertinência usadas em um sistema dependem de [14]:

- Precisão;
- Estabilidade;
- Facilidade de implementação;
- Manipulação;
- Manutenção.

Um número prático de funções de pertinência é algo entre dois (2) e sete (7) e, quanto maior o número de conjuntos, maior a precisão, entretanto, a demanda computacional é mais significativa [9].

Outro fator que interfere na precisão é o grau de superposição entre as funções de pertinência fuzzy. Em [9] demonstra-se que este grau deve estar em um mínimo de 25% e um máximo de 75%, sendo 50% um compromisso razoável, pelo menos para os primeiros testes de um sistema em malha fechada.

Os formatos mais freqüentemente encontrados são triângulos e trapezóides, devido suas facilidades de implementação em hardware. As funções de pertinência não precisam ser simétricas ou espaçadas igualmente: cada variável pode ter um conjunto de funções de pertinência diferente, lembrando que os conjuntos fuzzy devem abranger todo o universo de discurso (eixo X), mapeando as pertinências de uma entrada no intervalo de 0 a 1 do eixo Y [12].

2.3.3 Lógica de Tomada de Decisões

A lógica de tomada de decisões, acoplada à estrutura de inferência da base de regras, utiliza ligações fuzzy para simular ações de execução humana. Gera ações de controle (saída) como conseqüências de um conjunto de condições de entrada (antecedentes) [15].

2.3.4 Funções da Interface de Defuzzificação

O bloco defuzzificador tem a função de converter os valores fuzzificados em valores reais que melhor representam tal conjunto, fazendo o inverso do bloco de fuzzificação.

Esta função é necessária apenas quando a saída do controlador tiver de ser interpretada como uma ação de controle discreta. Existem sistemas que não exigem defuzzificação porque a saída fuzzy é interpretada de modo qualitativo, por exemplo, na fabricação de vinho o padrão de saída fuzzy é comparado com aqueles que correspondem a certos aspectos qualitativos gerados por experimentadores humanos que usam o olfato e o paladar para avaliar a qualidade.

Para selecionar o método apropriado de defuzzificação, pode-se utilizar tanto métodos de centróide como os valores máximos que ocorrem da função de pertinência resultante [11].

2.3.5 Defuzzificação por Centro de Área (C-o-A)

Este método consiste em calcular o centro geométrico da área composta que representa o termo de saída fuzzy (μ_{out}), composto pela união de todas as contribuições de regras. O centróide é um ponto que divide a área de μ_{out} em duas partes iguais [16].

O cálculo do centróide da área pode ser visto na equação (2.1).

$$u = \frac{\sum_{J=1}^{N} u_l \, \mu_{OUT}(u_l)}{\sum_{J=1}^{N} \mu_{OUT}(u_i)} \qquad (2.1)$$

Em que $\mu_{OUT}(u_i)$ é a área de uma função de pertinência (como, por exemplo, CONGESTIONADO) modificada pelo resultado da inferência fuzzy (como, por exemplo, 0.9); u_i é a posição do centróide da função de pertinência individual (CONGESTIONADO).

A expressão (2.1) calcula o centróide composto, em que há a contribuição das regras ativadas [16]. Na Figura 2 pode-se observar a composição e defuzzificação pelo centróide de um sistema com duas (2) regras [7].

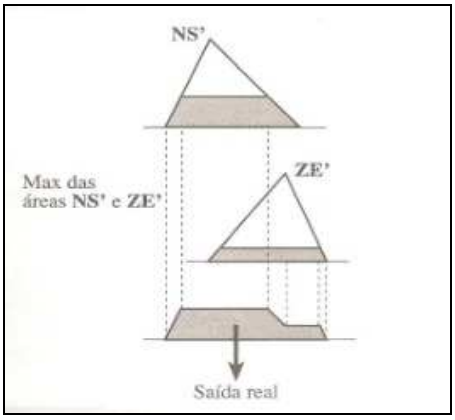


Figura 2 - Composição e defuzzificação pelo centróide em um sistema de duas regras.

Este método apresenta problemas quando as funções de pertinência não possuem sobreposição, pois não haveria o centro geométrico da figura. Outro fator de desvantagem é quando uma mesma saída fuzzy contenha o mesmo valor de outra saída fuzzy, assim haverá uma sobreposição de áreas que não é devidamente contabilizada, tendo um esforço computacional maior para o cálculo, devido à integração numérica [9].

2.3.6 Vantagens do Controlador Fuzzy Baseado em Regras

Por terem um grande número de vantagens práticas, os controladores fuzzy se tornaram muito popular nas configurações de softwares de sistema de desenvolvimento e dentre elas pode-se citar [9]:

- Fácil compreensão pelos responsáveis pela manutenção, pois sendo baseado no senso comum, o resultado de cada regra pode ser facilmente interpretado.

- Todas as funções de controle associadas a uma regra podem ser testadas individualmente, facilitando assim a manutenção, além de que a simplicidade das regras pode ser interpretada até por pessoal menos treinado.
- Processamento paralelo, em que se pode completar a tarefa de processamento sem envolver muitos cálculos, e com isso a velocidade de processamento é aumentada.

CAPÍTULO 3 – TECNOLOGIA FPGA E CONDICIONAMENTO DE SINAIS

3.1 Introdução

Tratando-se de um cenário industrial, os controladores lógicos programáveis (CLPs) são os dispositivos mais usados, pois executam o controle de processo básico e processual desde 1960.

Devido a configuração computacional dos CLPs, estes equipamentos não suportam técnicas de controle complexas, sendo necessário a utilização de um computador para realizar esta atividade. Uma alternativa para este problema é a utilização de dispositivos lógicos programáveis. [6].

Os hardwares reconfiguráveis ou PLDs (*Programmable Logic Devices*) são dispositivos em que sua configuração pode ser definida pelo usuário através de software, não havendo a necessidade de alteração física. Entre esses hardwares, pode-se destacar o FPGA (*Field Programmable Gate Arrays*) [6].

O FPGA é um chip que suporta a implementação de circuitos lógicos, programáveis, desenvolvidos com a arquitetura de matrizes simétricas, apresentando boa flexibilidade em seu roteamento interno por possuir canais de conexão, tanto na vertical quanto na horizontal [17].

A disponibilidade abundante de recursos de memória fizeram do FPGA um dispositivo adequado para implementação de sistema embarcado SOPC (*System on a Programmable Chip*) e a inserção de processadores [18] [19].

3.2 Estrutura

Os FPGAs (*Field Programmable Gate Arrays*) são circuitos programáveis, desenvolvidos com a arquitetura de matrizes simétricas. Sua configuração interna pode ser dividida em três grupos, conforme apresentado na Figura 3:

- Blocos lógicos configuráveis – CLBs (*Configurable Logical Blocks*).
- Blocos de entradas e saídas ou E/S - IOBs (*Input Output Blocks*).

- Chaves de interconexões - *Switch Matrix.*

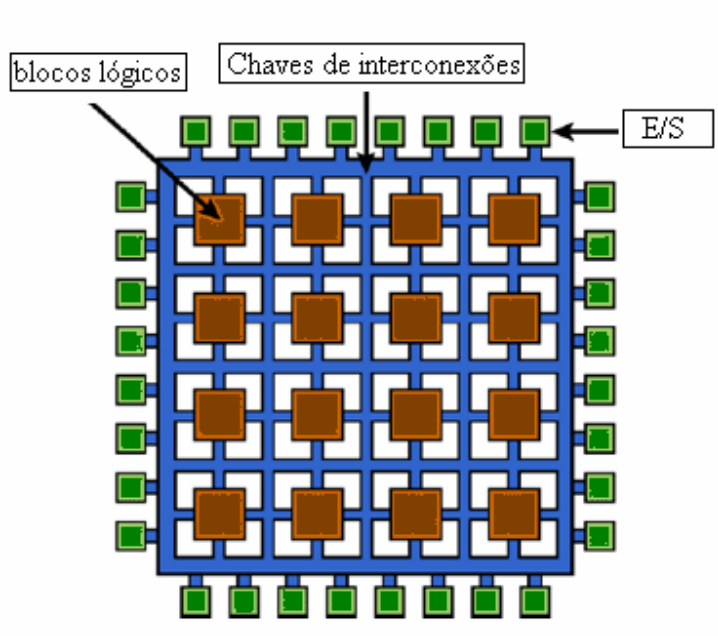


Figura 3 - Arquitetura de um FPGA.

Os CLBs permitem a construção de circuitos lógicos apropriados para cada aplicação e, usualmente, são construídos com dois a quatro flip-flops e lógica combinatória. Sua estrutura é apresentada na Figura 4 [20].

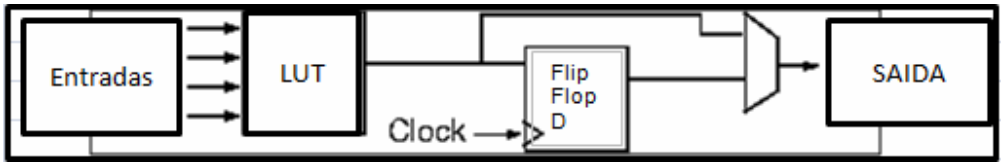


Figura 4 - Estrutura de um CLB.

Já os IOBs são blocos responsáveis pela comunicação do FPGA com o exterior. Por meio de sua configuração, podem funcionar como: entrada, saída, bidirecionais ou coletor-aberto.

E por fim, o grupo de Switch Matrix, os quais são incumbidos de conectar os CLBs e IOBs em redes apropriadas utilizando trilhas. A todo este procedimento se dá o nome de roteamento como pode ser visto na Figura 5.

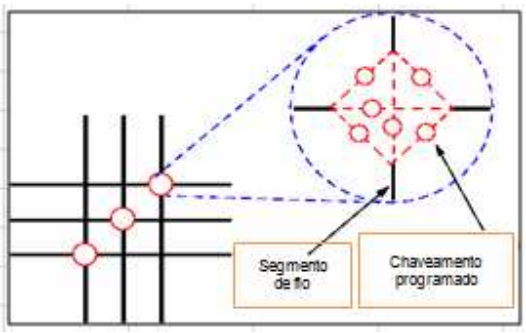


Figura 5 - Estrutura interna de um Switch Matrix.

Toda a estrutura lógica do FPGA pode ser descrita pelo usuário, ao contrário de outros chips, em que toda configuração é programada pelo fabricante, ou ainda em circuitos não programáveis, nos quais a estrutura é definida por meio de sua construção física [21].

Dentre as famílias de FPGAs oferecidas pela Altera, encontra-se a Cyclone II, que oferece ao usuário 68416 elementos lógicos e até 622 pinos de entrada e saída, além de memória embutida que pode chegar a 1,1 Mb [22], podendo ser observada na Figura 6.

Figura 6 - Altera Cyclone II.

3.2.1 SOPC BUILDER

SOPC Builder é uma ferramenta de desenvolvimento que gera um hardware, podendo incluir processadores, periféricos e memórias, permitindo definir e gerar um sistema completo sobre um chip programável [23].

Depois de especificado os componentes no SOPC Builder que serão utilizado no sistema, é gerado a interligação lógica entre eles automaticamente podendo gerar hardware descrito em VHDL (*VHSIC Hardware Description Language*) ou Verilog [23].

Após gerar o hardware no SOPC Builder, é necessário inserir as pinagens de todos os componentes para então fazer a síntese no Quartus II e o download no FPGA.

3.2.1.1 Interface Avalon

É a interface de conexão de dispositivos utilizada no SOPC Builder, dividida em seis tipos [24]:

- *Avalon Memory Mapped*: Interface típica de leitura/escrita de conexões mestre-escravo como microprocessadores, memórias, UARTS e *timers* possuindo interfaces mestre–escravo conectados ao barramento *system interconnect fabric.*
- *Avalon Streaming*: Interface que suporta fluxo unidirecional de dados, alta largura de banda e baixa latência cujas aplicações típicas incluem streams multiplexados, pacotes e dados DSP.
- *Avalon Memory Mapped Tristate*: Interface de leitura/escrita baseada em endereço com suporte para periféricos “off-chip”. Essa interface permite que os dados e os pinos de endereço possam ser compartilhados entre vários dispositivos tri-state, muito importante em sistemas que têm multiplos dispostitivos de memórias externas e pinos limitados.
- Avalon *Clock*: Interface que envia ou recebe sinais de clock e reset para sincronizar interfaces.
- Avalon *Interrupt*: Interface que permite componentes sinalizar em eventos prioritários para outros componentes.
- Avalon *Conduit*: Interface que permite agrupar conjunto arbitrário de sinais para que sejam exportados para fora do sistema *SOPC BUILDER*, consistindo de sinais de entrada e saída.

Podem-se observar na Figura 7 as interfaces Avalon de um projeto com processador NIOS II [24].

