UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS DE COMUNICAÇÃO E AUTOMAÇÃO

ALLYSON ARILSON LIMA FILGUEIRA

ESTUDO DE DESEMPENHO DE UM CONTROLADOR *FUZZY* DESCRITO EM VHDL PARA UM SISTEMA DE TANQUES

MOSSORÓ – RN

2017

ALLYSON ARILSON LIMA FILGUEIRA

ESTUDO DE DESEMPENHO DE UM CONTROLADOR *FUZZY* DESCRITO EM VHDL PARA UM SISTEMA DE TANQUES

Dissertação de mestrado acadêmico apresentada ao Programa de Pós-Graduação em Sistemas de Comunicação e Automação, como requisito para a obtenção do título de Mestre em Sistemas de Comunicação e Automação.

Orientador: Prof. Dr. Marcelo Roberto Bastos Guerra Vale – UFERSA
Co-orientador: Prof. Dr. Leonardo Augusto Casillo - UFERSA

MOSSORÓ – RN
2017

F478e Filgueira, Allyson Arilson Lima.
ESTUDO DE DESEMPENHO DE UM CONTROLADOR FUZZY DESCRITO EM VHDL PARA UM SISTEMA DE TANQUES / Allyson Arilson Lima Filgueira. - 2017.
73 f. : il.

Orientador: Marcelo Roberto Bastos Guerra Vale.
Coorientador: Leonardo Augusto Casillo.
Dissertação (Mestrado) - Universidade Federal Rural do Semi-árido, Programa de Pós-graduação em Sistemas de Comunicação e Automação, 2017.

1. Controle de nível. 2. Lógica fuzzy. 3. VHDL. I. Vale, Marcelo Roberto Bastos Guerra , orient. II. Casillo, Leonardo Augusto, co-orient. III. Título.

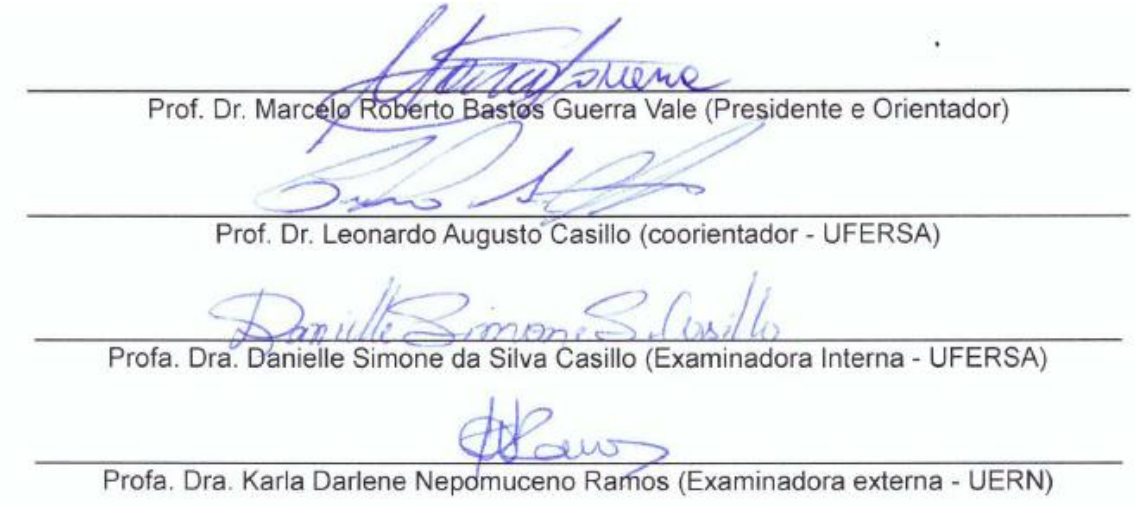
ALLYSON ARILSON LIMA FILGUEIRA

ESTUDO DE DESEMPENHO DE UM CONTROLADOR *FUZZY* DESCRITO EM VHDL PARA UM SISTEMA DE TANQUES

Dissertação de mestrado acadêmico apresentada ao Programa de Pós-Graduação em Sistemas de Comunicação e Automação, como requisito para a obtenção do título de Mestre em Sistemas de Comunicação e Automação.

APROVADA EM: 25 /08 /2017

BANCA EXAMINADORA



Mossoró, 25 de agosto de 2017

AGRADECIMENTOS

A Deus, por toda a sabedoria, pelo dom da vida e por permitir que este sonho seja concretizado.

Aos meus pais, pelo amor incondicional e por abdicar de várias coisas para que eu pudesse ter o necessário para minha formação como profissional e cidadão.

Aos meus orientadores, Marcelo Guerra e Leonardo Casillo, pelas contribuições e pelas palavras motivadoras em momentos de desmotivação, e aos demais membros da banca por todas as críticas construtivas.

À minha namorada, Carla Djaine, pelo grande incentivo para o término deste trabalho, pois sem ela, isto não seria possível, além da paciência e compreensão enquanto estive ausente para o desenvolvimento desta pesquisa.

Aos meus colegas de mestrado: Felipe Bezerra, Alisson Cunha Anamaria Sena, Diêgo Pires, Flávia Dantas, Edpo Rodrigues e Cássio Colaça, por todos os dias de estudo e trabalhos feitos durante a pós-graduação.

Aos meus colegas de graduação, que são modelos de profissionais para mim: Daniel Carlos, Samanta Holanda, Arimatéia Magno, Moisés Honorato, Talles Amony, Juan Guerra, Victor Lucena, Isaú Balbino, Thomas Tadeu, Manassés Rocha, Camila Lopes, Renan Abdon, Michael Carvalho, Tulio Morais, Diego César, João Marcos, João Paullo e demais companheiros que apenas uma folha desta não seria possível para citar aqui.

Aos meus colegas de IFRN, pelos quais sempre serei grato pela companhia e conhecimento compartilhado.

Aos familiares e aos demais que direta ou indiretamente torceram para esta conquista pessoal.

Agradeço à UFERSA, pela infraestrutura, e à CAPES, pelo auxílio financeiro.

“Não é o conhecimento, mas o ato de aprender, não é a posse, mas o ato de chegar lá que garantem maior satisfação. Quando esclareci e exauri um assunto, o deixei de lado para mergulhar novamente na escuridão. ”

(Carl Friedrich Gauss)

RESUMO

Com a necessidade de desenvolver controladores para processos industriais, surge a necessidade de estudar técnicas de controle avançado, como por exemplo, controladores *fuzzy*, que possam suprir não linearidades inerentes nos sistemas. A implementação destes controladores *fuzzy* baseados em Linguagem de Descrição de *Hardware (HDL)* tem como vantagem a alta velocidade e a reprogramação rápida para utilização em outro projeto, alterando apenas variáveis desejadas. O objetivo do trabalho é desenvolver um controlador *fuzzy* implementado em VHDL (*Very High Speed Integrated Circuit Hardware Description Language)* para um sistema de nível de tanques e avaliar sua eficácia, comparando-o com um controlador *fuzzy* desenvolvido em Matlab®. Sua metodologia consiste em desenvolver um sistema *fuzzy* descrito em VHDL e divididos em codificação, inferência e decodificação para um sistema de tanque cilíndrico da Quanser e outro sistema com tanque de formato trapezoidal. Os resultados desta proposta se mostraram satisfatórios diante de limitações apresentadas pelo sistema.

Palavras-chave: Controle de nível, Lógica *fuzzy*, VHDL.

ABSTRACT

Along with the need to develop controllers for industrial processes, it arises the need to study advanced control techniques, such as fuzzy controllers, that can supply nonlinearities inherent in the systems. The implementation of these fuzzy controllers based on Hardware Description Language (HDL) has the advantage of high speed and fast reprogramming for use in another project, by changing only desired variables. The objective of this study is to develop a fuzzy controller implemented in VHDL (Very High Speed Integrated Circuit Hardware Description Language) for a tank level system and to evaluate its effectiveness, comparing it with a fuzzy controller developed in Matlab®. Its methodology is to develop a fuzzy system described in VHDL and divided into coding, inference and decoding for a Quanser cylindrical tank system and another system with a trapezoidal format tank. The results of this proposal were satisfactory given the limitations presented by the system.

Keywords: Level controlling. Fuzzy logic. VDHL

.

