10th Brazilian Congress on Computational Intelligence (CBIC'2011), November 8 to 11, 2011, Fortaleza, Ceará Brazil

CONTROLADOR FUZZY PROGRAMADO EM PROCESSADOR NIOS II EMBARCADO EM FPGA

MACHADO, L. E. B. P., BATISTA, E. A., LEITE, L. C.

Universidade Federal de Mato Grosso do Sul
leandro.basmage@gmail.com , edson@del.ufms.br , luciana@del.ufms.br

Resumo – A contribuição deste trabalho está vinculada à inserção de técnicas de controle fuzzy no processador NIOS II embarcado em FPGA. A aplicabilidade deste hardware de controle relaciona-se ao posicionamento angular de uma válvula de fechamento e abertura a fim de manter a estabilidade de corrente em um motor de indução trifásico de 1/3 cv e 220 V. Trata-se de um protótipo que simula uma centrífuga presente na etapa de fabricação de açúcar, em que a matéria-prima (magma) deve ser centrifugada para dar origem ao açúcar. Eventualmente, este processo sofre interrupções pelo fato do motor ficar em estado de sobre-corrente, fazendo-o desarmar e ocasionando paradas indesejáveis. 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.

Palavras-chaves – controle fuzzy, NIOS II, FPGA, template.

1 Introdução

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 [1]. 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. Como a técnica de controle utilizada foi a fuzzy, foi realizada uma visita a uma usina sucro-alcooleira no município de Maracajú-MS, com intuito de adquirir dos operadores as estratégias utilizadas no setor de centrifugação contínua. Como neste processo havia parâmetros elétricos, foi necessário adquirir as configurações utilizadas pelos engenheiros e técnicos do setor. O maior problema relatado foi que, devido a este processo ser totalmente manual, muitas vezes a vazão de saída (por algum problema específico) era menor que o necessário para determinada vazão de entrada, ou seja, a matéria prima estava aumentando no interior da centrífuga, fazendo com que houvesse um aumento em sua corrente a ponto de ativar sua proteção de sobre-corrente. Desta forma, com a proteção ativada o motor era desligado e a matéria-prima continuava caindo na centrífuga, causando um grande desperdício à usina, já que neste estágio a matéria-prima já sofreu muitas alterações, agregando um considerável valor.

Este controle tem como principal objetivo tornar o processo de centrifugação totalmente automatizado, minimizando as perdas. Possui como atuador uma válvula de controle responsável pelo controle de vazão da matéria-prima (magma) a ser centrifugada e como variável de entrada a corrente da centrífuga. 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.

Alguns trabalhos relacionados com o desenvolvimento e implementação de controladores fuzzy dedicados em hardware são apresentados a seguir. Alguns autores apresentaram a inserção do sistema fuzzy integral totalmente embarcado, visando os ajustes dos parâmetros de um controlador PI aplicados ao controle direto de torque (DTC) de motores de indução (MI) [2]. Outros realizaram o desenvolvimento também integral de um controlador, nesse caso neuro-fuzzy, auto-ajustável em um DSP, utilizando-se a ferramenta Simulink/Matlab para implentar o algoritmo juntamente com o toolbox *real–time*, em que o *software* central-*desk* gerou automaticamente o código do programa para o DSP [3]. Um exemplo de implementação de sistema embarcado dedicado ao processo de inferência *fuzzy* pode ser encontrado em [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].

Na Figura 1 pode-se observar a arquitetura proposta no trabalho.

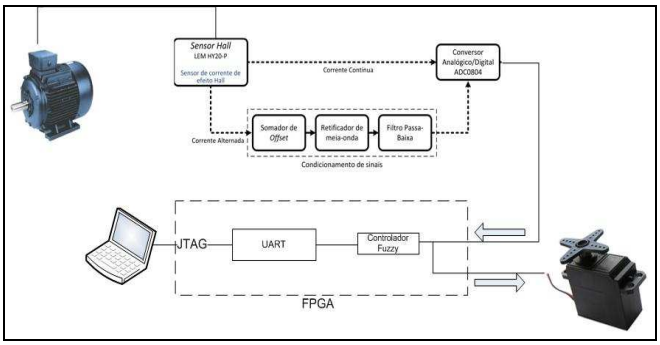


Figura 1 - Arquitetura proposta no trabalho.