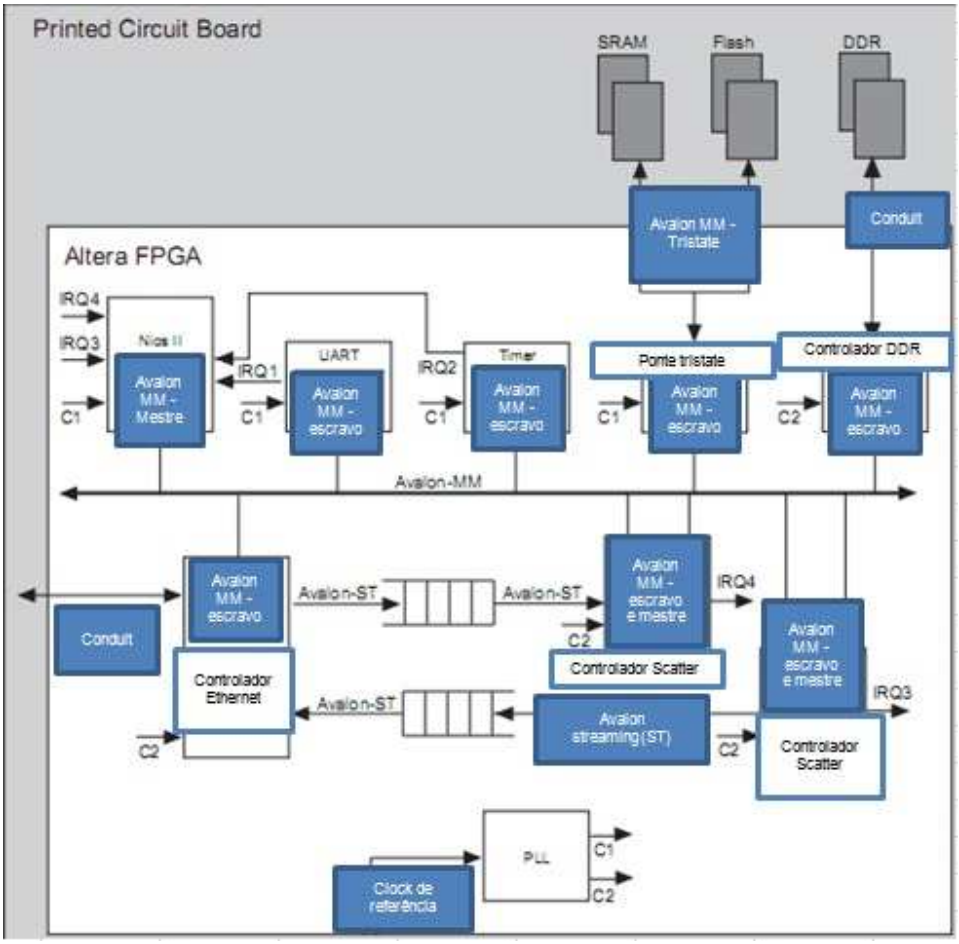


Figura 7 - Tipos de interfaces Avalon.

3.2.2 NIOS II

O NIOS II é um exemplo de processador que pode ser embarcado em FPGA. Este processador possui uma interface para utilização de memórias externas e sua programação pode ser desenvolvida através das linguagens C/C++ [25].

O processador NIOS II é fornecido pela *Altera Corporation*, focado como uma plataforma para processadores embarcados. Além de uma ferramenta de desenvolvimento baseada no projeto Eclipse e um compilador C/C++ específico, podem ser usados os programas SOPC Builder e Quartus II para o complemento do projeto [26].

Um sistema com processador NIOS II sendo um processador RISC (*Reduced Instruction Set Computer*) que possui um conjunto de instruções, caminhos de dados e espaços

de endereçamentos, equivalente a um microcontrolador ou "computador em um chip" que inclui um processador e uma combinação de periféricos e memória em um único chip [27].

O processador NIOS II consiste de um núcleo, um conjunto de periféricos *on-chip*, memória *on-chip*, e interfaces de memória *off-chip*, todas implementadas em um único dispositivo da Altera. Como um microcontrolador da família Altera, todos os sistemas com processador NIOS utilizam um conjunto consistente de instruções e um modelo de programação [28].

Um exemplo de sistema de hardware utilizando processador NIOS II pode ser visualizado na Figura 8 [27].

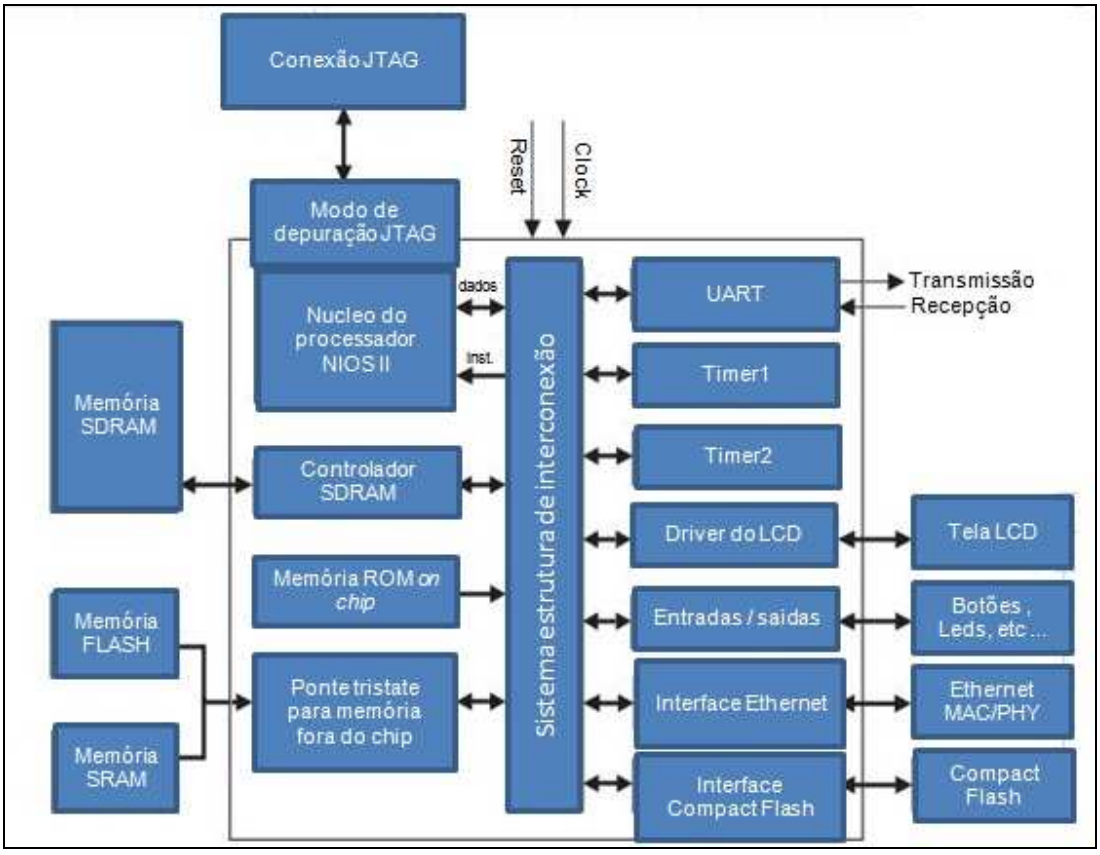


Figura 8 - Sistema de hardware utilizando processador NIOS II.

A grande vantagem do NIOS II é o fato de ser um processador com núcleo controlado por software e configurável o que permite grande flexibilidade [24]. A Altera disponibiliza alguns modelos de processador já pré-definidos como [26]:

1. NIOS II/e: versão “econômica”, modelado para projetos em que o tamanho do núcleo deve ser o mínimo possível e a velocidade de execução de instruções não é um aspecto crítico.
2. NIOS II/s: versão “padrão”, que exige menos recursos e um dispositivo FPGA como um *trade–off* para o desempenho reduzido, um modelo que equilibra o tamanho do núcleo do processador com a velocidade de execução, ou seja uma média entre o NIOS II/f e o NIOS II/e.
3. NIOS II/f: versão “rápida” concebida para um desempenho superior, tem um maior escopo de opções de configurações que pode ser usado para otimizar o processador; modelo cujo principal característica é acelerar a execução de instruções independentes do aumento que pode gerar no tamanho do núcleo do processador.

A execução de programas no NIOS II pode ser realizada com o processador em três (3) modos diferentes [27]:

- Supervisor: permite que o processador execute todas as instruções e permite executar todas as funções. Quando o processador é reiniciado, entra neste modo.
- Usuário: é o modo em que a maioria dos programas deve executar, porém, nem todas as funcionalidades do processador estão disponíveis e a tentativa de acessá-las pode ocasionar um pedido de exceção.
- Depuração: é o modo utilizado para fazer análises do condicionamento do NIOS II e é acessado por ferramenta externas ao processador. Neste modo, todas as funcionalidades do processador são acessíveis, assim como todos os registradores.

3.2.2.1 Ambiente de Desenvolvimento para NIOS II (NIOS II EDS)

O NIOS II EDS é um ambiente de desenvolvimento de software para o processador NIOS II. Para que haja comunicação entre o desenvolvimento de programas com o processador NIOS II são necessários um computador, uma placa FPGA e um cabo JTAG, para download [27]. Na Figura 9, pode-se observar o ambiente de desenvolvimento da programação do processador NIOS II, sendo que:

1. Menu para depurar.
2. Ambiente para compilar e enviar para a placa.
3. Ambiente para programação.
4. Biblioteca do *template*.
5. Descrição do sistema.
6. Novo projeto, biblioteca de templates.

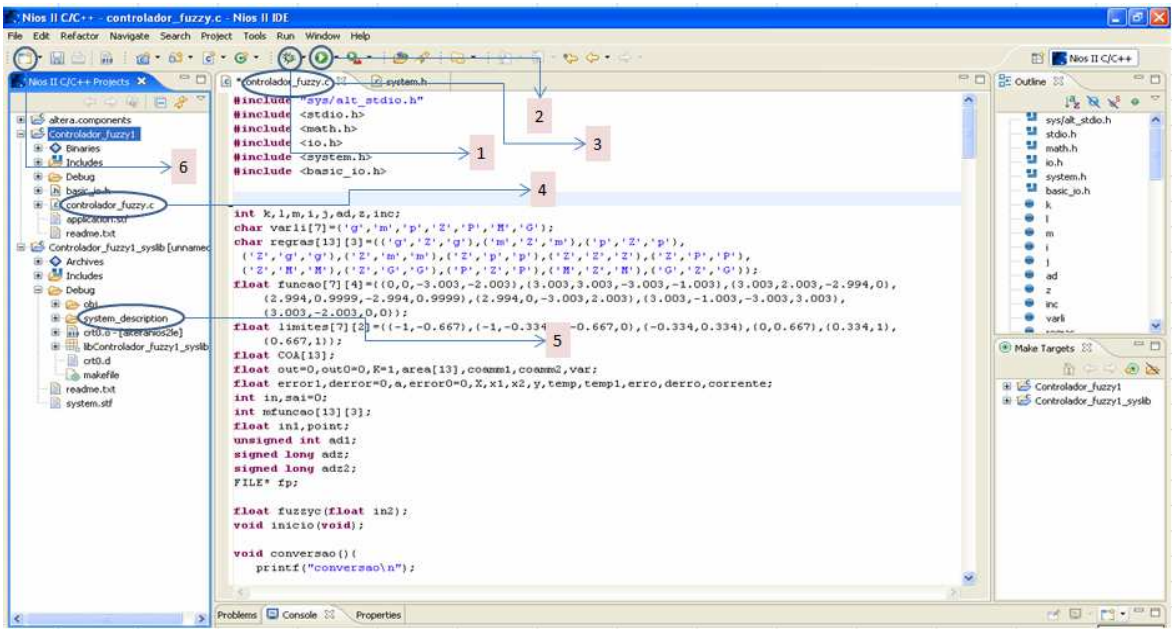


Figura 9 - Ambiente de desenvolvimento NIOS II IDE.

3.3 KIT ALTERA

O desenvolvimento e implementação deste projeto utilizou a placa de desenvolvimento DE2 da Altera, ilustrada na Figura 10. Esta placa contém um chip Altera Cyclone II e um FPGA com 33.216 elementos lógicos [29].

Figura 10 - Kit Altera Cyclone II C35.

3.4 Condicionamento de Sinais

A produção de grande variedade de bens materiais depende dos sistemas de automação industrial, os quais fazem uso de quaisquer dispositivos mecânicos ou eletro-eletrônicos para controlar máquinas e processos. Durante as etapas dos processos industriais, as variáveis podem ser medidas através de sensores. Os sensores desempenham uma função muito importante na constituição de um sistema de automação industrial como pode ser visto na Figura 11, pois é através deles que se consegue aferir uma grande variedade de grandezas, tais como temperatura, vazão, pressão, corrente elétrica e velocidade de rotação de um motor [30].

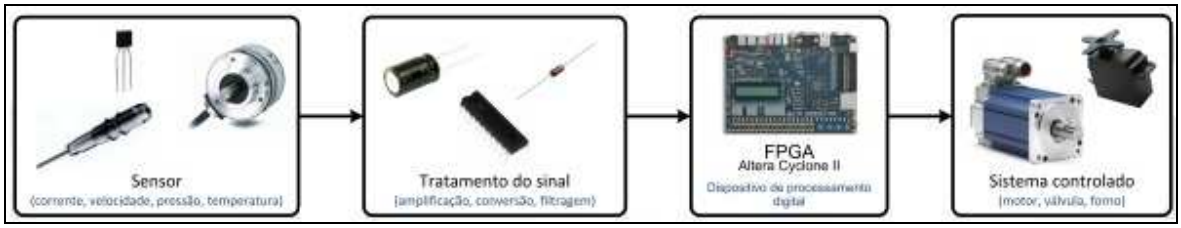


Figura 11 - Processo de condicionamento de sinais proposto.

Na maioria das vezes, o sinal de saída de um sensor industrial não é adequado para a leitura por um dispositivo de processamento digital. Em geral, os sensores fornecem sinal de saída de natureza analógica, porém o dispositivo de processamento em questão (FPGA) exige entrada de dados de natureza digital. Assim, o tratamento dos sinais é necessário para converter tais sinais de maneira que os dados fornecidos pelo sensor possam ser utilizados.

3.4.1 Sensor de Corrente de Efeito Hall

No campo da automação industrial, muitas vezes é preciso monitorar a corrente que alimenta um equipamento. Para tal medição, os sensores de corrente de efeito *Hall* são bastante utilizados devido à sua facilidade de uso e sua relação linear da tensão de saída com a corrente medida. O funcionamento deste sensor está relacionado ao surgimento de uma diferença de potencial em um condutor elétrico, transversal ao fluxo de corrente e um campo magnético perpendicular a corrente [31].

Um exemplo de aplicação de sensor de efeito *Hall* na automação industrial é o monitoramento da corrente de alimentação de uma centrífuga em uma usina de produção de álcool e açúcar, necessário para que seja possível fazer o controle do ângulo de abertura da válvula que controla o fluxo de entrada do material a ser centrifugado. Neste trabalho foi desenvolvido um circuito contendo a parte de condicionamento de sinais para o sensor de efeito Hall e tratamento de sinais para o servo-motor.