LISTA DE TABELAS

LISTA DE FIGURAS

NOMENCLATURA

Letras Latinas

Vp – Tensão aplicada na bomba	(Volt)
Km – Constante da bomba	(Q/Volt)
G – Gravidade	(m/s^2)
L - Altura do volume de água	(h)
L’ - Variação do nível dos tanques	(Q/s)
V’- Variação volumétrica	(m^3/s)
V_{out} – Vazão de saída	(cm/s)
Fout – Força de saída	(cm^3/s)
Fin – Força de entrada	(cm^3/s)
a – Área do orifício de saída	(cm^2)
A – Área da secção transversal	(cm^2)

Abreviações

PID – Proporcional Integral Derivativo

PI – Proporcional Integral

ENG – Erro Negativo Grande

ENP – Erro Negativo Pequeno

EZ – Erro Zero

EPP – Erro Positivo Pequeno

EPG – Erro Positivo Grande

DENG – Derivada Erro Negativo Grande

DENP – Derivada Erro Negativo

DEZ – Derivada Erro

DEPP – Derivada Erro

DEPG – Derivada Erro

VHDL – *Very High Speed Integrated Circuit Hardware Description Language*

FPGA – *Field Programmable Gate Array*

Sumário

1. INTRODUÇÃO

A automação do processo industrial e o processamento do sinal industrial implicam a necessidade de desenvolver sistemas de aquisição de dados e sistemas de controle. Diante disso, é necessário encontrar soluções técnicas para os problemas oriundos das grandes indústrias. Guarnizo e Avendano (2017) mencionam que os sistemas de controle oferecem uma alternativa ao desenvolvimento de soluções para problemas industriais. Muitos desses problemas podem ser resolvidos usando ferramentas que envolvam controle clássico, controle *fuzzy*, redes neurais, entre outros.

Segundo Jian-Jun (2014), comparado com as abordagens de controle convencional, o controle *fuzzy* de objetos não-lineares funciona melhor que o controlador do tipo PID (Proporcional Integrativo Derivativo) e é mais adaptável no processo de controle de nível de líquido devido à modelagem matemática não ser necessariamente especificada, além do controle de qualidade ser menos afetado pelas características de plantas controladas e mudanças de parâmetros. Este tipo de conclusão também pode ser verificado em Vieira (2017).

A vantagem da implementação de um controlador lógico *fuzzy* em um ASIC *(Application Specific Integrated Circuit)* é a alta velocidade, pois uma vez testado, os chips são de baixo custo para serem produzidos. Um fator negativo seria o tempo para criar o chip, que depois de criado, não pode ser modificado (SINGH AND RATTAN, 2003). Uma das maneiras de resolver este problema, é a implementação desta linguagem em um FPGA (*Field Programmable Gate Array*), que é um circuito digital integrado que pode ser programado para implementar funções digitais variadas. Aguilar et al (2014) caracteriza o FPGA pela simplicidade de reprojetar o controle baseado nas especificações sem ficar sujeito a uma engenharia não recorrente, além da capacidade de processamento em paralelo, obtendo-se um processamento em tempo real.

Usando VHDL para a implementação, o comportamento do sistema *fuzzy* pode ser representado por diferentes maneiras, por exemplo como um diagrama de sinal, ou para ser usado como apoio em um processamento, pois de acordo com Todinca e Butoianu (2012), esta linguagem é de programação em alto nível, no qual se permite uma descrição do comportamento do sistema a um alto nível de abstração. Além disso, é possível simular o comportamento tanto do controlador quanto da aplicação a ser controlada.

Neste trabalho, serão utilizados dois sistemas de tanque distintos, um de formato cilíndrico, sendo este o mais convencional, e outro com formato de trapézio, desenvolvido em escala laboratorial para averiguar a capacidade de adaptação dos controladores (Controlador *fuzzy* desenvolvido por Vieira (2017) no *Software* Matlab® e o *fuzzy* em VHDL) com relação a alteração da área da seção transversal do reservatório. A avalição da resposta quanto ao controle de nível surge da necessidade de resolver determinados tipos de problema em sistemas reais, como a linearização dos sistemas, onde há perdas de produtividade (FIGUEIREDO & JOTA, 2004; BARROS et al, 2006 Apud VIEIRA, 2017). A relevância deste trabalho se dá pelo estudo da viabilidade da implementação em VHDL de um controlador *fuzzy* Matlab® destinado a um FPGA, que poderá atuar em dois sistemas de nível distintos, variando o formato geométrico do tanque, onde solucionando este problema em simulação, posteriormente, poderá solucionar problemas em sistemas reais.

1.1 OBJETIVOS

Os objetivos do trabalho são divididos em geral, onde há uma explicação geral sobre o que está sendo proposto nesta pesquisa, e para alcançar este objetivo geral, foram elaborados objetivos específicos com o intuito de mostrar como o trabalho foi desenvolvido através de determinadas etapas.

1.1.1 Objetivo geral

Realizar um estudo comparativo de desempenho entre um controlador *fuzzy* Matlab e um controlador *fuzzy* descrito em VHDL, considerando sua modelagem matemática e verificando a eficácia deste tipo de implementação

1.1.2 Objetivos específicos

- Descrever o sistema *fuzzy* proposto em VHDL
- Modelar os sistemas de tanques no Matlab®;
- Comparar e analisar os resultados obtidos entre os controladores.

1.2 ORGANIZAÇÃO DO TRABALHO

O capítulo 2 apresenta a revisão da literatura, onde é feita a revisão teórica de conteúdos relativos para se ter um embasamento técnico acerca do tema. O capítulo 3 se refere à descrição do processo de modelagem dos tanques, implementação *fuzzy* descrito em VHDL e simulação do modelo de tanques propostos. O capítulo 4 trata sobre os resultados e discussões referentes à aplicação do controle desenvolvido neste trabalho. Por último, no capítulo 5, são feitas as conclusões do trabalho baseados nos resultados obtidos e propostas para pesquisas futuras, bem como sugestões para futuras pesquisas que envolvam a descrição de determinado controle *fuzzy* em VHDL.

2. REVISÃO DA LITERATURA

Este capítulo apresenta uma revisão sobre os assuntos envolvidos na pesquisa e como estão sendo explorados, para um melhor entendimento deste trabalho. Será feito um levantamento breve de estudos feitos na área, onde se possa situar a relevância desta pesquisa.

2.1 CONTROLE DE NÍVEL

Sistemas de nível de líquidos têm destaque em diversos ramos da atividade industrial, dentre eles o da petroquímica, nuclear e de celulose. Neste contexto, um dos controles com maior importância nas unidades industriais é o dos níveis (RAMOS et al, 2008). Com o conhecimento do processo de produção, um preciso controle de nível acarretará economia de material e tempo que resultam diretamente no aumento do lucro e da eficiência do sistema como todo.

Segundo Campos e Teixeira (2010), a geometria do reservatório também é importante para o controle. Na prática, podem-se encontrar esferas, cilindros horizontais e verticais. Controles em vasos cilíndricos verticais são mais simples de operar, pois independentemente de onde esteja o ponto de controle (*Setpoint)*, a resposta dinâmica do processo ou o aumento do nível será o mesmo para uma mesma variação de vazão de entrada. A Figura 1 mostra alguns tipos de tanques encontrados em uma indústria petroquímica

Figura 1 – Tipos de tanques quanto à sua geometria

Fonte: Adaptado de Mello (2015)

Entretanto, no caso de uma esfera, ao se controlar o nível no meio desta, tem-se uma resposta muito mais lenta em comparação com o caso de se controlar o nível próximo as extremidades. Esta não linearidade acarreta uma dificuldade para o ajuste dos parâmetros de sintonia de controladores clássicos, como o PID (CAMPOS & TEIXEIRA, 2010).

No trabalho de André et al (2013), foi feita a otimização de um controlador *fuzzy* usando um algoritmo genético. Este controle foi testado em uma planta de tanques acoplados da Quanser, na qual os tanques são de formato cilíndrico, e é usado uma bomba para controlar o nível entre eles. O tipo de controle *fuzzy* é Takagi-Sugeno-Kang com estratégia Proporcional Derivativa (PD). Os resultados mostraram que o algoritmo genético demonstrou ser uma técnica eficiente na busca da sintonia para controladores *fuzzy*.

Deepa e Sivakumar (2017) desenvolveram um controlador inteligente heurístico PI-*fuzzy* com um Sistema de Interface Adaptativa Neuro *Fuzzy* (termo em inglês referente a ANFIS). Este modelo é proposto para o controle de nível de um tanque esférico com diferentes pontos de operação. A performance deste controlador é comparada com a de outros tipos de controle, como controladores baseados em Rede Neural Artificial, Lógica *Fuzzy* e em Matrizes Dinâmicas, e conclui-se através dos resultados obtidos que o controlador PI-*fuzzy* foi mais adaptativo para este tipo de controle baseado na dinâmica da geometria esférica do tanque.