Para o desenvolvimento e implementação deste módulo embarcado utilizou-se a placa de desenvolvimento DE2 da Altera, a qual contém um chip Altera Cyclone II e um FPGA com 33.216 elementos lógicos [6].

A organização deste trabalho é feita da seguinte maneira: na seção 2 apresenta-se uma introdução do processo de centrifugação de uma usina sucroalcooleira; na seção 3 detalha-se as características do módulo embarcado desenvolvido, tanto a parte de hardware quanto o software; na seção 4 descreve-se o protótipo da centrífuga contínua; na seção 5 são expostos a simulação desenvolvida no software Simulink e os resultados experimentais obtidos relativo ao controlador fuzzy programado no processador NIOS II e; finalmente, a seção 6 apresenta as conclusões deste trabalho.

2 Processo de centrifugação dos cristais de açúcar

Até a metade do século XIX, a separação dos cristais do licor mãe (magma) era realizada somente por gravidade, em moldes cônicos perfurados, em que, após a drenagem do mel obtinha-se o açúcar. Porém, o tempo gasto nesta operação era maior. Desta forma, 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 nos cristais.

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].

2.1 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 se afasta 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. Então, 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 geral do processo de fabricação do açúcar pode ser visualizado na Figura 2.

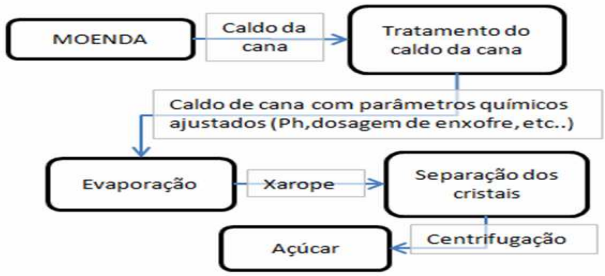


Figura 2 - Etapas da fabricação do açúcar.

3 Módulo embarcado

Um dos maiores problemas dos projetistas de hardware atualmente é o desenvolvimento de sistemas embarcados mais complexos em um espaço de tempo reduzido. Neste cenário uma alternativa interessante é o desenvolvimento em plataforma que consiste de um conjunto de componentes pré-caracterizados, os quais podem ser programados e configurados pelos projetistas, cabendo aos mesmos conhecer o ambiente de desenvolvimento e o hardware que deseja gerar.

A ferramenta utilizada neste projeto foi o SOPC Builder, pertecente ao software QUARTUS II da Altera, possui bibliotecas de componentes que serviram de base para o *hardware* gerado e nomeado "controlador fuzzy".

Para criar um hardware contendo processador *NIOS II* utiliza-se o software *Quartus II* que é um sistema *CAD (Computer Aided Design)*, e o ambiente *SOPC (System-On-a-Programmable-Chip) Builder*.

O módulo embarcado desenvolvido neste projeto recebe a corrente proveniente do motor da centrífuga, a qual é a variável de controle. Como se trata de um controlador fuzzy PD trabalhou-se com o erro e a derivada do erro como variáveis de entrada e o sinal da saída (corrente) é convertido para a faixa de atuação da válvula de controle, que é 4 mA – 20 mA.

O desenvolvimento de um sistema embarcado como o "controlador fuzzy" consiste de duas partes fundamentais: a parte hardware e a parte software, aém do circuito de condicionamento de sinais.

3.1 Geração do hardware

A parte hardware do "controlador fuzzy" é composta de um *Driver* chamado controlador_fuzzy, da PLL e das pinagens necessárias para endereçar as *I/O* deste hardware. O *Driver* controlador_fuzzy favorece a inserção das regras fuzzy (lógica de controle) e gerenciamento das ações de entra e saída de dados. Além da PLL, o "controlador fuzzy" possui um processador *NIOS II/s*, uma memória *SDRAM*, uma interface *JTAG UART (Universal Asynchronous Receiver Transmitter)*, uma entrada de 8 bits proveniente do conversor A/D, 1 saida de 1 bit e um timer. Na Figura 3 mostra-se os componentes devidamente configurados no SOPC Builder para gerar o *controlador_fuzzy*.