CAPÍTULO 4 - IMPLEMENTAÇÃO DO PROTÓTIPO

4.1 Introdução

Desenvolveu-se neste projeto um módulo embarcado que recebe os sinais provenientes do motor da centrífuga, o qual é a variável de controle. Em se tratando de um controlador *fuzzy* PD (Proporcional Derivativo) trabalhou-se com o erro, a derivada do erro e a saída.

O desenvolvimento de um sistema embarcado como o "controlador *fuzzy*" consiste de duas partes fundamentais: a parte hardware e a parte software, contando com o auxílio de uma etapa preliminar descrita como condicionamento de sinais.

A proposta deste trabalho é aplicação de um controlador fuzzy para controlar as atividades de uma centrífuga contínua. Para simular o funcionamento da centrífuga, utilizou-se um motor de indução trifásico 220 V e 1/3CV, variando a carga em seu eixo.

O controlador fuzzy deve atuar na abertura da válvula, ou seja, no ângulo de posicionamento do servo-motor, sendo que neste caso, a grandeza a ser controlada é a corrente do motor, a qual se deve manter na faixa estabelecida pelo operador; havendo variância, o ângulo de abertura do servo-motor deve atuar.

A lógica do controlador fuzzy foi inserida no processador NIOS II e está embarcado no FPGA (placa DE2). O experimento proposto funciona da seguinte maneira:

1º O operador deve estabelecer uma referência, ou seja, uma corrente que a máquina trabalhe normalmente. No processo de centrifugação contínua é utilizada a I_n (corrente nominal do motor).

2º O operador deve estabelecer a corrente máxima para que não haja parada no processo de centrifugação contínua. Como o motor é acionado por uma *soft-starter*, sua proteção está em 2,4 x I_n, ou seja, se a corrente exceder esse limite a *soft-starter* desligará o motor, ocasionando parada no processo.

3º É feita a leitura da corrente e, através de sua variação, o servo motor atuará modulando a válvula (variação de ângulos), ou seja, aumentando ou diminuindo a vazão de

entrada, no processo de centrifugação contínua será realizado o controle da entrada de magma para produção de açúcar.

A arquitetura implementada neste trabalho pode ser observada na Figura 12.

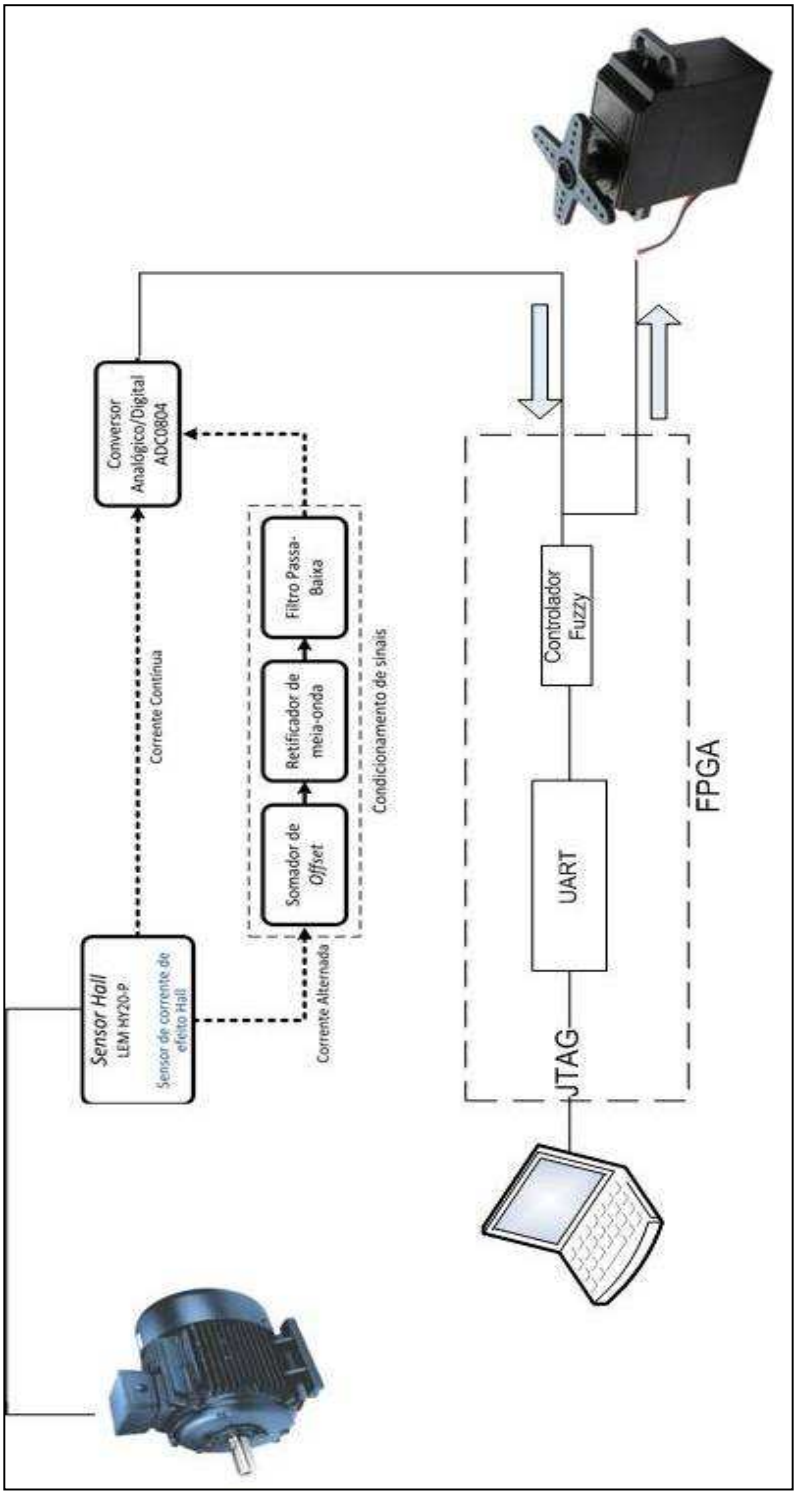


Figura 12 - Esquemático do projeto proposto.

No item 4.2 descreve-se sobre os circuitos de condicionamento de sinais, no item 4.3 a etapa de desenvolvimento do hardware e no item 4.4 a etapa de geração de firmware.

4.2 Condicionamento de Sinais

O sensor de corrente utilizado foi o LEM HY-20-P. O tratamento do sinal foi feito considerando a corrente medida no intervalo de aproximadamente 0 a 12 A_{rms}, e na saída utilizou-se um sinal digital de oito (8) bits para entrada no FPGA.

O circuito de condicionamento de sinais conta com o sensor de corrente, um somador de *offset* de tensão, um retificador de meia-onda e um filtro passa-baixa como pode ser visto na Figura 13.

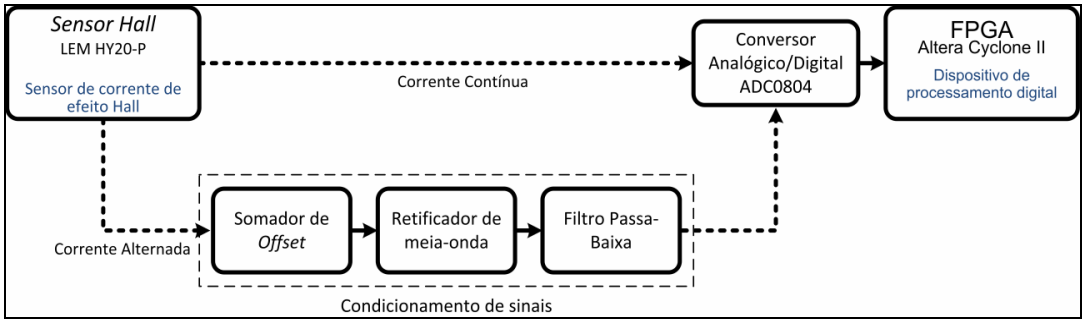


Figura 13 - Condicionamento de sinais CA/CC.

O somador de tensão de offset foi montado utilizando um amplificador operacional LM741 [32]. A utilização de um somador de *offset* se dá devido à queda de tensão no diodo retificador; caso o somador não fosse utilizado, o sinal seria perdido para pequenas tensões na saída do sensor. A tensão de *offset* foi gerada com um divisor resistivo e é acompanhada de um *buffer*, que tem a função de garantir que o divisor resistivo não seja influenciado por resistências de carga. Os valores das resistências do divisor resistivo foram escolhidos a fim de que se compense a queda de tensão de diodo.

Após a etapa do somador de *offset*, o sinal é retificado por um retificador de meia-onda e então passa por um filtro passa-baixa. O filtro foi dimensionado de maneira que seja suficiente para linearizar a meia-onda e não deixar a onda cair lentamente com a diminuição na corrente de entrada do sensor.

Após o estágio de condicionamento de sinais, vem o estágio de conversão analógica/digital. O conversor ADC0804 foi escolhido devido ao seu bom desempenho e sua disponibilidade no mercado.

A montagem do circuito foi feita de acordo com as informações contidas no manual do conversor [33]. As equações [4.1] e [4.2] descrevem o circuito montado.

$$V_{analógico} = (I_{rms} * \sqrt{2}) * 0{,}2088 + 0{,}415 \quad (4.1)$$

$$V_{digital} = V_{analógico} * \frac{255}{5} \quad (4.2)$$

sendo que:

0,2088 é uma escala do sensor de efeito Hall, determinada através de ensaios, em que mensurado que para 1 A equivale 0,2088 V.

0,415 é o valor para compensar a perda de tensão do diodo.

O esquema de ligação e componentes da placa de condicionamento de sinais utilizado pode ser visualizado na Figura 14.

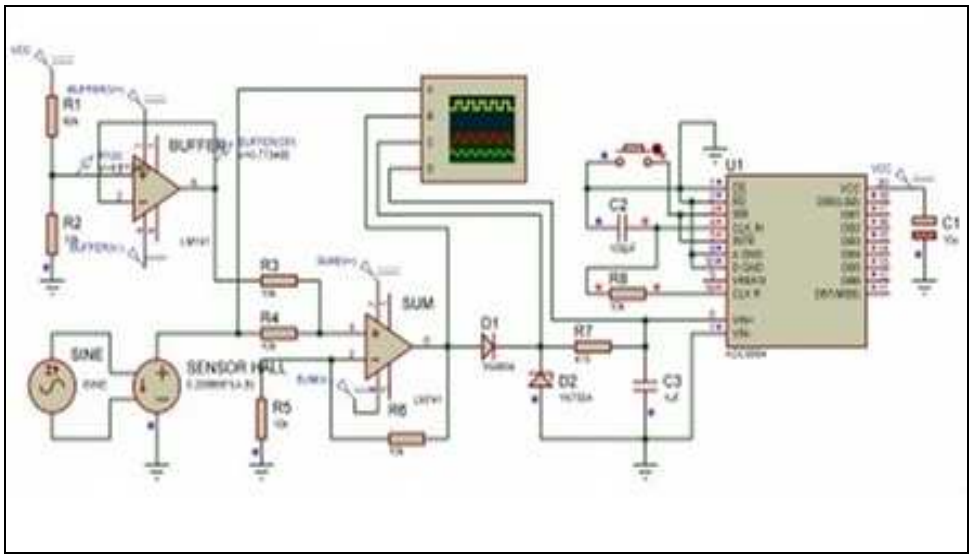


Figura 14 - Esquema de ligação do condicionamento de sinais.

Como se pode observar na Figura 15, uma placa de circuito eletrônico foi confeccionada para conectar os transdutores (sensor de efeito Hall, servo-motor) e um barramento para a placa de FGPA.

Esta placa tem como finalidade regular o sinal proveniente do sensor de efeito Hall para depois convertê-lo para digital.

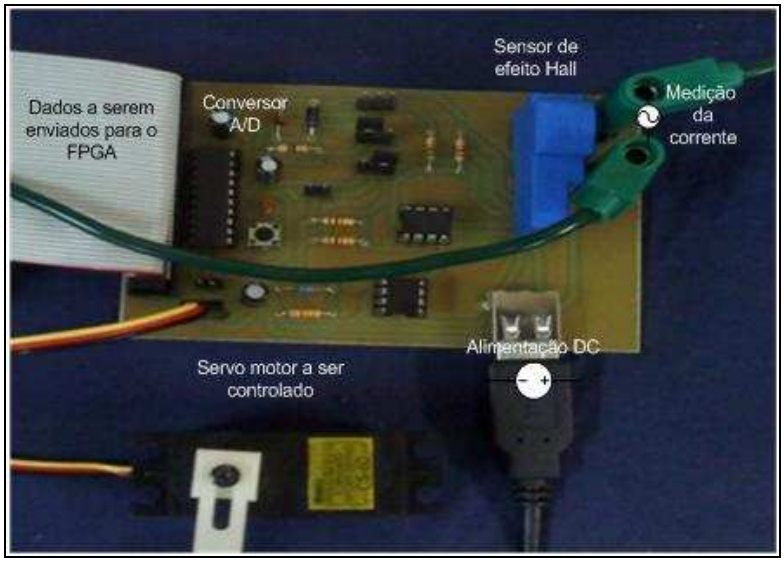


Figura 15 - Circuito eletrônico para viabilizar o projeto de controle.

A validação do circuito apresentado na Figura 15 se deu através da recepção de valores conhecidos e a saída, necessariamente, teria que respeitar as equações (4.1) e (4.2).

Na Figura 16 é representada esta validação: a tensão de aquisição no A/D é de 0,68V, o que resulta na saída digital $00100011_{(2)}$, ou seja, $35_{(10)}$, representando uma corrente de linha no motor de $0{,}9A_{rms}$.