Blachuta et al (2017) faz uma análise detalhada de um controlador PI para um sistema de nível que está incluso um tanque de formato cilíndrico. Parâmetros como a atenuação da perturbação da carga e mudança de valores de referência são levados em consideração. Para a modelagem do tanque, foi baseado na solução das equações diferenciais não lineares que descrevem o seu escoamento de fluido, e após a sintonização do controle PI, as considerações apresentadas pelo autor foram simples o suficiente para este trabalho ser um modelo didático para um problema de engenharia, podendo ser feito até um ambiente virtual interativo.

Guarnizo e Avendano (2014) desenvolveram uma ferramenta estudantil para a prática de lógica *fuzzy*, focada no controle *fuzzy*, para a aplicação em um sistema de nível de tanque de formato cilíndrico, onde este é controlado através do *software* Matlab® e do bloco *Fuzzy Logic Toolbox*. Os resultados experimentais mostraram que a proposta

pedagógica permitiu com que os estudantes de graduação possam testar a lógica *fuzzy* para este sistema de nível, focado nos conceitos teóricas aplicados em um processo real. Isto reforça que a lógica *fuzzy* apresenta uma alternativa para o controle de sistemas não lineares através de uma maneira intuitiva.

2.2 APLICAÇÕES DA LÓGICA *FUZZY*

As aplicações da lógica *fuzzy* abrangem diversas linhas de pesquisas, atuando nas áreas de sistemas inteligentes, telecomunicações, proteção de sistemas elétricos, biomédica, dentre outras. Devido a sua capacidade de abordar não linearidades do sistema, ela se torna uma interessante estratégia de controle e classificação nos ramos da ciência, e para reforçar esta ideia, serão mostrados alguns trabalhos nos quais a lógica *fuzzy* foi usada em diversas áreas do conhecimento

Nasibov et al (2016) desenvolveu um classificador baseado em lógica *fuzzy* para indicar ao cliente qual o melhor tamanho da roupa, baseado nas medidas corporais e a marca escolhida. Dessa forma, o número de devoluções e trocas iria diminuir, causando uma margem de lucro maior devido ao fator logística. Para que a tomada de decisão tenha sido correta, as regras foram estipuladas baseadas em fatores estatísticos levantados pelo autor. As funções de pertinência são de formas triangulares, e são classificadas em: “pequeno”, “médio”, “grande”, “muito grande”. O comprador também precisa inserir alguns valores como medida dos ombros, cintura, peitoral etc. Ao fim do trabalho, os autores fazem um teste e mostram que o controle baseado em lógica *fuzzy* se mostrou eficaz.

O trabalho de Cruz et al (2017) foi de desenvolver um sistema híbrido baseado na lógica *fuzzy* para diagnosticar falhas em motores de indução, que usa os dados obtidos de valores de vibração e de sensores de corrente para detectar falhas em um estágio inicial. Os sinais são processados no domínio da frequência e do tempo através de uma técnica chamada *Short Time Fourier Transform* (STFT) e da análise multi-resolução *Wavelet*, na qual fornecem as entradas para um sistema inteligente baseado na lógica *fuzzy*.

Nos resultados, os arquivos gerados com a simulação foram testados na prática, onde para os cenários analisados, o sistema conseguiu detectar a falha de

desbalanceamento e falha de barras quebradas em sua condição de severidade em 100% dos casos. Na conclusão, o trabalho contribui para uma nova técnica de detecção de falhas em motores, aumentando a confiabilidade na detecção do diagnóstico do motor, seja de desbalanceamento, seja nas barras quebradas.

Ghosh et al (2017) apresentou um sistema baseado em lógica *fuzzy* para analisar um estilo de vida estudantil categorizando-o em temas como “educação”, “saúde” “socializar” e outras atividades nas quais podem ser incluídas em tarefas do dia-a-dia. O trabalho mostrou como uma aplicação móvel usando lógica *fuzzy* e o Sistema de Posicionamento Global (GPS) analisa o estilo de vida do usuário do sistema e provê sugestões e recomendações baseados nos resultados. Os dados de entrada são coletados através de um aplicativo que identifica onde o usuário está e o tempo gasto naquele local. Por fim, analisando as categorias já mencionadas, o programa fornece uma distribuição de tempo que deve ser gasto de acordo com seu tempo gasto naquilo.

O trabalho de Várkonyi-Kóczy et al (2017) consiste no desenvolvimento de um classificador baseado em lógica *fuzzy* suportado em modelagem 3D baseada em ortodontia, onde as posições dos dentes são mapeadas com o uso de aparelho de raio-x, e comparado com a posição ideal dos dentes, o controle mostra a posição ideal tanto dos dentes, quanto da raiz, após a devida correção com o uso do aparelho dental. Ao aplicar a nova técnica de modelagem de processo baseada em modelagem de processamento 3D, o tratamento ortodôntico resulta em locais de dentes estáveis, permanentes e estéticos.

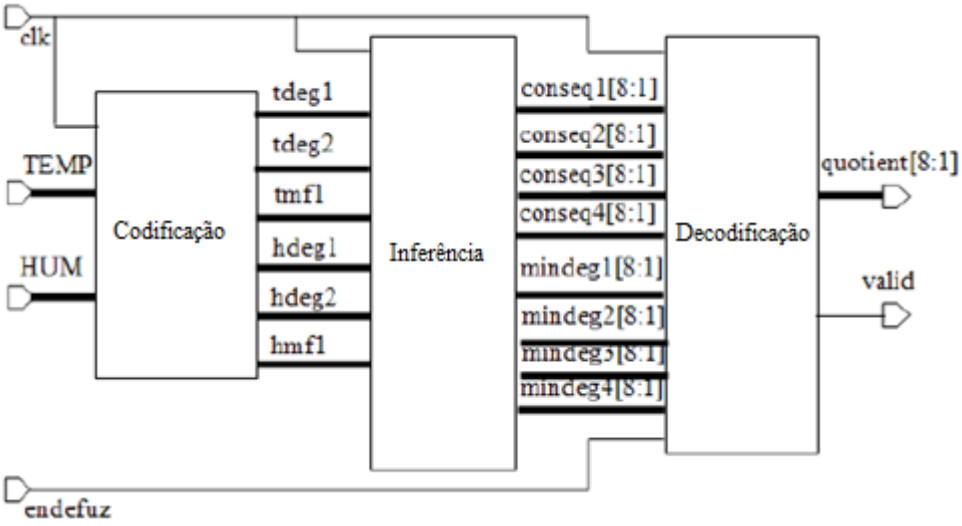
2.3 CONTROLE *FUZZY* EM VHDL

Os trabalhos envolvendo implementação *fuzzy* em VHDL variam desde sistemas solares fotovoltaicos, sistemas de pêndulo amortecido e incubadoras. Contribuindo para o início de novas pesquisas neste tipo de aplicação na Universidade, este trabalho irá ser comparado aos trabalhos relacionados neste tópico com intuito de verificar a variação do erro quadrático percentual, para testar se a implementação para o sistema proposto é adequada. Dentre os trabalhos estudados, os de maior relevância serão destacados a seguir.

Uppalapati e Kaur (2010) fizeram um sistema de inferência *fuzzy* implementado em um FPGA, escrito em VHDL, de um sistema caseiro de irrigação, onde são

monitoradas a umidade do solo e a temperatura, no que resulta no tempo de irrigação como variável de saída, mudando o tempo de irrigação baseado nestes parâmetros de entrada. Neste trabalho, é mostrado cada etapa do processo de criação *fuzzy* e sua respectiva implementação em VHDL. Com os resultados, a comparação entre a saída do FPGA e a saída simulada em Matlab® gerou um erro quadrático médio de 0,8%. A Figura 2 mostra o modelo estrutural do sistema *fuzzy* completo, com as variáveis de pertinência na codificação, número de consequências na inferência e a saída "v*alid"* na etapa de decodificação.

Figura 2 – Modelo de blocos do controlador fuzzy descrito em VHDL



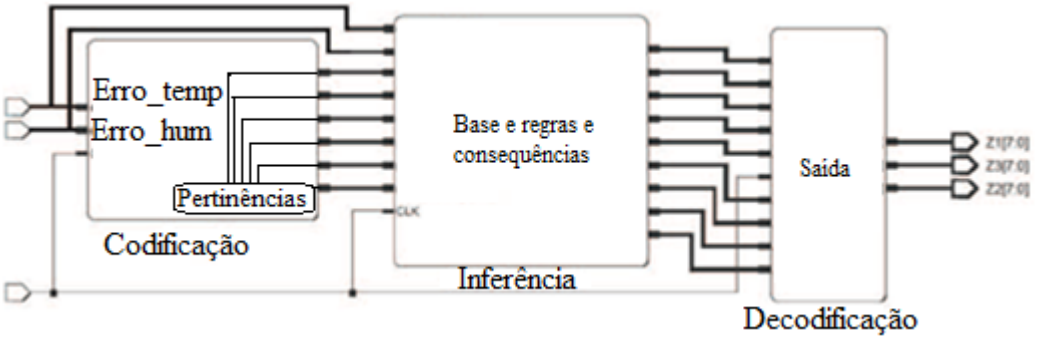
Fonte: UPPALAPATI E KAUR (2010)

Vuong et al (2006) descrevem, de maneira didática, como realizar uma implementação *fuzzy* em VHDL, além de exemplos e de como se construir o sistema *fuzzy* nesta linguagem baseada em um pseudo-código sugerido pelos autores, desde a etapa de codificação, até a decodificação.