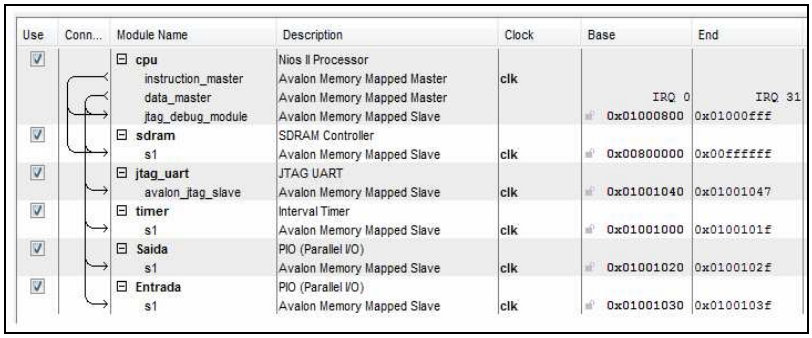


Figura 3 - Configuração do Hardware "controlador fuzzy".

A memória *SDRAM* que compõe o "controlador fuzzy" possui capacidade de armazenamento de 8 *MBytes*, e foi configurada de modo a possuir 12 linhas por 8 colunas de endereçamento, 16 bits de dados, 4 bancos *chipselect*. Esta memória foi necessária pois a aplicação no *NIOS II IDE* excede o tamanho das memórias *on-chip*.

Foram realizados testes para mensurar a eficiência dos processadores, e para este processo em particular o processador NIOS /s foi o que teve uma melhor perfomance, por permitir alocação das instruções memória *cache*, utilização de sistema de *pepiline* e execução de previsão de desvio *(Branch Prediction)*, características ausentes na versão NIOS/*e*.

O PLL(*Phase-Locked Loop*) 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.

3.2 Geração do firmware

Após a implementação do hardware "controlador_fuzzy", foi desenvolvido o firmware deste módulo embarcado, responsável pela lógica de controle. O algoritmo do controlador fuzzy foi implementada no software NIOS II IDE, cuja linguagem do desenvolvimento foi feita em C.

De forma geral, para aplicações em controle de processos o número ideal de funções de pertinência pode ser considerado entre o mínimo de 2 (duas) e máximo de 7 (sete). Considera-se ainda que neste intervalo quanto maior o número de funções de pertinência, maior será a precisão, consequentemente, exige-se um maior esforço computacional [7]. Tratando-se de um processador NIOS II/s e 8 Mbytes de memória optou-se por inserir o número máximo deste intervalo. Assim, no processador NIOS II inseriu-se a lógica do controlador fuzzy com 7 (sete) funções de pertinências triangulares e eqüidistantes. Desta forma, foram criadas matrizes cujo conteúdo seriam os limites e os coeficientes das funções de retas que as compõem. Estas funções de pertinências foram nomeadas como **g**, **m**, **p**, **Z**, **P**, **M**, **G**, o qual significam negativo grande, negativo médio, negativo pequeno, zero, positivo pequeno, positivo médio, positivo grande, respectivamente. Estas características podem ser visualizadas na Figura 4.

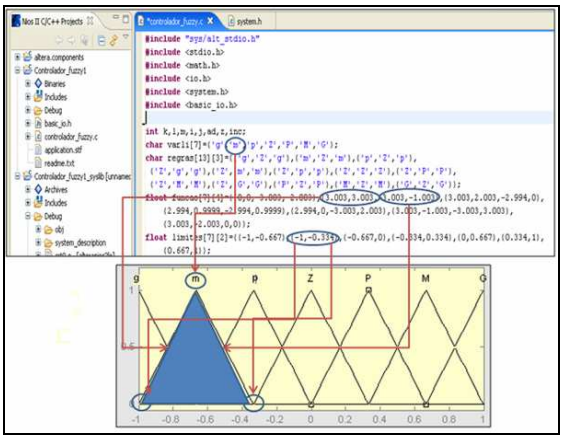


Figura 4 - Características do controlador fuzzy.