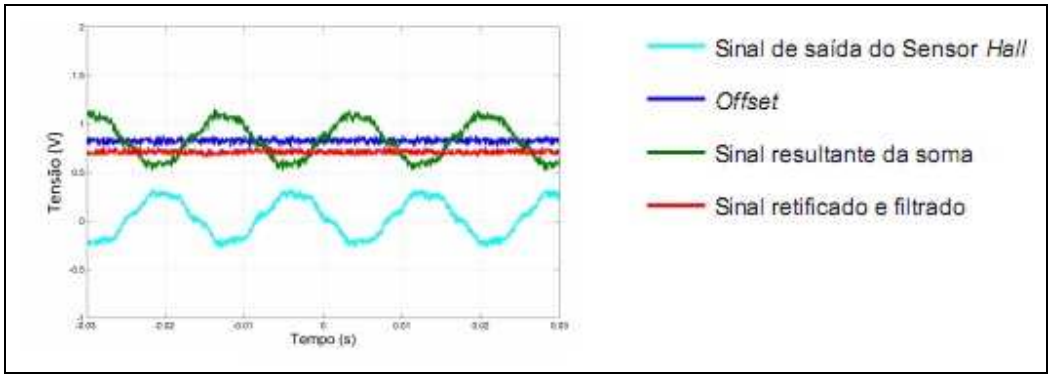


Figura 16 - Dados adquiridos do osciloscópio digital.

4.3 Desenvolvimento de Hardware

A parte hardware do "controlador *fuzzy*" é composta de um *Driver* chamado controlador_fuzzy, do PLL (*Phase-Locked Loop*) e das pinagens necessárias para endereçar as *I/O* deste hardware. O *Driver* controlador_fuzzy foi gerado pelo *SOPC Builder*, sendo a arquitetura básica do módulo embarcado, que é a base para o hardware de controle utilizado na automação da centrifugação contínua.

O hardware desenvolvido no *SOPC Builder* tem as seguintes configurações:

CPU: um processador NIOS II/s de 50 MHz.

SDRAM: uma memória SDRAM de 8 Mbytes com largura de dados de 16 bits.

JTAG_UART: uma porta de saída para *downloads* de programas ou hardwares projetados no FPGA via USB.

SYS_CLK_TIMER: um módulo de timer interno de 1 μs para ter uma referência de clock.

INPUT: uma porta de oito (8) bits de entrada para receber os dados provenientes do conversor A/D.

OUTPUT: uma porta de um (1) bit de saída para o envio do sinal ao servo-motor.

A Figura 17 representa o hardware configurado no SOPC Builder.

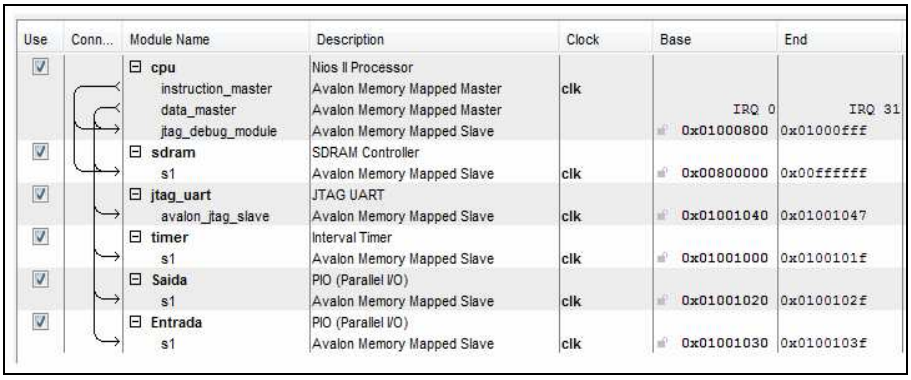


Figura 17 - Configuração do Hardware “controlador *fuzzy*”.

A memória *SDRAM* que compõe o “controlador fuzzy” possui capacidade de armazenamento de 8 *MBytes*, e foi configurada de modo a possuir doze (12) linhas por oito (8) colunas de endereçamento, dezesseis (16) bits de dados, quatro (4) bancos *chipselect*. Esta memória foi necessária, pois a aplicação no *NIOS II IDE* excede o tamanho das memórias *on-chip*. Há flexibilidade na configuração desta memória, porém, como foi utilizada a placa DE2, é esta a versão de memória disponível.

Foram realizados testes para mensurar a eficiência dos processadores e, para este processo em particular, o processador NIOS/s foi aquele que obteve uma melhor perfomance por permitir alocação das instruções memória *cache*, utilização de sistema de *pipeline* e execução de previsão de desvio *(Branch Prediction)*, características ausentes na versão NIOS/*e*. Já a versão NIOS/f foi preterida por consumir maior quantidade de elementos lógicos, quando comparado com a versão *NIOS II/s.*

O PLL (*Phase Lock Loop*) foi gerado através de uma ferramenta chamada Mega*Wizzard plug-in Manager* e sua saída ligada ao clock da *SDRAM*. Tal ferramenta pode ser visualizada na Figura 18.

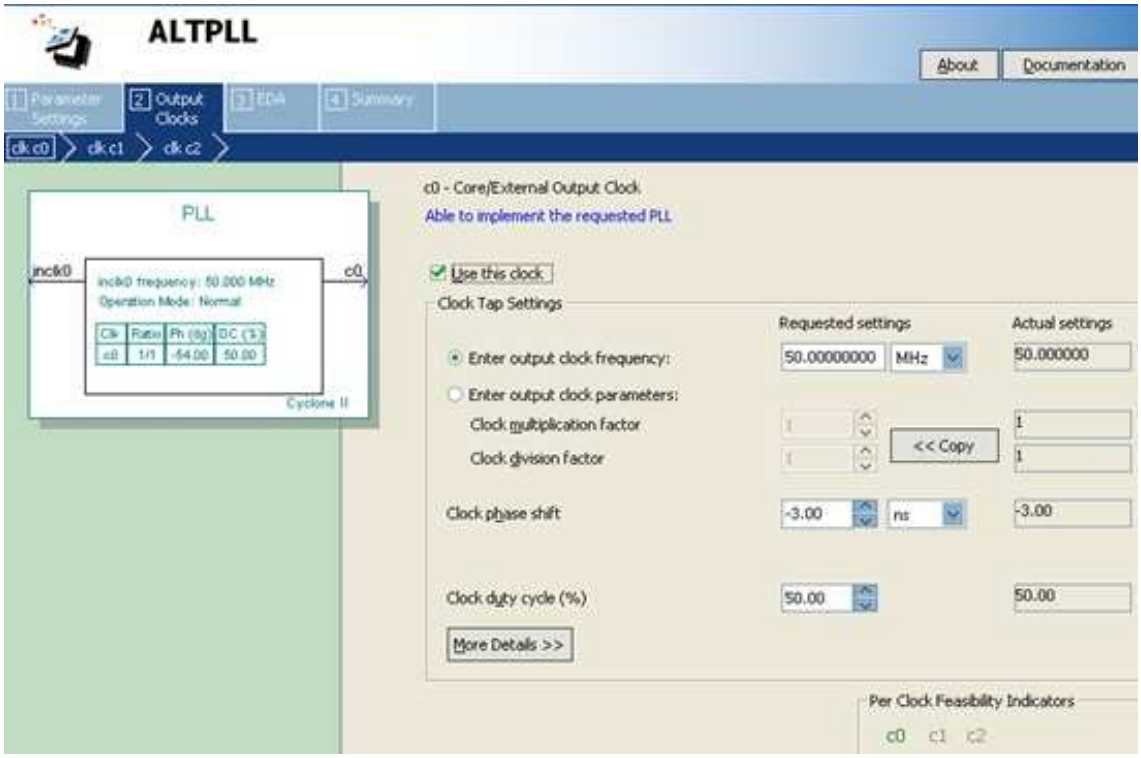


Figura 18 - Configurando o PLL.

O PLL é utilizado no sincronismo do *clock* da SDRAM com o clock do SDRAM *Controller*, de modo que os dados, endereços e sinais de controle cheguem estabilizados nos pinos da SDRAM. Para gerá-los utilizou-se os recursos da *MegaWizard Plug-in Manager* pertencentes à biblioteca de símbolos do *Quartus II*.

O Driver “controlador_fuzzy” e o PLL são vistos no software QUARTUS II através da Figura 19.

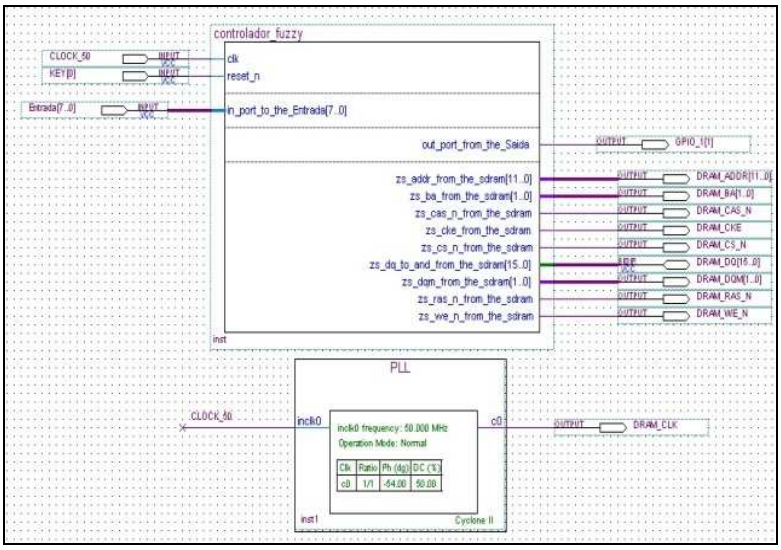


Figura 19 - Hardware gerado pelo SOPC *Builder*.

Este hardware possui uma entrada que é configurada para receber oito (8) *bits* referentes ao sinal digital. Como o sinal de entrada é uma tensão gerada pelo sensor de efeito Hall, deve-se transformá-lo para digital utilizando um conversor A/D, conforme descrito no item 4.2.

Para que o supervisório possa ler a corrente eficaz do motor criou-se uma fórmula respeitando a equação 4.1 que pode ser visualizada na segunda linha dentro do *While* da Figura 20.

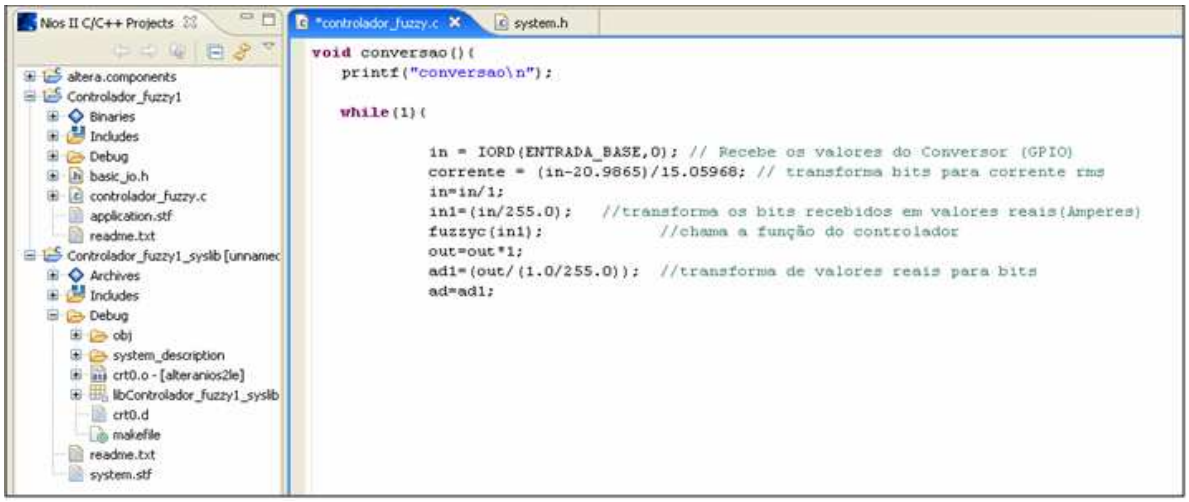


Figura 20 - Recebimento e envio e transformação de sinais.

4.4 Geração do Firmware

A implementação do controlador Fuzzy no FPGA se deu através da programação no processador NIOS II e dos seguintes elementos: entrada, saturadores, derivador, fuzzificador, controlador lógico fuzzy (funções de pertinência, regras de inferência e defuzzificador). Todos estes elementos podem ser observados no capítulo 5, na Figura 28.

No processador NIOS II inseriu-se a lógica do controlador fuzzy com sete (7) funções de pertinências triangulares e eqüidistantes, e utilizou-se o método de defuzzificação COA (*Centre-of-Area*) [6], [9].

Foi desenvolvida uma biblioteca já com as funções do controlador fuzzy chamada *controlador_fuzzy.c*, de tal maneira que quando acionada precise mudar apenas algumas variáveis ou funções para outro tipo de controle. O *template* pode ser visualizado na Figura 21.

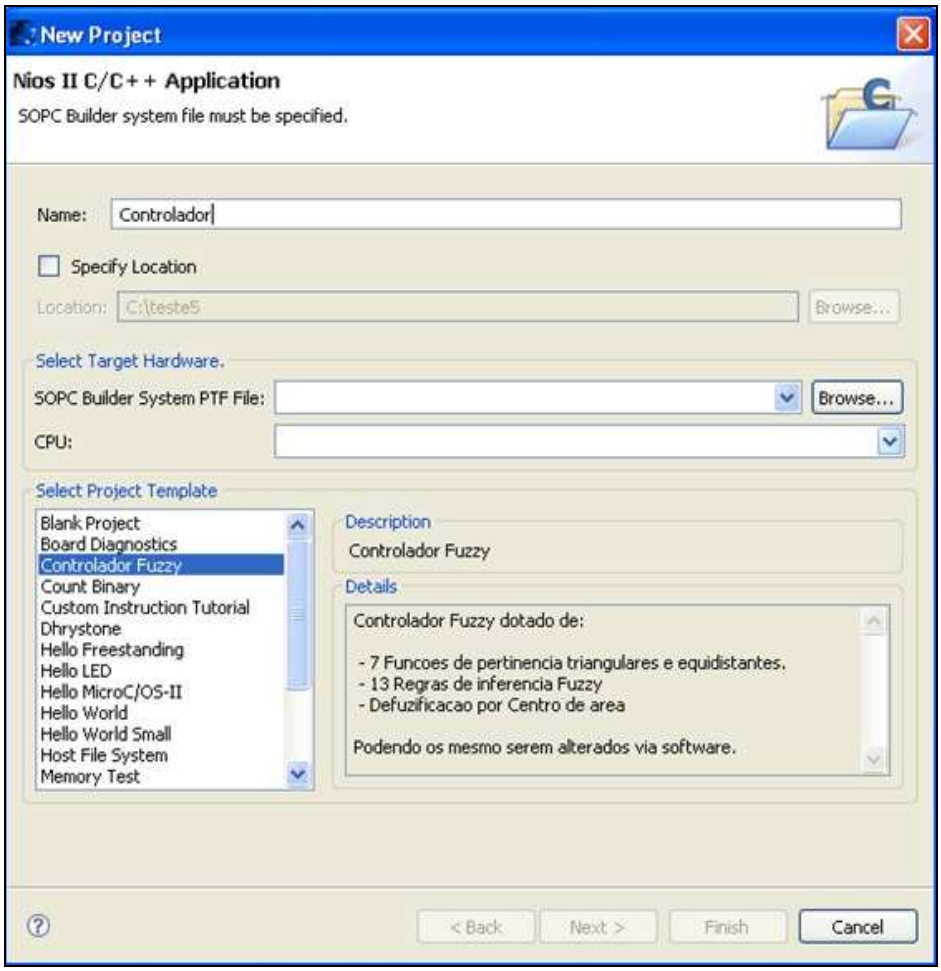


Figura 21 - Biblioteca do controlador fuzzy já inserido nos *templates* do NIOS II IDE.