Oliveira et al (2010) desenvolveram um sistema de pêndulo amortecido implementado em VHDL baseado no sistema de lógica *fuzzy*. As entradas do sistema são o erro e a variação do erro, e a saída era a tensão no motor para atuar no estado do pêndulo. Resultados simulados e experimentais foram feitos, e comparados os valores respectivos de erro e variação do erro com as saídas do sistema simulado e o sistema implementado em FPGA, e o resultado do erro quadrático médio foi de 0,84%, mostrando sua eficácia quando comparado com demais resultados encontrados na literatura.

No trabalho de Aguilar et al (2014), a implementação *fuzzy* em VHDL se dá em um sistema de incubadora, onde é monitorado duas variáveis, sendo estas a temperatura e a umidade, e a partir do estado destas entradas, a saída pode ter três estados: “aquecer” “refrescar” e “umedecer”. Comparando a implementação das três saídas simuladas com o sistema real, os erros quadráticos médios foram, respectivamente, de 8,02%, 7,79% e 10,46%. Este resultado foi devido à utilização do método de decodificação de Takagi-Sugeno média dos máximos, e quando alterado para o método da média ponderada, o erro quadrático médio total foi reduzido para 0,5%. A Figura 3 mostra o modelo estrutural de cada bloco *fuzzy* e suas variáveis interligadas pelo controlador proposto.

Figura 3 – Controlador *fuzzy* proposto para um sistema de incubação



Fonte: Adaptado de Aguilar et al (2015)

Bouselham et al (2015) elaboraram um trabalho para descrever uma implementação em *hardware* de um controlador lógico *fuzzy* MPPT em um FPGA utilizando a linguagem VHDL, de um sistema fotovoltaico. Os resultados mostraram que o desempenho foi satisfatório em comparação ao método tradicional, que utiliza outro *software* de implementação e simulação, baseado na inserção das variáveis de entrada do sistema, enaltecendo ainda a melhor eficiência do sistema e a flexibilidade de programação.

Abbes et al (2015) utilizaram um sistema *fuzzy* implementado em VHDL para uma técnica chamada MPPT (*Maximum Power Point Tracking)* em um sistema solar fotovoltaico. Os resultados foram comparados com o modelo que utiliza o *hardware* na malha (*Hardware-in-the-Loop)* e o modelo do Simulink foram muito próximos, praticamente com erro nulo, afirmando a importância do VHDL e FPGA em simulações em tempo real.

Os assuntos abordados na revisão da literatura serviram para contextualizar o foco principal do trabalho, onde há controlador *fuzzy* desenvolvido em Matlab®, cuja principal função é atingir regiões não lineares do sistema, aplicado ao controle de nível, que necessita solucionar este problema. Assim, será feita uma implementação deste controle em VHDL que visa melhorar o tempo de processamento e, através do *software* Altera Quartus II, poder criar um circuito digital associado a este, podendo desenvolver a possibilidade de criar um chip para este propósito. As etapas de desenvolvimento deste tipo de implementação se dão no capítulo 3, referente a metodologia.

3. MATERIAIS E MÉTODOS

Este capítulo aborda todo o processo de desenvolvimento do controlador *fuzzy*, e como implementá-lo em linguagem de descrição de *hardware*, começando pela descrição e modelagem do sistema, quais os sensores e atuadores envolvidos no processo, e explicando como a planta trabalha nos ambientes de simulação Matlab®/Simulink e no Modelsim/Altera. O processo de obtenção das variáveis do processo foi baseado no trabalho de Vieira (2017), assim como o controlador *fuzzy* mostrado nos tópicos a seguir.

3.1 DESCRIÇÃO E MODELAGEM DO SISTEMA DE NÍVEL

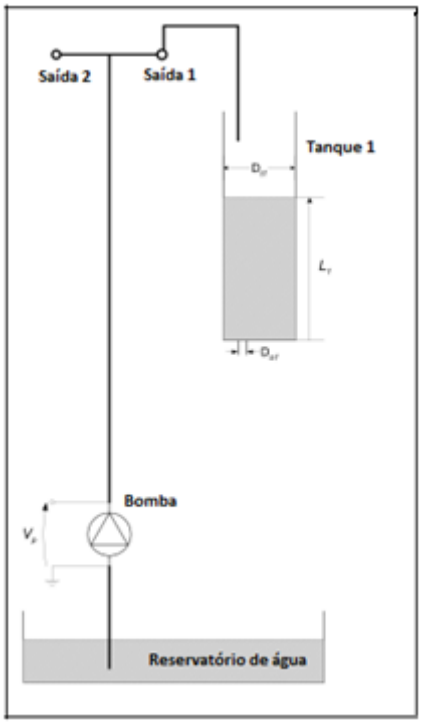
O sistema de nível em estudo consiste em uma bancada Quanser® de dois tanques com seções transversais diferentes, com uma bomba e um reservatório de água na base do sistema, que é autônomo, de forma fechada e com recirculação de água.

Em cada um dos dois tanques, o líquido é retirado do fundo através de um orifício de saída, sendo a pressão de saída, a atmosférica. Ambas as inserções de saída são configuráveis e podem ser ajustadas alterando os orifícios roscados na parte inferior de cada tanque, aumentando ou diminuindo a vazão destes. O nível da água em cada tanque é medido usando um sensor sensível à pressão, localizado na parte inferior do tanque.

3.1.1 Tanque cilíndrico

O sistema feito com o tanque cilíndrico pode ser ilustrado de acordo com a Figura 4**,** onde a bomba fornece água para o tanque através do reservatório de água.

Figura 4 - Configurações dos modelos de tanques



Fonte: Adaptado de Quanser (2017)

O tanque contém um orifício responsável pela vazão de saída do sistema, e dentro deste está inserido o sensor de pressão. Sua modelagem pode ser desenvolvida tomando por base a vazão de entrada (F_{in}), que é dado pela relação entre a tensão da bomba (V_p) e a constante da bomba (K_m). Sua unidade é dada em metro cúbico por segundo. A Equação 1 relaciona a vazão de entrada com as variáveis mencionadas anteriormente:

$$F_{in} = K_m * V_p \left[\frac{cm^3}{s}\right] \quad (1)$$

Têm-se também a equação de saída de água do sistema, regida pela equação de Bernoulli para escoamento da água, que pode ser vista na Equação 2:

$$V_{out} = \sqrt{2gL}\left[\frac{cm}{s}\right] \quad (2)$$

Onde g é a gravidade (cm/s²) e L é a altura do volume de água, neste caso, em centímetros.

Tomando por base as duas equações anteriormente citadas, e relacionando vazão de saída do sistema com a área do orifício de saída, sendo determinado pelo produto de

ambas, a equação que rege a velocidade de escoamento de saída do sistema é determinado pela equação 3:

$$F_{out} = aV_{out} = a\sqrt{2gL}\ \left[\frac{cm^3}{s}\right] \quad (3)$$

Sendo *a* a área do orifício de saída, em cm².

A fim de se obter a variação do nível dos tanques (L'), uma das variáveis de controle do sistema, Vieira (2017) coloca que é necessário definir a variação volumétrica (V'), dada pela diferença entre as vazões de entrada e saída do tanque, e por fim, resulta-se a equação 4 para a variação do nível:

$$L' = \frac{Km}{A} * V_p - \frac{a}{A}\sqrt{2g}\sqrt{L}\ \left[\frac{cm^3}{s}\right] \quad (4)$$

Onde:

L': variação do nível do tanque, em cm;

A: área da seção transversal do tanque, em cm.

Com as considerações de modelagem do tanque de formato cilíndrico da Quanser, será apresentado a modelagem do tranque de modelo proposto, com geometria trapezoidal.