A fim de obter transições mais suaves no sistema fuzzy, todas as funções de pertinências utilizadas foram configuradas para uma sobreposição de 50%. Por conseguinte, um método eficaz de defuzzificação utilizado neste processo de automação e para essa configuração é o COA (*center of area)* [7]. Para obter agilidade no sistema desenvolveu-se uma biblioteca já com as funções do controlador fuzzy chamada controlador_fuzzy.c.

O controlador é dotado de 16 regras e pode-se dizer que é flexível, ou seja, as mesmas podem ser alteradas e/ou adicionar novas regras dependendo da aplicação e conveniência, assim como os limites das funções de pertinência e suas funções. Portanto, foi desenvolvido um controlador fuzzy genérico e de fácil utilização para futuros projetistas. As regras utilizadas para este projeto são dadas por:

R_1:	SE erro = g	and	Δerro = g	THEN	saída = PG
R_2:	SE erro = m	and	Δerro = g	THEN	saída = PG
R_3:	SE erro = p	and	Δerro = g	THEN	saída = PG
R_4:	SE erro = Z	and	Δerro = g	THEN	saída = g
R_5:	SE erro = Z	and	Δerro = m	THEN	saída = m
R_6:	SE erro = Z	and	Δerro = p	THEN	saída = p
R_7:	SE erro = Z	and	Δerro = Z	THEN	saída = Z
R_8:	SE erro = g	and	Δerro = Z	THEN	saída = g
R_9:	SE erro = m	and	Δerro = Z	THEN	saída = m
R_{10}:	SE erro = p	and	Δerro = Z	THEN	saída = p
R_{11}:	SE erro = P	and	Δerro = Z	THEN	saída = P
R_{12}:	SE erro = M	and	Δerro = Z	THEN	saída = M
R_{13}:	SE erro = G	and	Δerro = Z	THEN	saída = G
R_{14}:	SE erro = Z	and	Δerro = P	THEN	saída = P
R_{15}:	SE erro = Z	and	Δerro = M	THEN	saída = M
R_{16}:	SE erro = Z	and	Δerro = G	THEN	saída = G

3.2.1 Fuzzificação

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 associada a rótulos lingüísticos [7]. Para a realização do algoritmo de fuzzificação fez-se a varredura nas 16 regras, utilizou-se os coeficientes de retas e limites das funções de pertinências, comparando o valor do erro e derivada do erro. Na Figura 5 ilustra-se o algoritmo responsável por "guardar" o erro de uma regra para, então, compará-lo com o erro anterior (derivada do erro) desta mesma regra. O menor será utilizado para fazer o cálculo da área já modificada por essa inferência.

```
if(erro<=limites[j+1][0])
{

    temp=funcao[j][0]*erro+funcao[j][1];
}
if(erro>limites[j+1][0])
{
    temp=funcao[j][2]*erro+funcao[j][3];
```

Figura 5 - Parte do algoritmo responsável por fazer o cálculo da inferência fuzzy.

3.2.2 Defuzzificação

O COA (centro de área), método utilizado neste projeto, 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 das regras ativadas sob uma determinada condição. O centróide é um ponto que divide a área de μ_{out} em duas partes iguais. Este cálculo é feito através da equação 1.

$$u = \frac{\sum_{j=1}^{N} u_j \, \mu_{OUT}(u_j)}{\sum_{j=1}^{N} \mu_{OUT}(u_j)} \qquad (1)$$

em que μ_{out} é a área de uma função de pertinência modificada pelo resultado da inferência fuzzy, u_i é a posição do centróide da função de pertinência individual.