Pode-se observar na Figura 22 os limites das funções de pertinência e os coeficientes da equação das retas que compõem cada função desta biblioteca.

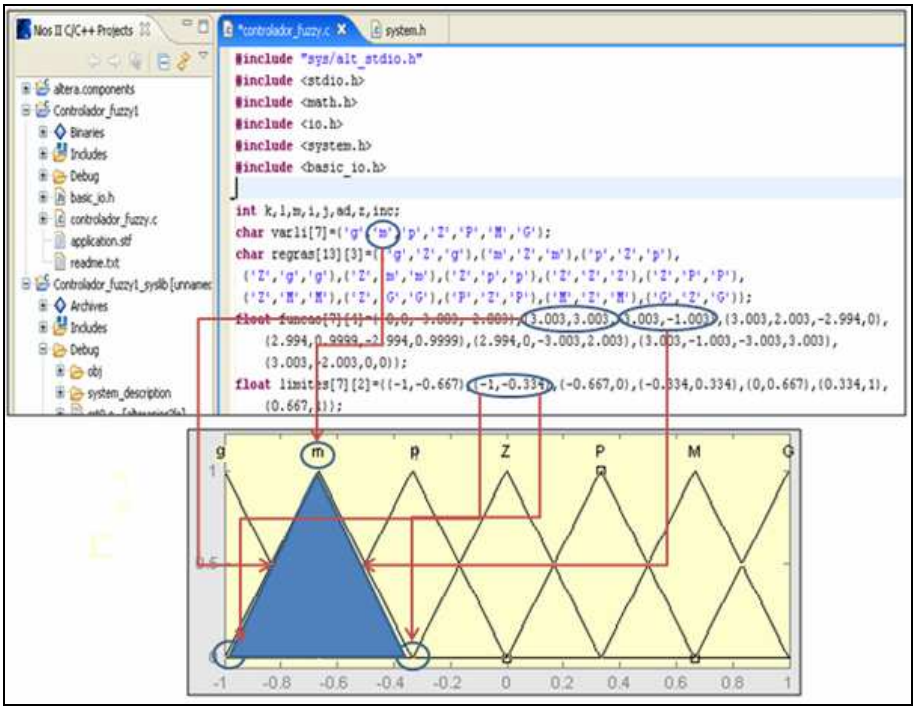


Figura 22 - Matrizes que contêm os limites das funções de pertinência e os coeficientes da equação das retas que compõem cada função.

O controlador é dotado de treze (13) regras, as quais podem ser alteradas e/ou adicionadas novas regras por qualquer futuro projetista da maneira que achar mais conveniente. As regras utilizadas para este projeto podem ser observadas na Tabela 1 e seu comportamento exposto na Figura 23.

Tabela 1: Regras fuzzy configuradas para o controlador.

Erro	Derivada do Erro						
	g	m	p	Z	P	M	G
g				g			
m				m			
p				p			
Z	g	m	p	Z	P	M	G
P				P			
M				M			
G				G			

sendo que:

g = negativo grande, m = negativo médio , p = negativo pequeno , Z = zero , P = positivo pequeno , M = positivo médio , G = positivo grande.

As regras podem ser descritas da seguinte maneira, por exemplo:

Regra 1: SE erro = Z e Derivada do Erro = g ENTÃO saída = g.

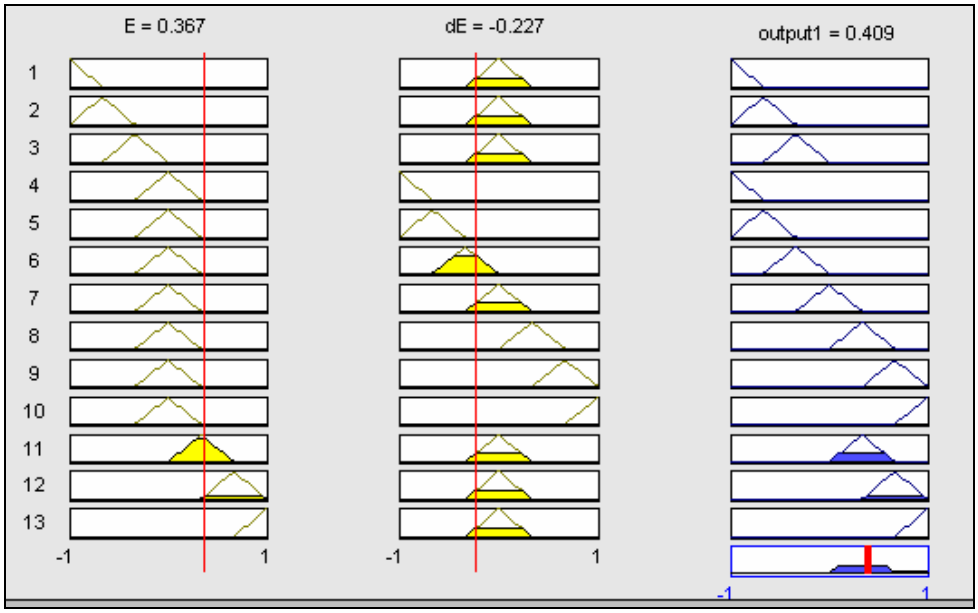


Figura 23 - Comportamento das regras.

Na Figura 24 é exposto parte do algoritmo que realiza a inferência e na Figura 25 encontram-se parte da função que realiza o cálculo da área da função de pertinência, para, posteriormente, fazer defuzzificação.

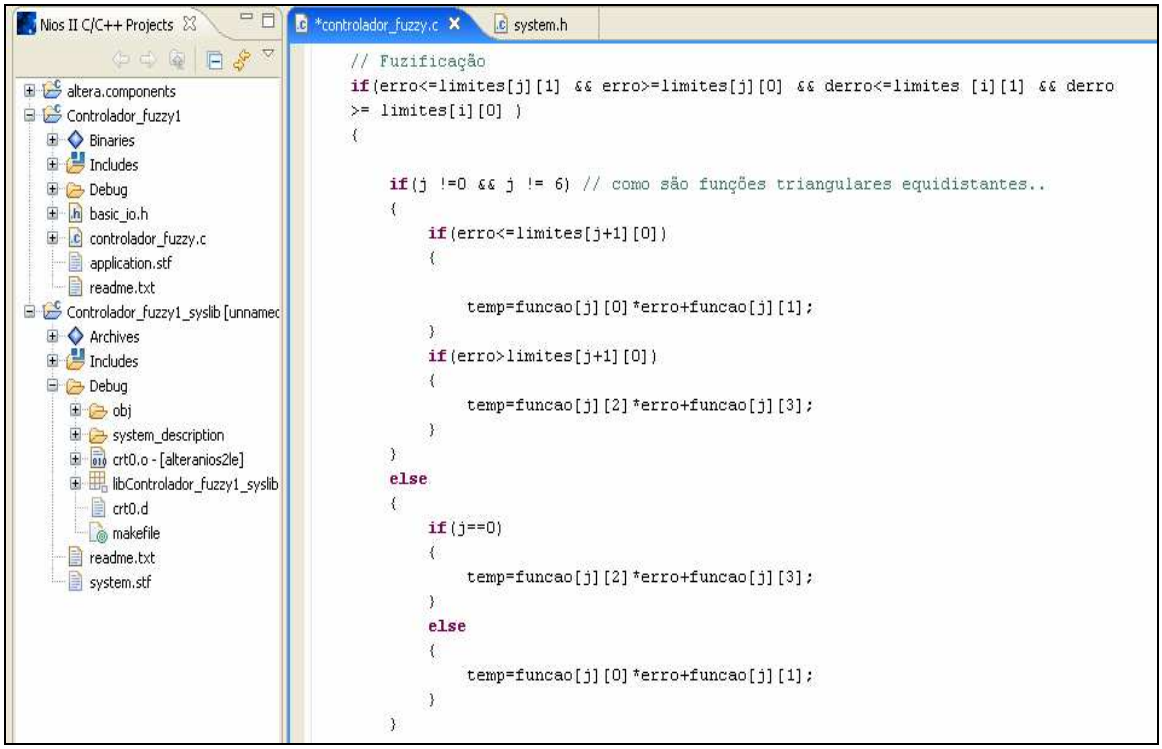


Figura 24 - Lógica de fuzzificação programada no processador NIOS II.

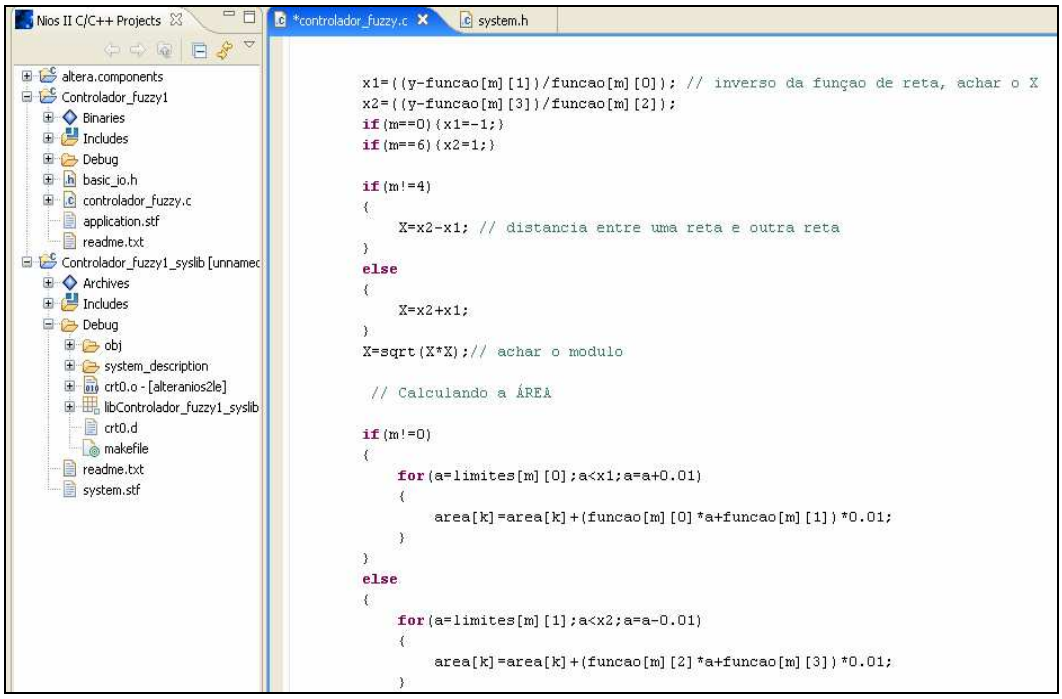


Figura 25 - Cálculo da área.

O processo de defuzzificação utilizado foi o do centro da área (COA), que pode ser observado no código da Figura 26.

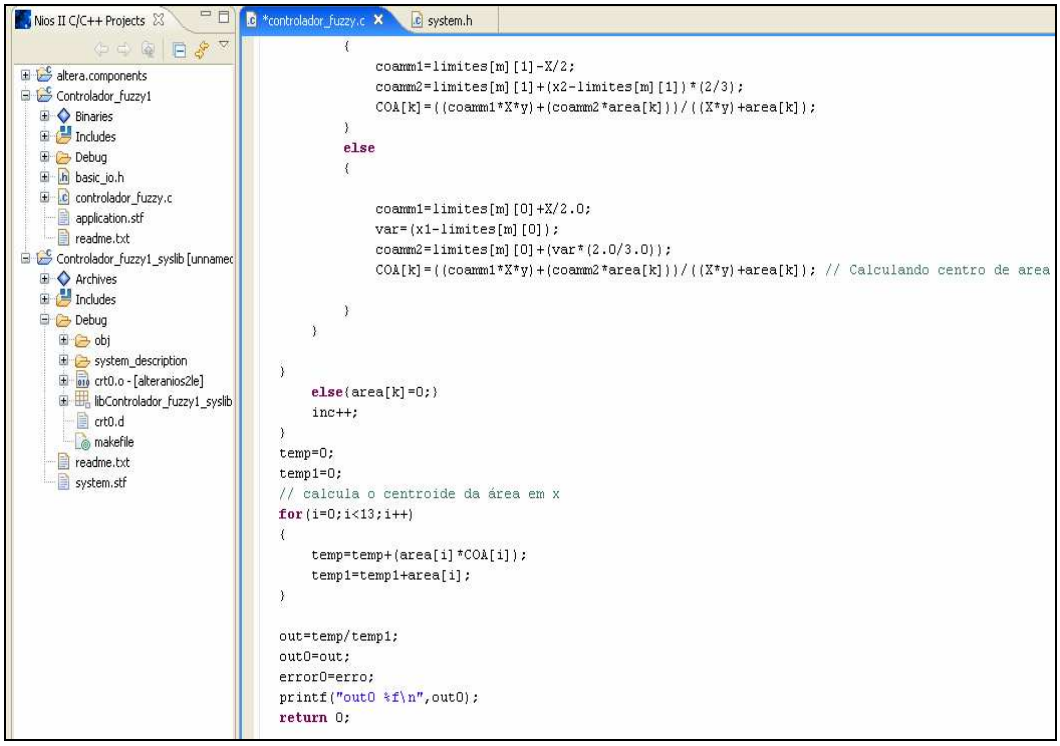


Figura 26 - Defuzzificação por COA (centro de área).