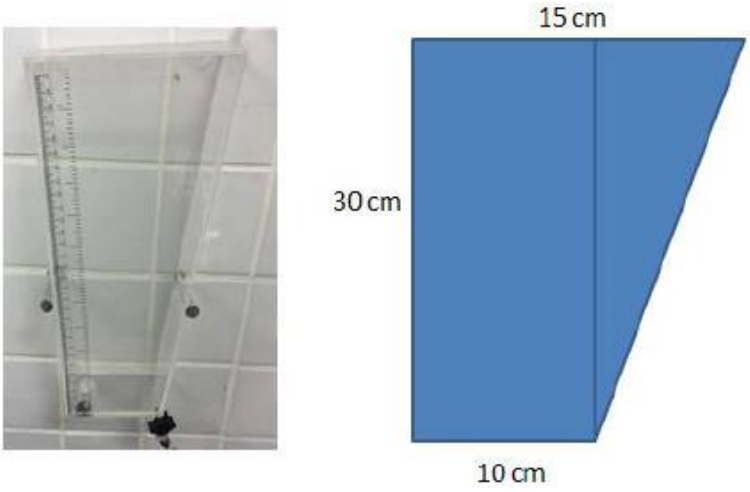
3.1.2 Tanque proposto de formato trapezoidal

No tocante à área da seção transversal, a equação que rege seu comportamento baseado na altura da coluna d'agua pode ser vista na equação 5, onde:

$$A = 50 + 0.833 * Altura \quad (5)$$

Esta equação foi obtida de acordo com as dimensões do tanque, que pode ser visto na Figura 5.

Figura 5 - Formato do tanque proposto



Fonte: VIEIRA (2017)

Dividindo a área total do tanque em duas partes, forma-se um retângulo e um triângulo. A inclinação (coeficiente angular) deste triângulo foi obtida com base no ângulo da tangente dos catetos, e assim, um fator de crescimento foi desenvolvido por Vieira (2017), mostrando que há um fator de crescimento de 0.833 cm² à medida que a altura do tanque se modificava, sendo a área da secção transversal a única variável adicional com relação a modelagem do tanque vista no tópico 3.1.1.

3.2 ATUADOR

Segundo Thomazini e Albuquerque (2007), os atuadores são dispositivos que modificam uma variável controlada. Recebem um sinal proveniente do controlador e agem sobre o sistema controlado. Geralmente trabalham com potência elevada. São exemplos de atuadores:

- Válvulas (pneumáticas, hidráulicas);
- Relés (estáticos, eletromecânicos);
- Cilindros (pneumáticos, hidráulicos);
- Motores (motor de passo, servomotor, em corrente contínua ou alternada);
- Solenoides.

A maioria dos atuadores pode ser classificada em três categorias, segundo o tipo de amplificador utilizado, que pode ser elétrico, hidráulico e pneumático, sendo o atuador

elétrico o mais comum. Os atuadores hidráulicos utilizam fluido hidráulico para amplificar o sinal de comando do controlador; já os atuadores pneumáticos utilizam o ar comprimido como energia propulsora (GROOVER, 2011).

O atuador neste processo é uma bomba de engrenagem composta por um motor de corrente contínua de 15 Volts com aletas que disseminam o calor. Os materiais que entram em contato com o fluido bombeado, no caso a água, são duas engrenagens montadas no corpo da bomba, de aço inoxidável, um diafragma de teflon e um selo de borracha nitrílica. Também está equipado com mangueiras de 3/16’’.

3.3 SENSOR

Os sensores são os componentes mais utilizados no mundo da eletroeletrônica. Eles estão presentes no dia a dia nas mais variadas situações (carros, elevadores, portas automáticas etc.). Estes dispositivos também constituem toda a base da automação, seja ela predial, comercial ou industrial (CAPELLI, 2008).

Groover (2011) define sensor como um transdutor, ou seja, um dispositivo que converte um estímulo físico ou variável de interesse (tal com temperatura, força, pressão ou deslocamento) em outra mais útil para a aplicação em questão (em geral um sinal elétrico de tensão), cujo propósito é medir o estímulo.

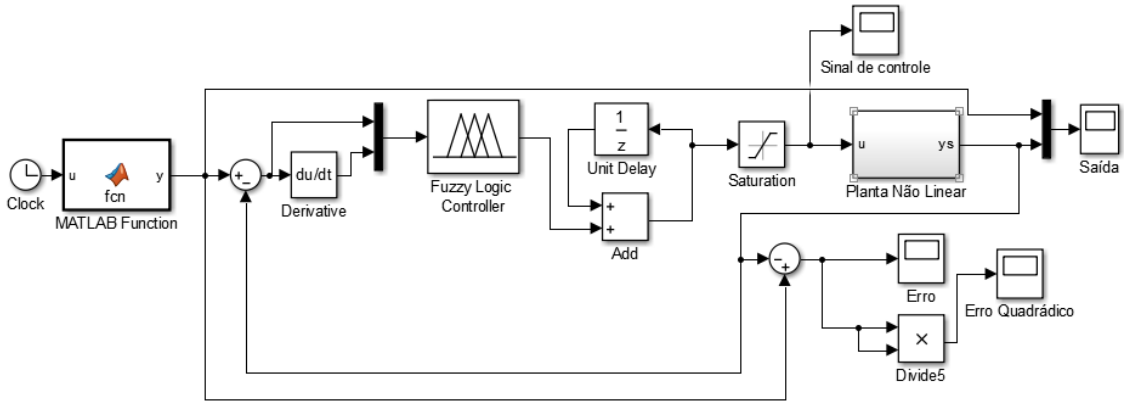
Além do tipo de estímulo, os sensores também são classificados como analógicos ou discretos, conforme as variáveis do processo. Um dispositivo ativo de medição analógico produz um sinal analógico contínuo como uma tensão elétrica, cujo valor varia de modo analógico com a variável sendo medida. Já um dispositivo discreto produz uma saída que pode ter apenas determinados valores. Dispositivos de sensoriamento discretos costumam ser divididos em binários e digitais (GROOVER, 2011).

Em cada tanque, seus respectivos níveis de líquidos atuais são medidos por meio de um sensor de pressão. Os sensores estão localizados no fundo de cada tanque e fornecem níveis de leituras lineares em relação ao nível vertical de líquido completo. Em resumo, a tensão de saída aumenta proporcionalmente com a pressão aplicada. A medição na saída é processada através de uma unidade de condicionamento de sinal e a torna como um sinal contínuo de 0 a 5V.

3.4 PLANTA

O sistema simulado pode ser dividido em duas plantas: uma contendo o tanque padrão da Quanser, e a outra, com o tanque de modelo trapezoidal proposto. Os dois modelos estão desenvolvidos na plataforma Matlab®/Simulink, e estão modelados de acordo com as equações mostradas nos tópicos 3.1.1 e 3.1.2. O simulador do tanque da Quanser implementado por Vieira (2017) contém um bloco *fuzzy* realizando o controle do sistema e o sinal de saída é monitorado comparando com o valor desejado. A Figura 6 mostra o simulador para o tanque da Quanser.

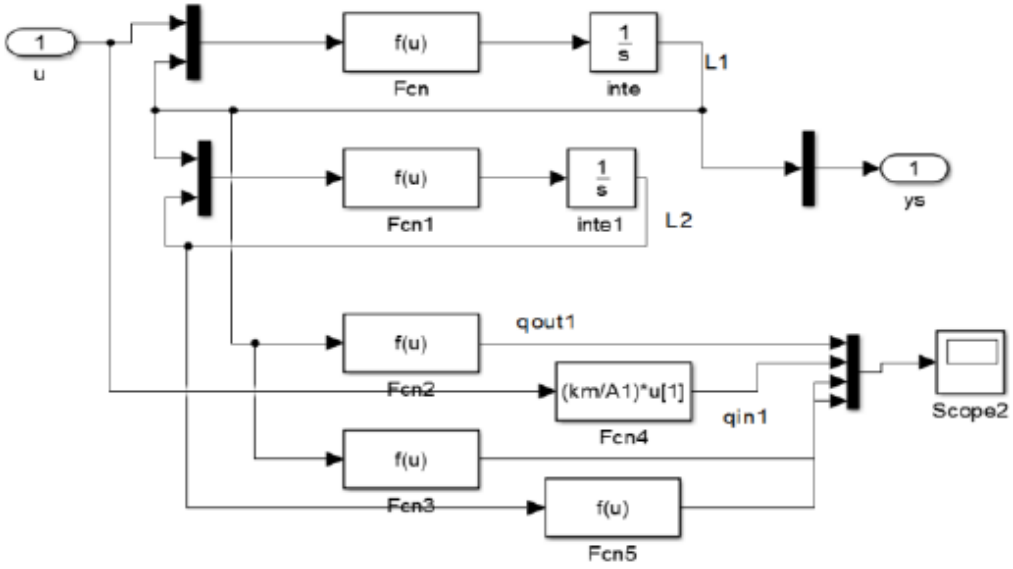
Figura 6 - Simulador tanque da Quanser



Fonte: VIEIRA (2017)

Neste simulador, serão aplicados valores de referência de entrada (*Set-points*) para observar o comportamento do sistema. O bloco “planta não-linear” representa o comportamento do sistema e esta responde à entrada de tensão fornecida pelo controlador *fuzzy*. Seu interior foi expandido, mostrando as variáveis previamente declaradas em um *Script* de programa, dentre elas: altura, gravidade, constante da bomba e área da secção transversal. A figura 7 representa a expansão do bloco “planta não-linear”.