Para programar tal algoritmo utilizou-se o cálculo das áreas, já modificadas pela inferência, e o cálculo dos centróides das 16 regras fuzzy. O cálculo da área é feito através de uma integral que divide a área em ***x*** retângulos de base 0.01, conforme é apresentado na Figura 6.

```
if(m!=0)
{
    for(a=limites[m][0];a<x1;a=a+0.01)
    {
        area[k]=area[k]+(funcao[m][0]*a+funcao[m][1])*0.01;
    }
}
else
{
    for(a=limites[m][1];a<x2;a=a-0.01)
    {
        area[k]=area[k]+(funcao[m][2]*a+funcao[m][3])*0.01;
    }
}
```

Figura 6 - Algoritmo responsável pelo cálculo da área.

A determinação do centróide é feita calculando-se o ponto que divide a área em duas partes iguais. Parte do algoritmo que executa tal função é ilustrada na Figura 7.

```
coamm1=limites[m][0]+X/2.0;
var=(x1-limites[m][0]);
coamm2=limites[m][0]+(var*(2.0/3.0));
COA[k]=((coamm1*X*y)+(coamm2*area[k]))/((X*y)+area[k]);
```

Figura 7 - Parte do algoritmo responsável pelo cálculo do centróide.

Já com os valores de u_{out} e u_i armazenados em memória é possível calcular o centro de área (COA) para, então, enviar a saída defuzzificada ao supervisório e à válvula de controle. A equação (1) programada no processador NIOS é vista na Figura 8.

```
    temp=temp+(area[i]*COA[i]);
    temp1=temp1+area[i];
}

out=temp/temp1;
```

Figura 8 - Defuzzificação por centro de área (COA).

4 Resultados

A ferramenta utilizada na simulação deste processo foi o *toolbox* Simulink do software MATLAB. Nela, modelou-se um motor similar ao testado na prática, porém, com ajustes nos ganhos. O diagrama completo pode ser visualizado na Figura 9. Para "emular" uma carga semelhante à de uma centrífuga contínua utilizou-se uma seqüência interpolada com repetições. Na Figura 10 é possível notar a mudança exercida pela carga "emulada" no torque eletromagnético.

O objetivo desta simulação é de produzir uma carga com características semelhantes ao do motor testado na prática e comparar os valores obtidos com intuito de validar o firmware desenvolvido.

Foi necessário utilizar, na saída do controlador fuzzy, um bloco denominado "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 e 20 mA.

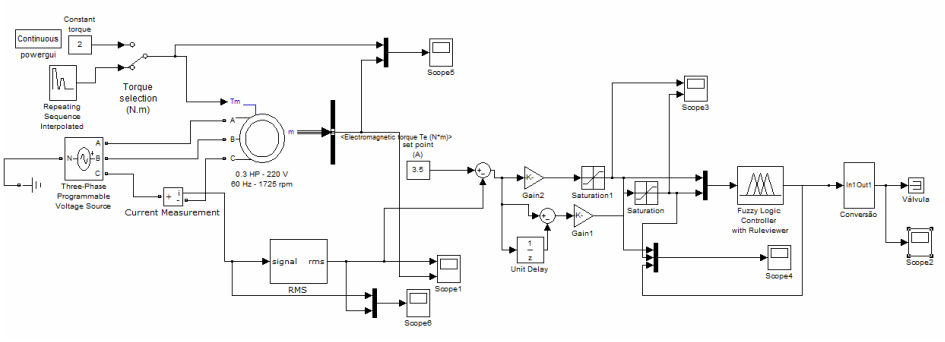


Figura 9 - Sistema proposto para automação.