Na saída do controlador fuzzy foi gerado o PWM (*Pulse Width Modulation*) correspondente a um ângulo de abertura/fechamento, sendo capaz de atuar na vazão da matéria desejada. Por exemplo, para um ângulo de -90º teve-se que gerar um pulso positivo de 2,3 ms em um período de 20 ms. O código para gerar este pulso é apresentado na Figura 27.

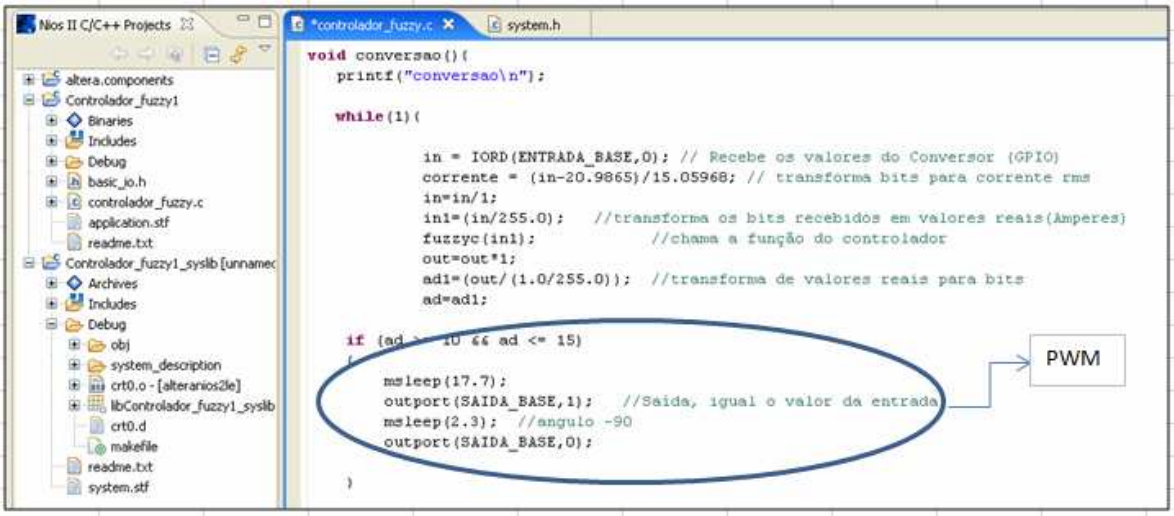


Figura 27 - PWM que resultará no ângulo -90º.

CAPÍTULO 5 – SIMULAÇÃO

5.1 Introdução

Para fazer a simulação deste processo foi utilizado o Simulink do software MATLAB. Nesta, foi modelado um motor similar ao testado na prática, porém, com ajustes nos ganhos. O diagrama completo pode ser visualizado na Figura 28. Para “emular” uma carga semelhante à de uma centrífuga contínua utilizou-se uma seqüência interpolada com repetições. Na Figura 29 é possível notar a mudança exercida pela carga “emulada” no torque eletromagnético e na Figura 31 é visto a corrente de uma fase do estator e seu valor RMS.

O objetivo desta simulação é de produzir uma carga com características semelhantes a do motor testado na prática e comparar os valores obtidos.

Foi necessária, na saída do controlador fuzzy, a criação de um bloco intitulado “conversão”, para facilitar o entendimento do leitor. Esse bloco tem como finalidade converter o valor fuzzy para o valor real, referente à faixa de corrente entre 4 mA – 20 mA.

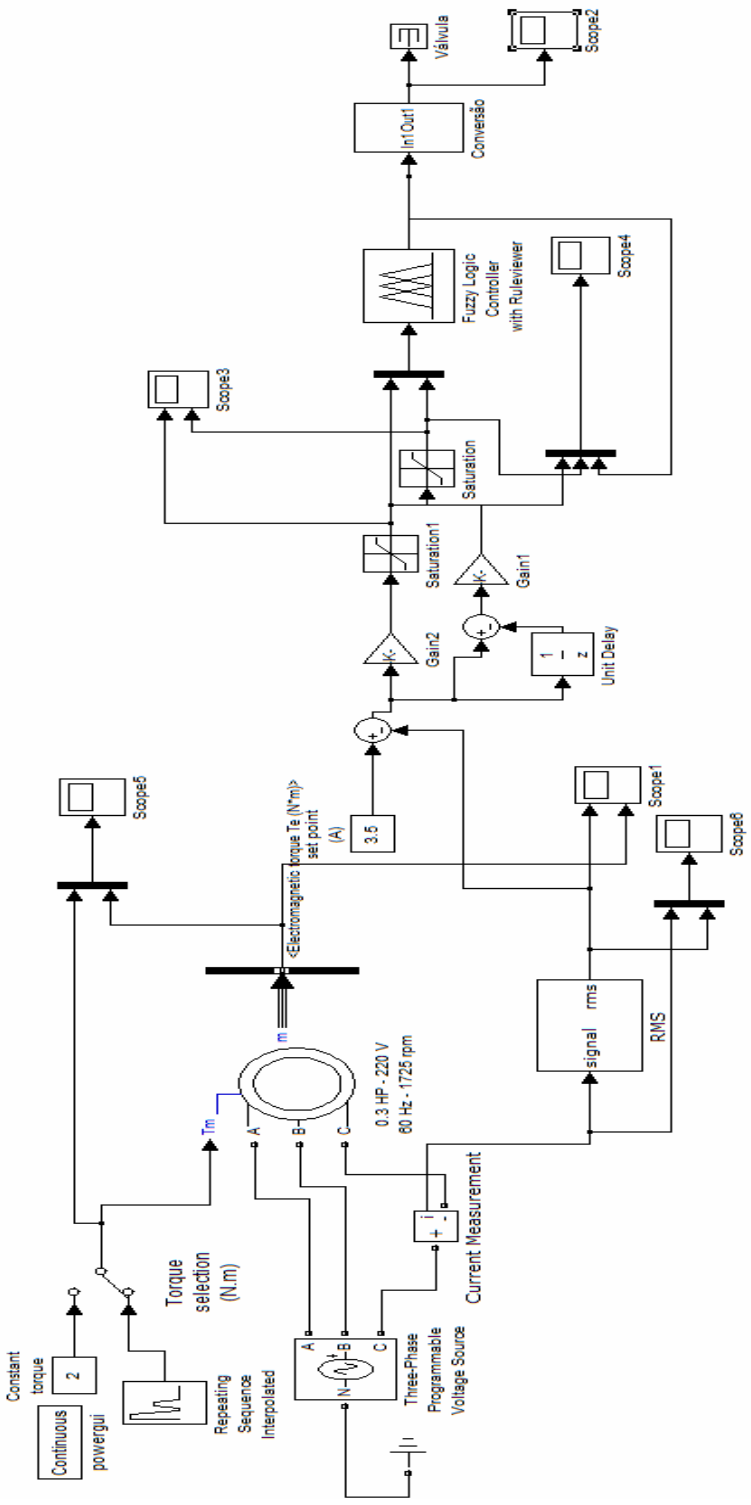


Figura 28 - Sistema proposto para automação.

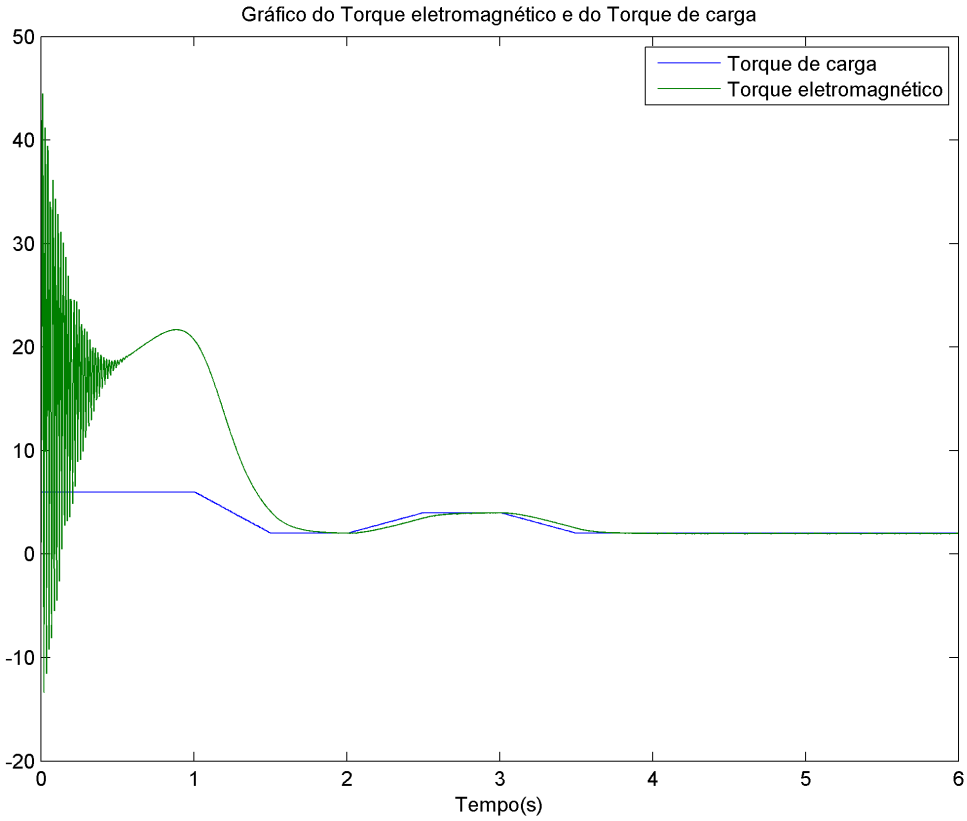


Figura 29 Torque de carga e Torque Eletromagnético

Para que fosse possível utilizar a corrente RMS foi necessária a utilização de um bloco conversor de sinal para valor RMS. O gráfico da corrente da fase C em conjunto com seu sinal RMS pode ser observado na Figura 30.

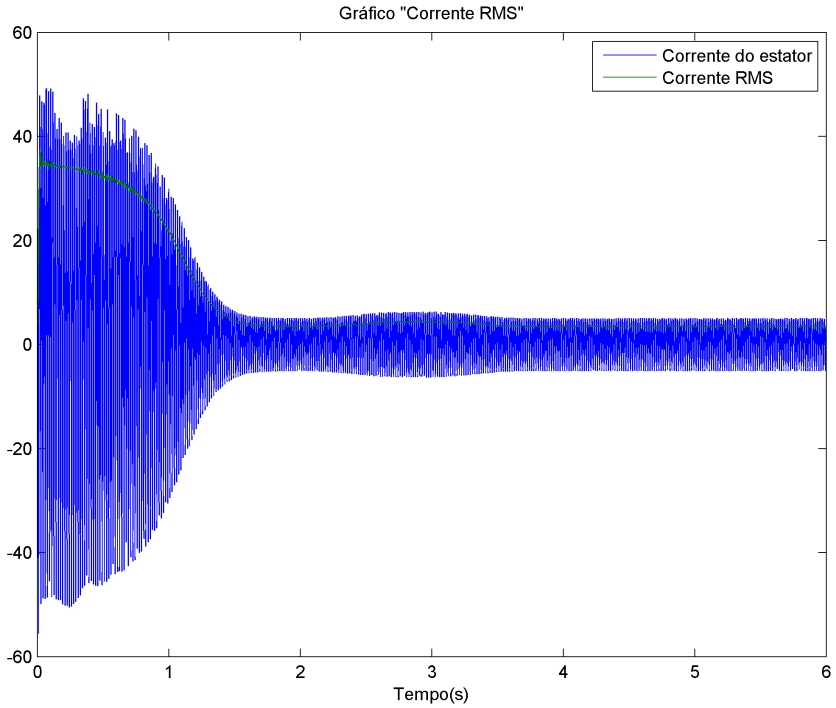


Figura 30 Corrente do estator e seu valor RMS .

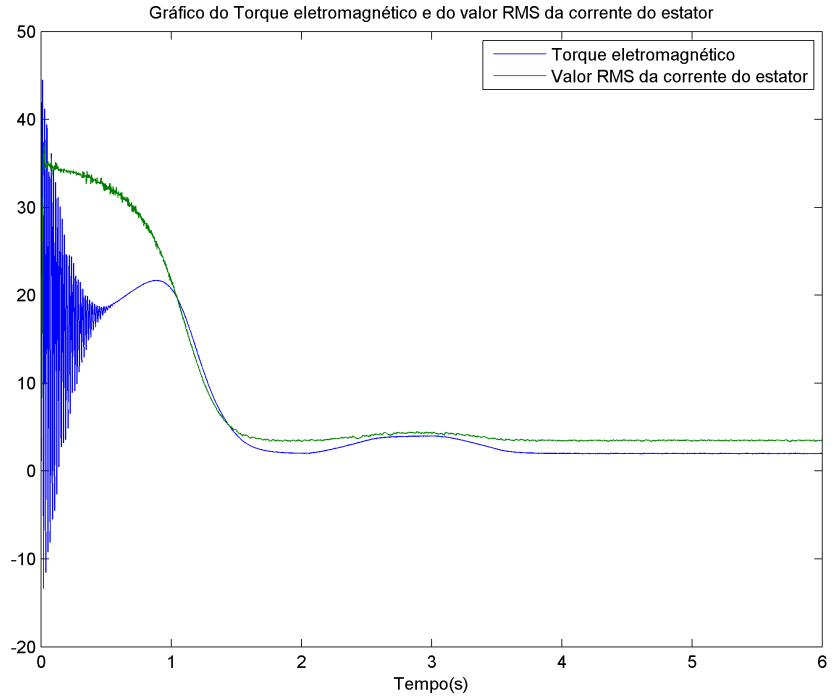


Figura 31 - Corrente RMS do motor e seu Torque Eletromagnético.