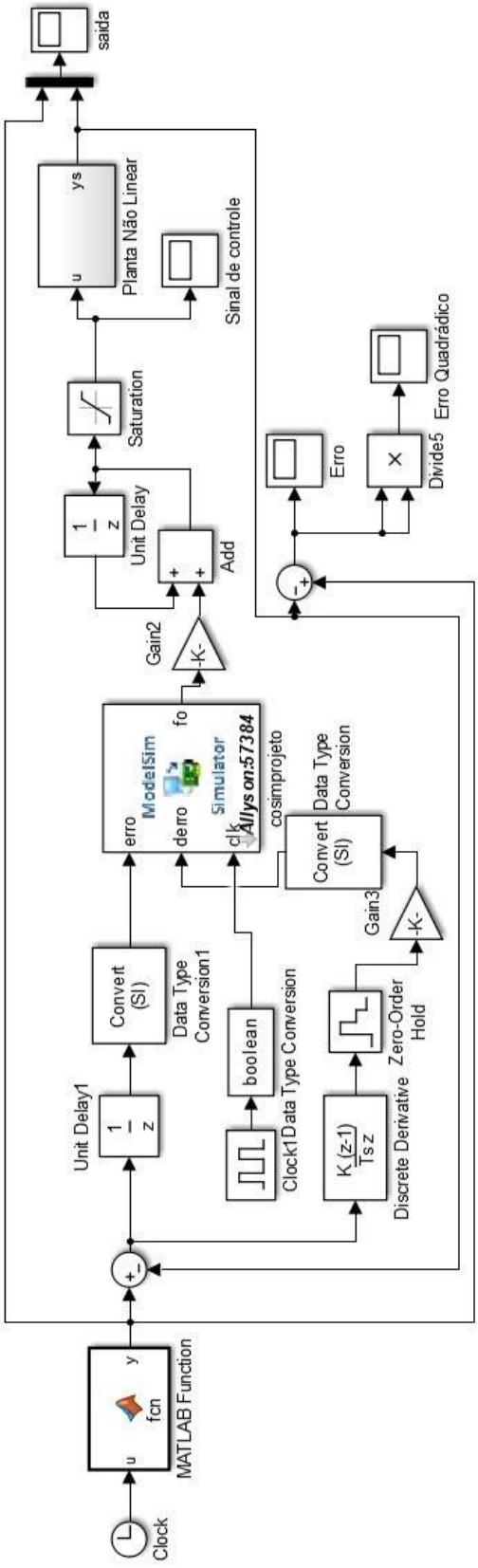
Figura 7 - Bloco “planta não-linear”



Fonte: Adaptado de Viera (2017)

O sistema simulado com tanque da Quanser precisa ser adaptado para que a implementação em VHDL seja possível. Substituindo o bloco *fuzzy* por um bloco de co-simulação Modelsim/Matlab®, é possível que a implementação seja feita, pois é no Modelsim que a descrição *fuzzy* em VHDL é executada, gerando valores para o sistema de nível, enquanto o Matlab®, através da ferramenta Simulink, fornece os parâmetros de modelagem do sistema (modelo matemático dos tanques). A Figura 8 mostra o simulador com tanque da Quanser adaptado para implementação *fuzzy* em Linguagem de Descrição de Hardware.

Figura 8 - Simulador tanque da Quanser adaptado para o ambiente de co-simulação



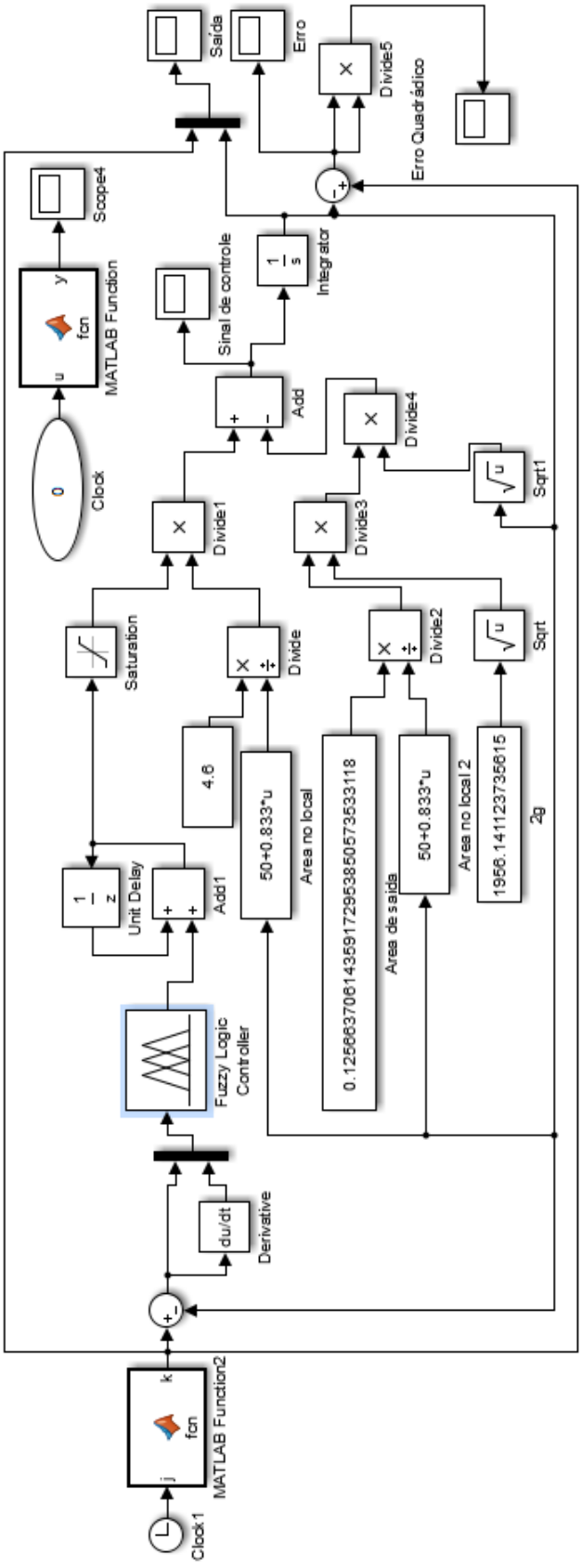
Fonte: Autoria própria (2017)

Como pode-se observar no sistema de implementação *fuzzy*, há a adição de outro tipo de variável: o sinal de *clock* digital, onde a cada pulso positivo deste (ou transição para a borda de subida, como se pode definir), será calculado um novo valor de saída, baseado nas entradas do sistema. Outros blocos auxiliares também foram inclusos, para discretizar o sistema, com a derivada discreta, por exemplo.

Com a modelagem do sistema para o tanque da Quanser e o controlador *fuzzy* Matlab® desenvolvido por Vieira (2017), será comparada a resposta deste sistema com o *fuzzy* implementado em VHDL. Para testar sua eficácia, alguns parâmetros (saída, erro, erro quadrático e tensão na bomba) serão levados em consideração, além de resultados encontrados na literatura previamente consultada.

Com os modelos do tanque da Quanser devidamente apresentados, a simulação do tanque em formato trapezoidal será feita. Primeiramente será levado em consideração que as equações de estado e o controlador serão os mesmos, a única diferença é de conter uma equação na modificação na área da secção transversal à medida que o nível aumenta, diferentemente do sistema anterior, que era constante. O sistema proposto com tanque trapezoidal pode ser visto na Figura 9.