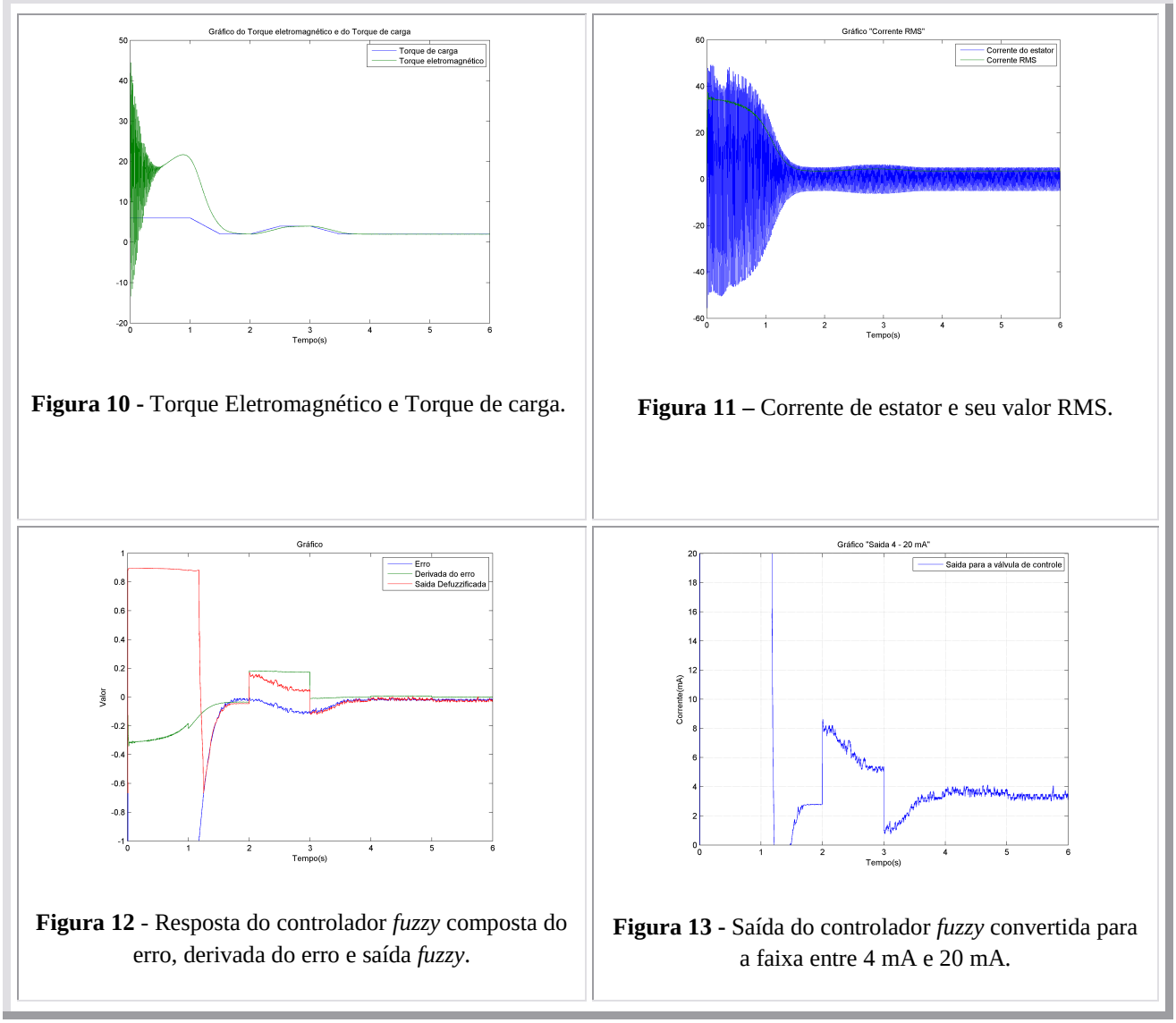


Figura 10 - Torque Eletromagnético e Torque de carga.

Figura 11 – Corrente de estator e seu valor RMS.

Figura 12 - Resposta do controlador *fuzzy* composta do erro, derivada do erro e saída *fuzzy*.

Figura 13 - Saída do controlador *fuzzy* convertida para a faixa entre 4 mA e 20 mA.

Essa variação de corrente tornou possível visualizar o comportamento do controlador *fuzzy*. Observa-se na Figura 12 a saída do controlador, a variação do erro e o erro. Pode-se também observar que com a corrente de partida o controlador fuzzy enviou um 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 13 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)

5 Testes Experimentais

Para a realização de testes no controlador fuzzy, desenvolveu-se um protótipo a partir de um motor de indução trifásico de 220 V e 1/3 cv, similar aos utilizados em processos industriais, porém, com uma potência menor.

O protótipo foi montando em um kit de instalações elétricas industriais, 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 12.