Com essa variação de corrente foi possível visualizar o comportamento do controlador *fuzzy*. Na Figura 32 estão destacados a saída do controlador, a variação do erro e o erro. Pode-se observar que com a corrente de partida o controlador fuzzy enviou sinal para o fechamento total da válvula (entre 0.5 e 1), pois neste estágio houve uma corrente de pico maior que o $I_{máx}$ definida pelo usuário (no caso da centrifugação contínua, $2.4*I_{nom}$). Durante o processo, a modulação ocorreu nos cinco (5) ângulos. Na Figura 33 é observada a saída do controlador já convertida para escala de corrente. Como nas indústrias as válvulas são atuadas por valores de escala 4 mA – 20 mA, esse foi o intervalo realizado, em que:

- 20 mA – totalmente fechada (90º)
- 16 mA – (45º)
- 12 mA – (0º)
- 8 mA – (- 45º)
- 4 mA – totalmente aberta (- 90º)

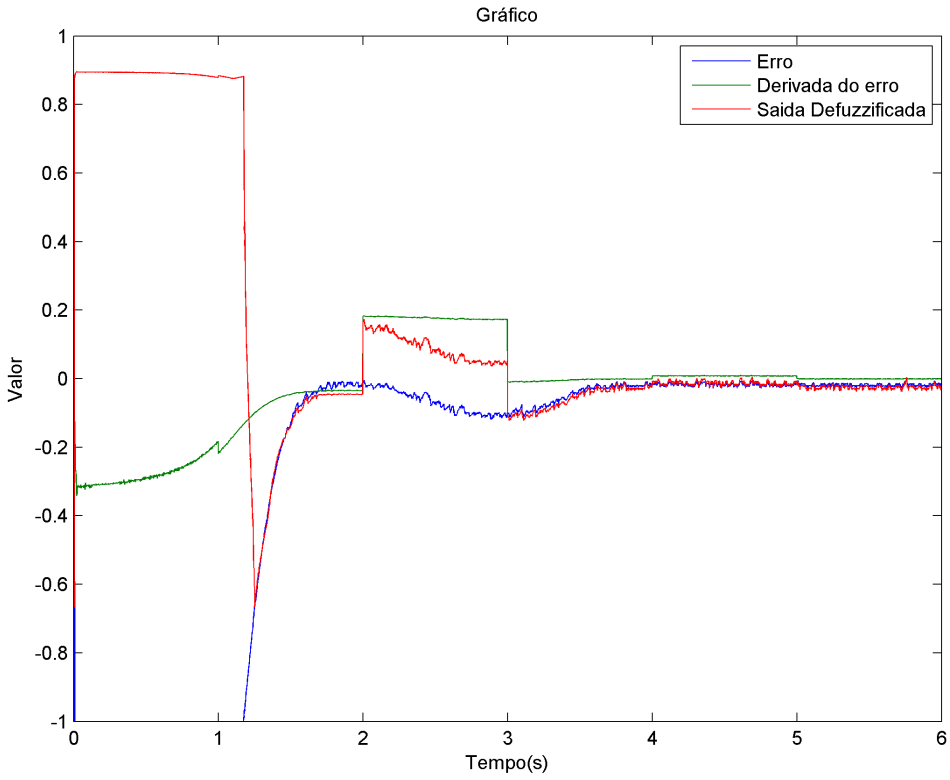


Figura 32 - Resposta do controlador *fuzzy* composta do erro, derivada do erro e saída *fuzzy*.

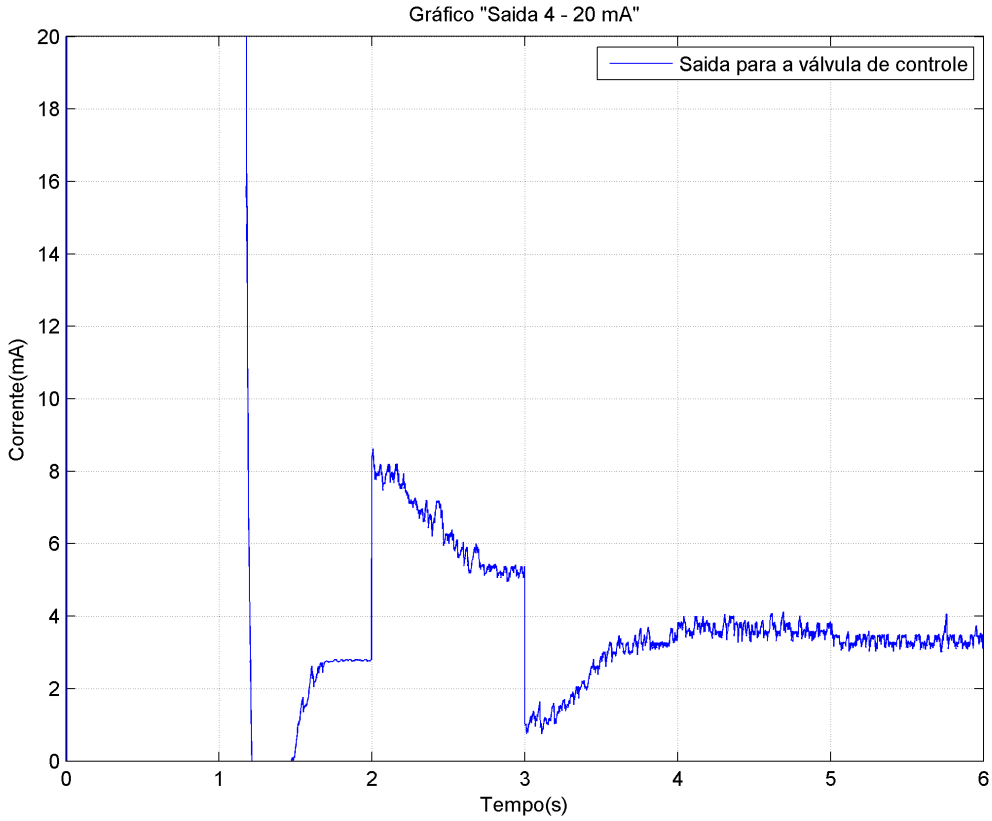


Figura 33 Saída do controlador fuzzy convertida para 4 mA – 20 mA.

CAPÍTULO 6 - RESULTADOS

Para a realização de testes do controlador fuzzy foi montado um protótipo feito a partir de um motor de indução trifásico 220 V, similar ao utilizados em processos industriais, porém, com uma potência menor.

O protótipo foi montando em um kit de instalações elétricas industriais, cedido pela Universidade Católica Dom Bosco - UCDB, em que foi feita a ligação de diversos motores de corrente alternada e corrente contínua (CA/CC), com suas devidas proteções, a fim de se testar o controlador. O kit e sua interação com o FPGA podem ser visualizados na Figura 34.

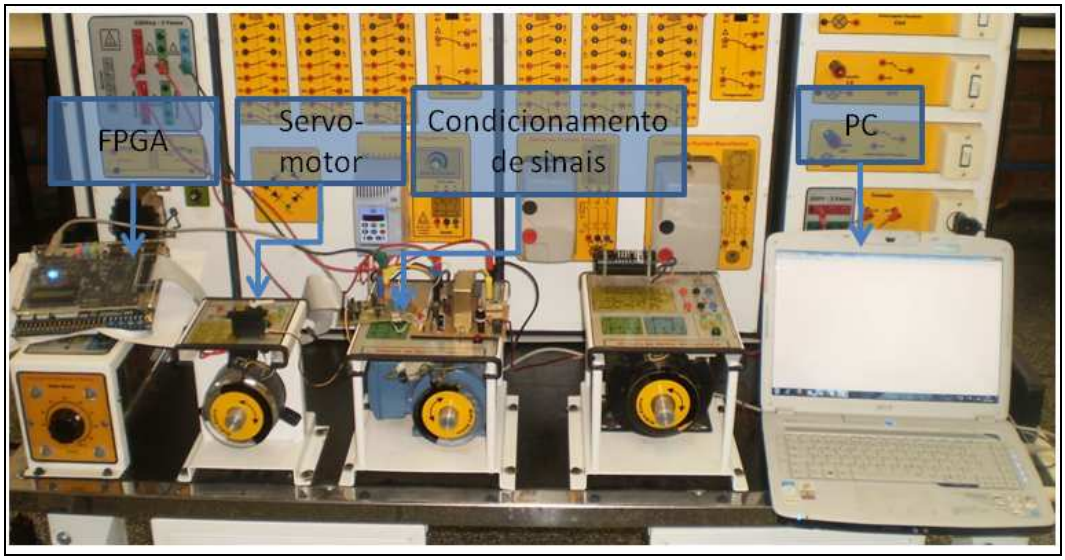


Figura 34 - Protótipo montado no kit de elétrica.

O motor utilizado para o protótipo foi motor de indução em gaiola de 1/3 cv, cujas especificações estão na Figura 35. Sua ligação foi feita em 220 volts trifásico.

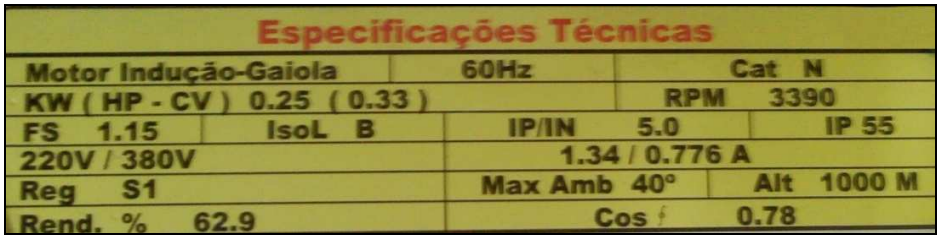


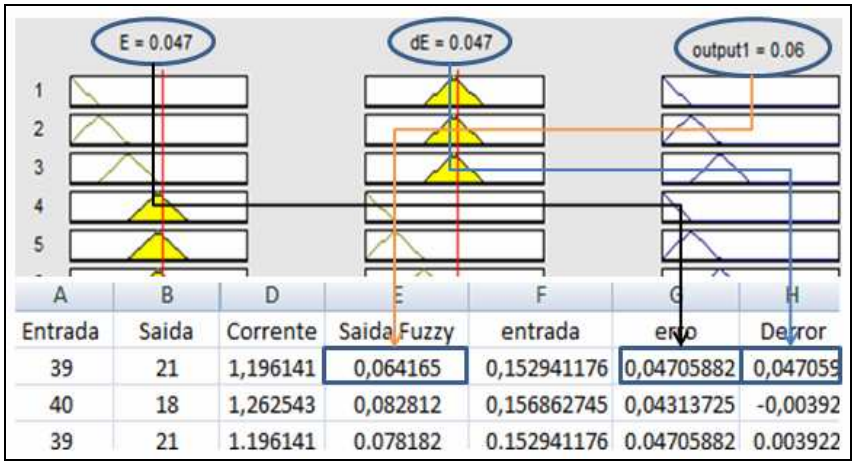
Figura 35 - Especificações do motor do protótipo.

Para futuras aplicações em supervisório desenvolveu-se uma fórmula que resulta na corrente RMS do motor. O software NIOS II IDE proporciona a visualização de certos parâmetros estabelecidos na programação. Na Figura **Erro! Nenhum texto com o estilo especificado foi encontrado no documento.**36, pode-se visualizar o valor de entrada em bits no FPGA, a saída do controlador fuzzy em bits, a corrente RMS (em Ampères) e a saída defuzzificada.

```
Valor [Entrada]: 39 [Saída]: 21 Corrente [A]: 1.196141
out0 0.064165
Valor [Entrada]: 40 [Saída]: 16 Corrente [A]: 1.262543
out0 0.071363
Valor [Entrada]: 40 [Saída]: 18 Corrente [A]: 1.262543
out0 0.071363
Valor [Entrada]: 40 [Saída]: 18 Corrente [A]: 1.262543
out0 0.071363
Valor [Entrada]: 40 [Saída]: 18 Corrente [A]: 1.262543
out0 0.071363
Valor [Entrada]: 40 [Saída]: 18 Corrente [A]: 1.262543
out0 0.082812
Valor [Entrada]: 39 [Saída]: 21 Corrente [A]: 1.196141
out0 0.078182
Valor [Entrada]: 39 [Saída]: 19 Corrente [A]: 1.196141
out0 0.064165
Valor [Entrada]: 40 [Saída]: 16 Corrente [A]: 1.262543
out0 0.071363
Valor [Entrada]: 40 [Saída]: 18 Corrente [A]: 1.262543
out0 0.071363
Valor [Entrada]: 40 [Saída]: 18 Corrente [A]: 1.262543
out0 0.082812
Valor [Entrada]: 39 [Saída]: 21 Corrente [A]: 1.196141
out0 0.064165
Valor [Entrada]: 40 [Saída]: 16 Corrente [A]: 1.262543
out0 0.071363
Valor [Entrada]: 40 [Saída]: 18 Corrente [A]: 1.262543
out0 0.071363
Valor [Entrada]: 40 [Saída]: 18 Corrente [A]: 1.262543
out0 0.071363
Valor [Entrada]: 40 [Saída]: 18 Corrente [A]: 1.262543
out0 0.082812
Valor [Entrada]: 39 [Saída]: 21 Corrente [A]: 1.196141
out0 0.064165
Valor [Entrada]: 40 [Saída]: 16 Corrente [A]: 1.262543
out0 0.071363
Valor [Entrada]: 40 [Saída]: 18 Corrente [A]: 1.262543
out0 0.071363
Valor [Entrada]: 40 [Saída]: 18 Corrente [A]: 1.262543
out0 0.071363
Valor [Entrada]: 40 [Saída]: 18 Corrente [A]: 1.262543
out0 0.071363
Valor [Entrada]: 40 [Saída]: 18 Corrente [A]: 1.262543
out0 0.071363
Valor [Entrada]: 40 [Saída]: 18 Corrente [A]: 1.262543
```

Figura **Erro! Nenhum texto com o estilo especificado foi encontrado no documento.**36 - Captura das correntes pelo NIOS II EDS.