Figura 9 - Sistema proposto com tanque trapezoidal

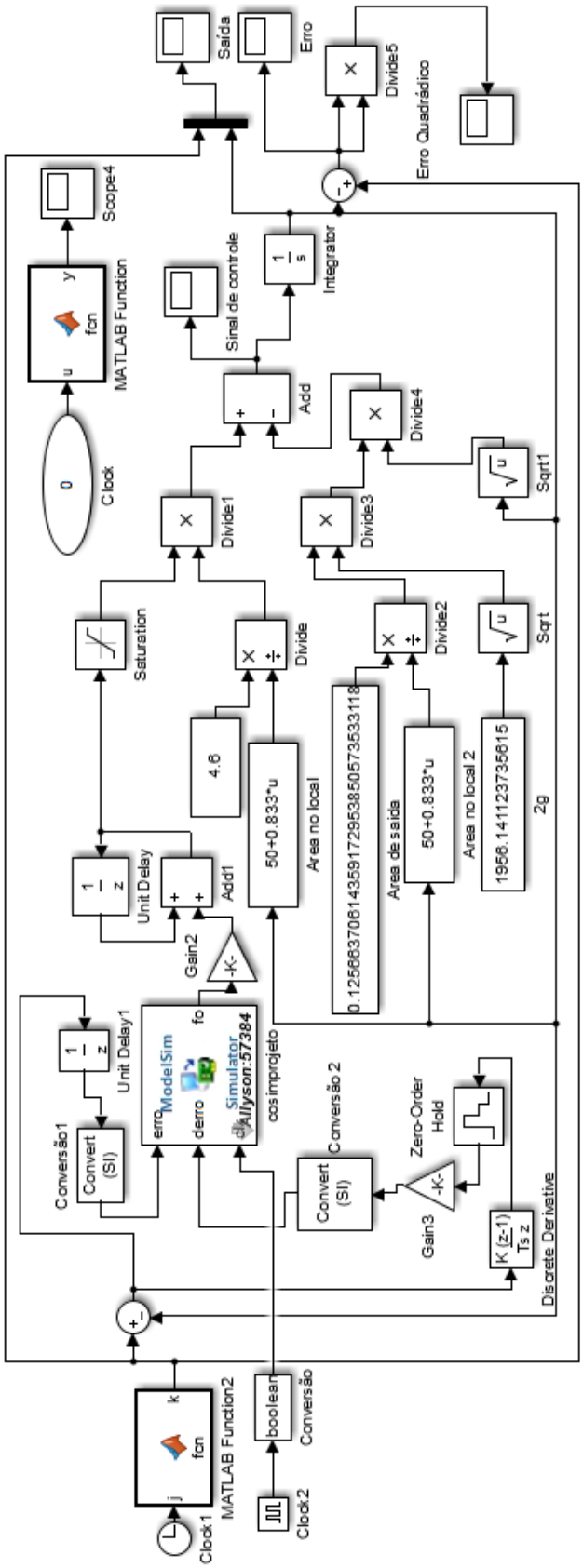


Fonte: Adaptado de Vieira (2017)

Assim como no sistema de tanque da Quanser, em seu início, há um bloco de função Matlab® que rege a alteração do nível de referência ao longo do tempo. Os blocos *Unit Delay* e *add1* tem por finalidade guardar o último valor de tensão que foi aplicado na bomba para que o sistema *fuzzy* incremente ou decremente este valor, baseado na tomada de decisão do controlador. Tem-se também o bloco *Saturation* que limita os valores que serão aplicados na bomba, para fins de proteção. Por fim, os blocos *Area no local* e *Area no local1* são responsáveis pelo regimento da equação que corresponde ao crescimento da área com a secção transversal do tanque.

Em relação ao sistema implementado em VHDL, o controle é o mesmo e as adaptações também, como a inclusão do *clock* e a discretização das entradas no sistema. Ao fim da simulação, será comparado os valores de saída em *fuzzy* com os valores dados pelo ambiente de co-simulação Matlab®/Simulink – Modelsim/Altera e os respectivos erros gerados. Na Figura 10, podem ser vistas as modificações feitas para o ambiente de co-simulação.

Figura 10 - Sistema com tanque trapezoidal para o ambiente de co-simulação



Fonte: Autoria própria (2017)

3.5 PROJETO DO CONTROLADOR

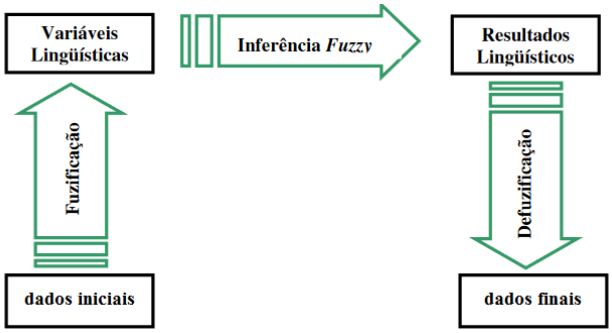
Para este trabalho, foi utilizado um sistema de controle baseado na lógica *fuzzy* - e de acordo com este tipo de controle, foi implementado seu funcionamento baseado em linguagem de descrição de hardware, mais especificamente, em VHDL. A forma de como os controladores são implementados serão mostrados a seguir.

3.5.1 Controlador *fuzzy*

A Teoria de Conjuntos *fuzzy* e os Conceitos de Lógica *fuzzy* são utilizados para traduzir em termos matemáticos essas informações imprecisas e expressá-las em um conjunto de regras linguísticas. Um exemplo dessa tradução é analisar um processo realizado por um ser humano e verificar se o mesmo pode ser descrito por um conjunto de regras formadas por operadores "se...então", dessa forma pode-se construir um algoritmo para ser implementado em um sistema computacional. O resultado desse trabalho é um sistema *fuzzy* baseado em regras, no qual a Teoria de Conjuntos *fuzzy* e lógica *fuzzy* fornecem o ferramental matemático para se lidar com as tais regras linguísticas (CALDEIRA et al., 2007). Dentre as diversas áreas de atuação da lógica *fuzzy*, pode-se destacar a utilização dessa ferramenta para controlar processos industriais.

Segundo LEE (1990), a estrutura básica de um controlador lógico *fuzzy* possui quatro componentes principais: Base de conhecimentos, onde há a entrada dos dados iniciais; codificação (ou fuzificação), onde haverá a conversão dos valores reais em valores *fuzzy*; Inferência, onde será avaliada a tomada de decisões; e decodificação (defuzificação), que é a conversão dos valores *fuzzy* para valores reais. A Figura 11 mostra o fluxograma dos componentes.

Figura 11 - Componentes de um controlador *Fuzzy*



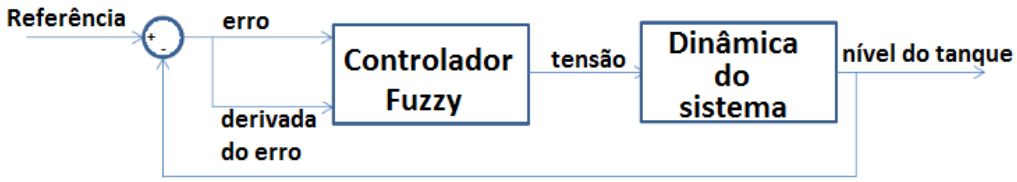
Fonte: Adaptado de Almeida (2004)

Na codificação, são feitas as análises de variáveis de entrada e saída do projeto, escolha do universo de discurso de cada variável e a escolha das formas das funções de pertinência, além de nomeá-las e obter as formas das funções.

No processo de inferência, é introduzida a base de regras, que envolve a base de conhecimento e a lógica de tomada de decisões baseadas em condições propostas pelo especialista/projetista. Já na última etapa, a da decodificação, consiste na aplicação dos valores obtidos na etapa de inferência e transformá-los em valores reais, aplicando métodos matemáticos.

No controlador *fuzzy* deste trabalho, o sistema é composto por duas variáveis de entrada, sendo estas o *erro*, representado pela diferença do valor de referência (*set-point)* e o valor lido; e a *derivada do erro*, que representa a variação entre o estado do nível em relação a leitura anterior. A saída do controle é dada por um valor de incremento de tensão, seja positiva ou negativa, influenciando diretamente o nível do tanque. O esquema do controle *fuzzy* está representado na Figura 12.

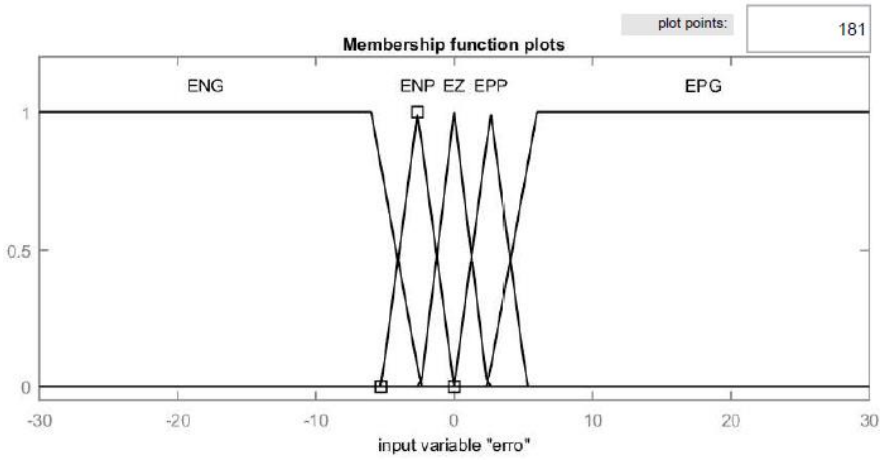
Figura 12 - Esquema do controlador *fuzzy* em malha fechada**.**



Fonte: Adaptado de Vieira (2017)

A variável de entrada *erro* consiste em cinco funções de pertinência: Erro Negativo Grande (ENG), Erro Negativo Pequeno (ENP), Erro Zero (EZ), Erro Positivo Pequeno (EPP) e Erro Positivo Grande (EPG). Seu universo de amostragem consiste em um intervalo de -30 a 30, que pode representar o tanque totalmente cheio ou totalmente vazio. Valores intermediários estão diretamente ligados ao estado do nível do tanque. A Figura 13 representa as funções de pertinência com seus respectivos intervalos.