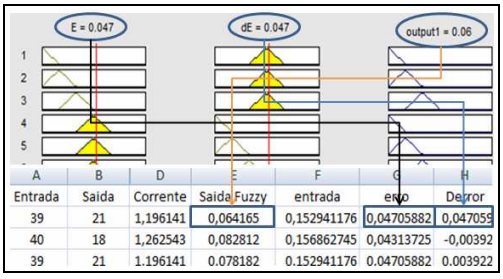
Figura 12 – Protótipo montado: servo motor, condicionamento de sinais, FPGA e interação com o PC.

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 13 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
```

Figura 13 - Captura das correntes pelo NIOS II EDS.

Com a aquisição dos sinais, apresentada na Figura 15, foi possível comparar o modelo prático com o teórico (simulado) já configurado no software MATLAB 2008a. A Figura 14 permite visualizar o quanto foi bem sucedida a comparação dos valores, 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 14 - Comparação dos resultados obtidos no experimento com a simulação.

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

Este protótipo foi configurado para uma corrente máxima de 10A, ou seja, valores iguais ou superiores a 10, a entrada do controlador receberá um (1). Como se pode observar na tabela de aquisição de sinais (Figura 15), os valores de entrada são pequenos devidos à baixa corrente fornecida pelo motor. Tornou-se possível, com a aquisição de 305 amostras, plotar a saída fuzzy gerada pelo erro e sua derivada, apresentado na Figura 17. 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 na aquisição final das amostras coincidiu de haver uma elevação expressiva de corrente, gerando 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.

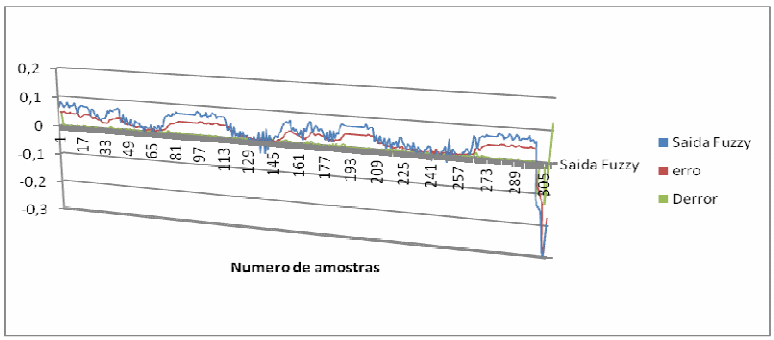


Figura 15 - Saída *fuzzy* com o erro (*e*) e a derivada do erro (*Δe*).

Como não foram ajustados os ganhos do controlador fuzzy, verifica-se na Figura 15 que o universo de discurso ficou entre -0.3 à 0.2. Os ajustes foram feitos após a defuzzificação, utilizando ganhos na entrada válvula de controle.

6 Conclusões

O desenvolvimento do módulo embarcado, as simulações requeridas e a inserção das técnicas fuzzy utilizadas na modelagem de uma válvula de controle interagindo com a tecnologia FPGA mostram que os objetivos propostos foram concluídos com plena satisfação.

Desta forma, o presente trabalho pode ser aplicado numa planta do setor sucro-alcooleiro, pois os resultados obtidos nos experimentos 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 da matéria-prima (magma) para que o motor atue com carga nominal e, portanto, não havendo interrupções no processo.

A técnica proposta consiste de um método genérico e flexível, 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.

No processador NIOS II foi possível inserir as funções de pertinência e regras pré-estabelecidas, em que as mesmas podem ser reconfiguradas com outros limites ou formas, favorecendo a aplicabilidade deste hardware de controle em diversos setores da automação industrial. Assim, criou-se o template "controlador_fuzzy.c", tornando-o uma ferramenta flexível para futuros projetos.

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 e possuem reais possibilidades de geração de patentes, pois trata de um controle de vazão de alimentação de uma centrífuga contínua com controlador fuzzy embarcado em FPGA.

Referência Bibliográfica

[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, 3.
[6] 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.
[7] Simões, M. G. and Shaw, Ian. S. (2007). "Controle e modelagem fuzzy". São Paulo: **Editora Blucher**, FAPESP.