Com a aquisição dos sinais foi possível comparar o modelo prático com o teórico já configurado no software MATLAB versão 2008a. A Figura 37 permite visualizar o quanto a comparação dos valores foi bem sucedida, permitindo possíveis novas configurações de controladores fuzzy.



A	B	D	E	F	G	H
Entrada	Saida	Corrente	Saida Fuzzy	entrada	erro	Derror
39	21	1,196141	0,064165	0,152941176	0,04705882	0,047059
40	18	1,262543	0,082812	0,156862745	0,04313725	-0,00392
39	21	1.196141	0.078182	0.152941176	0.04705882	0.003922

Figura 37 - Comparação dos resultados obtidos no experimento com a simulação.

O protótipo testado possui uma potência de 1/3 cv, limitando o *range* da corrente a ser controlada. Com intuito de tornar o controlador fuzzy mais fácil de ajustar, ganhos foram inseridos em sua programação, caso o projetista necessite ocupar todo o universo de discurso das funções de pertinência.

Este protótipo foi configurado para uma corrente máxima acima de 12 A, ou seja, valores iguais ou maiores que 12 A, a entrada do controlador receberá um (1). Como se pode observar na tabela de aquisição de sinais, os valores de entrada são pequenos devidos à baixa corrente fornecida pelo motor.

Devido à aquisição de 305 amostras foi possível plotar um gráfico com a saída fuzzy gerada pelo erro e sua derivada, conforme apresentado na Figura 38. Pode-se observar que os valores da derivada de erro foram baixos durante muitas amostras, já que o motor em condições normais não poderia variar mais que 2 A. Nota-se que a aquisição final das amostras coincidiu de haver uma elevação expressiva de corrente fazendo com que houvesse

uma variação maior no erro. Contudo, observou-se que o controlador fuzzy logo corrigiu a variação, fechando a válvula, o que na prática significa a diminuição/corte da vazão que alimenta a centrífuga.

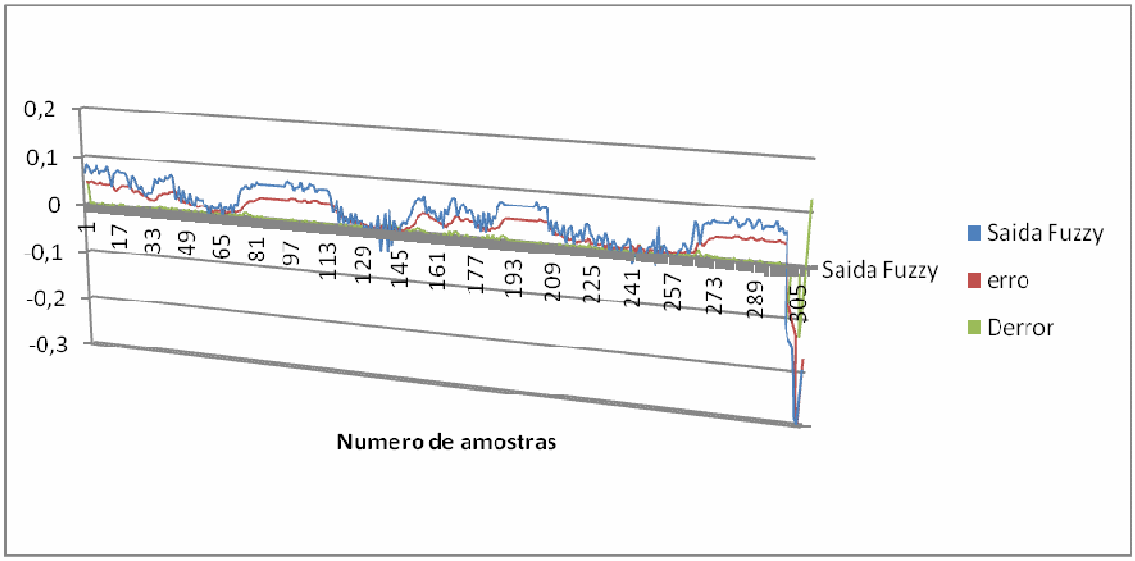


Figura 38 - Saída fuzzy com o erro (*e*) e a derivada do erro (*Δe*).

Como não foram ajustados os ganhos do controlador fuzzy, pode-se verificar na Figura 38 que o universo de discurso ficou entre -0.3 à 0.2. Os ajustes foram feitos após a defuzzificação, utilizando ganhos na entrada do servo-motor.

CAPÍTULO 7 – CONCLUSÕES

O trabalho proposto foi concluído com plena satisfação, alcançando os objetivos propostos como a elaboração do módulo embarcado e simulações requeridas; a técnica utilizada foi modelagem *fuzzy* interagindo com a tecnologia FPGA.

Desta forma, o presente trabalho pode ser aplicado nesta planta, pois os resultados obtidos nos experimentos em bancadas foram satisfatórios, ou seja, conseguiu-se modular a válvula de controle com diferentes ângulos (0º, -45º, 45º, -90º, 90º), a ponto de aumentar ou diminuir a vazão de magma para que o motor trabalhe em carga nominal, com o benefício de não parar o processo.

A técnica proposta consiste de um método genérico, podendo ser aplicada em diversos processos, apenas alterando o *set point* (ponto de referência) de cada motor. No entanto, deve-se salientar que em função das diferentes condições de operações específicas da carga acoplada ao eixo e tipos de motores, a fim de se garantir um melhor desempenho, pode ser necessário o reajuste dos parâmetros do controlador *fuzzy*, incluindo o formato e quantidade de funções de pertinências.

A biblioteca *controlador_fuzzy.c* possui funções de pertinência e regras pré-estabelecidas no processador NIOS II. Apesar de o controlador ser dedicado, suas funções de pertinências e regras poderão ser alteradas para outros limites e/ou formas conforme aplicabilidade do processo, tornando-a uma ferramenta flexível para futuros projetos.

Todavia, para a utilização deste controlador em campo é necessário fazer o teste com motores de maior potência, tornando-o similar aos utilizados em usinas do setor sucro-alcooleiro, sendo, portanto, necessária a utilização de placas de condicionamento de sinais de maior potência. Como proposta de trabalho futuro, o controlador pode ser implementado em um protótipo composto por motores e válvulas com as especificações semelhantes de uma centrífuga de uma usina de açúcar e álcool.

Os resultados apresentados neste trabalho favorecem a aplicabilidade deste hardware de controle para outros setores da automação industrial. A utilização de FPGA no setor industrial converge para atender a necessidade de sistema de controle cada vez mais complexo

e com flexibilidade no desenvolvimento e de manutenção. Portanto, os resultados deste trabalho podem auxiliar na popularização da aplicabilidade de FPGA na indústria.

Adicionalmente, pode-se dizer que este módulo possui reais possibilidades de geração de patente, pois se trata de um controle de vazão de alimentação de uma centrífuga contínua com controlador *fuzzy* embarcado em FPGA.

Este trabalho propiciou a publicação de um artigo no ICECE'2011- VII Conferência Internacional de Educação em Engenharia e Computação sediada em Guimarães, Portugal. Além disso, outro artigo foi submetido e está em processo de análise pela revista IEEE América Latina.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Delgado, A. A., Cesar, M. A. A. (1977). Elementos de tecnologia e engenharia do açúcar de cana.

[2] Deng, J. and Tu, L. (2006). *Improvement of Direct Torque Control Low-speed Performance by Using Fuzzy Logic Technique*. IEEE International Conference on Mechatronics and Automation.

[3] Uddin, M. N. and Hao, W. (2007). *Development of a Self-Tuned Neuro-Fuzzy Controller for Induction Motor Drives*. IEEE Transactions on Industry Applications, vol. 43, pp. 1108-1116.

[4] Cao, Q., Lim, M. H., Li, J. H., et alii. (2006). *A Context Switchable Fuzzy Inference Chip*. IEEE Transactions on Fuzzy Systems, vol. 14, pp. 552-567.

[5] Suetake, M., Goedtel, A. e Silva, N.I. (2010). *Sistema Fuzzy Compacto Embarcado em DSP e sua aplicação para Controle V/F de Motores de Indução*. Revista Controle & Automação, vol. 21, nº 3.

[6] Yen, J., Langari, R. and Zadeth, L. A. (1995). "*Industrial Applications of Fuzzy Logic an Intelligent Systems*", New York: IEEE Press.

[7] Zadeth, L. A., Fu, K., Tanaka, K. and Shimura, M. (1975). "*Fuzzy Sets and Their Applications to Cognitive and Decision Processes*", New York: Academic Press.

[8] LEE, C. C. (1990). *Fuzzy Logic in Control Systems: Fuzzy Logic Controller, Parts I and II*". IEEE Transaction on System, Man and Cybernetics, vol. 20, nº 2, pp 404 – 435.

[9] Simões, M. G. and Shaw, Ian. S. (2007). "*Controle e modelagem fuzzy*". São Paulo: Editora Blucher, FAPESP.

[10] Borgert, G.; Borba, A. J. and Murcia, D. F. (2006). "*Alocação de Custos Indiretos através de um Modelo Experimental de Fuzzy ABC*", Universidade Federal de Santa Catarina - UFSC.

[11] Driankov, D.; Hellendoorn, H. and Reinfrank, M. (1996). "*An Introduction to Fuzzy Control*" Munique: Editora Verlag.

[12] Bilobrovec, M. (2005). "*Sistema Especialista em Lógica Fuzzy para o Controle, Gerenciamento e Manutenção da Qualidade em Processo de Aeração de Grãos*", Programa de Pós-Graduação em Engenharia de Produção - UTFPR, Campus Ponta Grossa.

[13] Silveira P. R. (2004). "*Projeto de Automação de um Sistema de Inspeção de Peças de Cerâmica*", Dissertação do Programa de Pós-Graduação em Engenharia Mecânica da Universidade Federal de Santa Catarina - UFSC.

[14] Fernandes, A. M. R. (1996). "Sistema Especialista Difuso aplicado ao Processo de Análise Química Qualitativa de Amostras de Minerais". Dissertação de Mestrado, Universidade Federal de Santa Catarina - UFSC.

[15] Bandemer, H. and Gottwald, S. (1995). "*Fuzzy Sets, Fuzzy Logic, Fuzzy Methods with Applications*", Chichester: Editora Wiley.

[16] Tusset, M. A. (2008). "*Controle ótimo aplicado em modelo de suspensão veicular não linear controlado através de amortecedor magneto – reológico*". Tese de Doutorado, Universidade Federal do Rio Grande do Sul - UFRGS.

[17] Economakos, C.; Economakos, G. (2008). "*Optimized FPGA Implementations of Demanding PLC Programs based on Hardware High-level Synthesis*", Departamento de Automação, Instituto Tecnológico de Halkis, ISBN: 978-1-4244-1505-2.

[18] Monmasson, Eric and Cirstea, Marcian N. (2007). *FPGA Design Methodology for Industrial Control Systems - A Review.* IEEE Trans. on Industrial Electronics, vol. 54, nº 4, aug-2007.

[19] Plavec, F.; Fort, B.; Zvonko, G.; Vranesic, S. and Brown, D. (2005). *Experiences with Soft-Core Processor Design.* 19th IEEE International Parallel and Distributed Processing Symposium, vol. 4, pp.167b.

[20] Brown, Stephen and Rose, Jonathan. (1996). *Architecture of FPGAs and CPLDs: A Tutorial*". IEEE Design and Test of Computers, vol. 13, nº 2, pp. 42-57.

[21] Sulaiman, Nasri; Obaid, Zeyad Assi; Marhaban, M. H. and Hamidon, M. N. (2009). *Design and Implementation of FPGA-Based Systems - A Review*. Australian Journal of Basic and Applied Sciences, vol. 3, nº 4: 3575-3596, ISSN 1991-8178.

[22] *Differences between Cyclone II & Cyclone Devices*, disponível em: http://www.altera.com/products/devices/cyclone2/features/differences/cy2-differences.html - Acesso em 04/06/2010.

[23] Altera, "*Quartus II Handbook Version 10.0 Volume 4: SOPC Builder*", disponível em: http://www.altera.com/literature/ug/ug_sopc_builder.pdf - Acesso em 05/03/2010.

[24] Altera, "*Avalon Interface Specifications*", disponível em: www.altera.com/literature/manual/mnl_avalon_spec.pdf - Acesso em 10/03/2010.

[25] Altera, "*Nios II Processor Reference Handbook*", disponível em: http://www.altera.com/literature/hb/nios2/n2cpu_nii5v1.pdf - Acesso em 07/05/2010.

[26] Altera, "*Nios II Custom Instruction User Guide"*, disponível em: http://www.altera.com/literature/ug/ug_nios2_custom_instruction.pdf - Acesso em 05/03/2010.

[27] Altera, "*Introduction to the Altera Nios II Soft Processor*", disponível em: ftp://ftp.altera.com/up/pub/Tutorials/DE2/Computer_Organization/tut_nios2_introduction.pdf - Acesso em 10/03/2010.

[28] Monmasson, E. and Chapuis, Y. A. (2002.). *Contributions of FPGAs to the Control of Electrical Systems - A Review*. IEEE Industrial Electronics Society Newsletter. ISSN 0746-1240, vol. 49, nº 4.

[29] Altera, "*DE2 Development and Education Board*", disponível em http://courses.engr.illinois.edu/ece385/documents/DE2_UserManual.pdf - Acesso em 22/03/2010.

[30] Elnatan Chagas Ferreira. Curso IE-763. Sensores e condicionamento de Sinais, disponível em:http://www.cefetsp.br/edu/sertaozinho/professores/Fernando_Fortuna/material_341_3ano.pdf Acesso em 07/07/2010.

[31] Safa Kassap. (2001). "*Hall Effect in semiconductors*" ,Department of Electrical Engineering, University of Saskatchewan, Canadá.

[32] *Current Sensor* ACS755xCB-050, disponível em: http://www.allegromicro.com/en/Products/Part_Numbers/0755/0755-050.pdf - Acesso em 07/07/2010.

[33] ADC0804 8-Bit µP Compatible A/D Converters, disponível em: http://www.national.com/mpf/DC/ADC0804.htm - Acesso em 03/04/2010.