Figura 13 - Funções de pertinência para a variável de entrada *erro*



Fonte: adaptado de Vieira (2017)

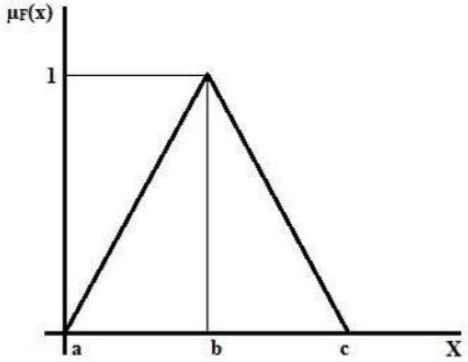
Como pode ser visto nas Figuras 13, 16 e 18, as funções de pertinência usadas neste trabalho são do tipo triangular e trapezoidal.

A função de pertinência triangular é dada pela Equação 6:

$$\mu_F(x) = \begin{cases} 0, & x < a \\ \dfrac{x-a}{b-a}, & a \leq x < b \\ \dfrac{c-x}{c-b}, & b \leq x \leq c \\ 0, & x > c \end{cases} \tag{6}$$

A forma desta função e os pontos *a*, *b*, *c*, e *x* podem ser observados na Figura 14.

Figura 14 - Função de pertinência triangular



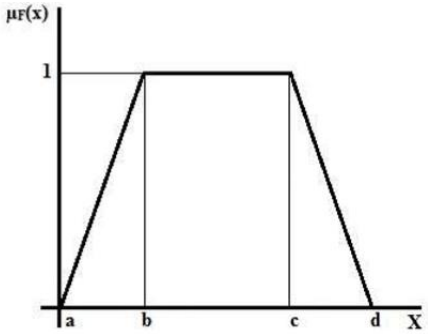
Fonte: (TEIXEIRA JÚNIOR, 2014).

A função de pertinência trapezoidal é dada pela equação 7:

$$\mu_F(x) = \begin{cases} 0, & x < a; \\ \dfrac{x-a}{b-a}, & a \leq x < b; \\ 1, & b \leq x < c; \\ \dfrac{d-x}{d-c}, & c \leq x \leq d; \\ 0, & x > d \end{cases} \quad (7)$$

A forma da função e seus respectivos pontos representados na equação podem ser vistos na Figura 15.

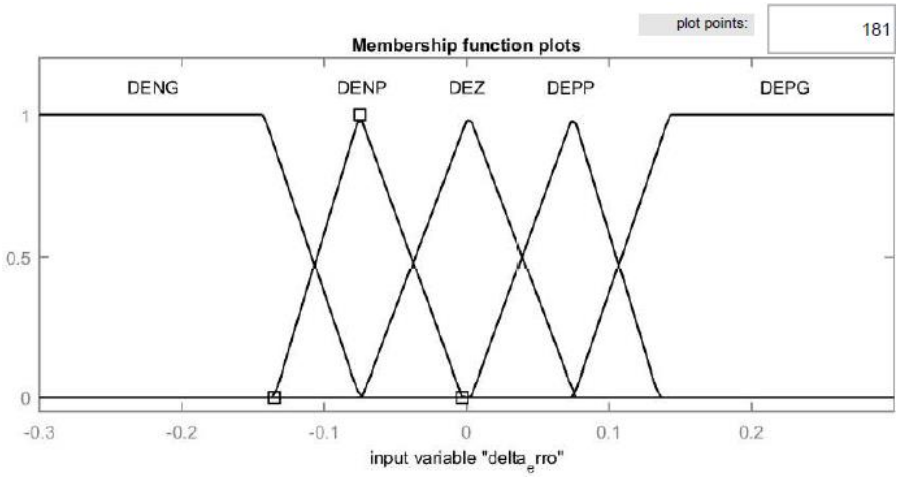
Figura 15 - Função de pertinência trapezoidal



Fonte: (TEIXEIRA JÚNIOR, 2014).

A outra variável de entrada do controlador, *derivada do erro*, também é composta por cinco funções de pertinência: Derivada do Erro Positivo Grande e Pequeno (DEPG E DEPP, respectivamente) Derivada do Erro Zero (DEZ) e Derivada do Erro Negativo Grande e Pequeno (DENG e DENP, respectivamente). Com relação ao seu universo de amostragem, ele varia com o intervalo de -0,3 a 0,3, onde estes valores serão apenas para demonstrar se há uma tendência de crescimento ou decrescimento do sistema. A Figura 16 representa as funções de pertinência da variável derivada do erro com seus respectivos intervalos.

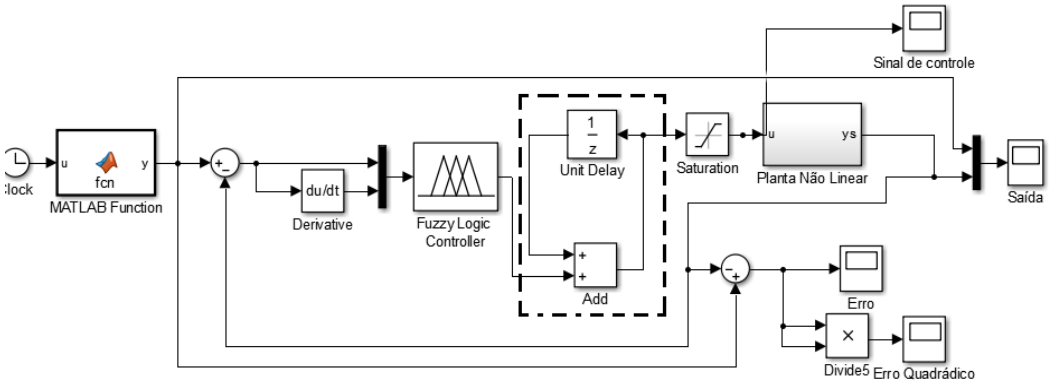
Figura 16 - Funções de pertinência para a variável derivada do erro



Fonte: adaptado de Vieira (2017)

A forma de como o controle se comporta está baseada no acréscimo ou decréscimo do valor atual de tensão que está sendo aplicado na bomba, ou seja, a saída será apenas um incremento, sendo este valor positivo ou negativo, tornando este tipo de controle inercial. O bloco responsável pelo incremento de tensão está em destaque na Figura 17.

Figura 17 – Saída do controlador *fuzzy* em estudo no Matlab®



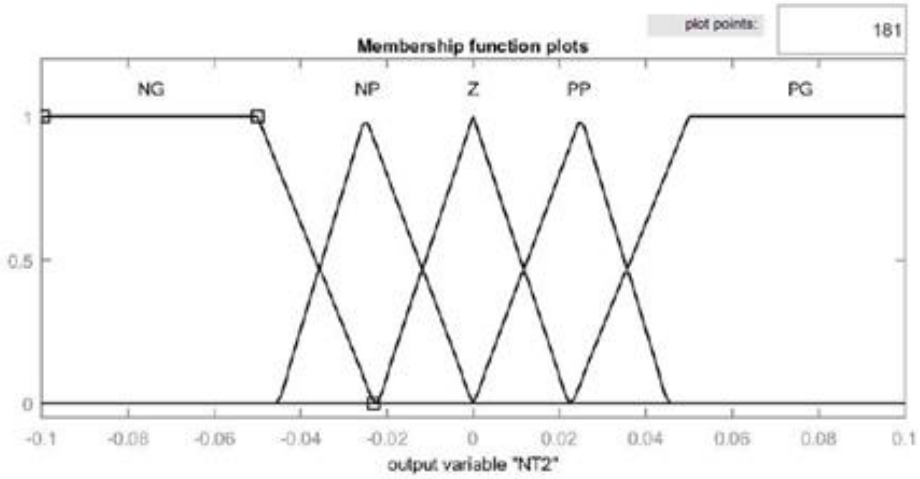
Fonte: Adaptado de Vieira (2017)

Após a saída do Controlador Lógico *fuzzy* da figura 17, seu valor é enviado para o atuador da Planta não linear junto do valor armazenado no bloco de *delay*, que devido ao atraso de tempo, incrementou ou decrementou o valor de tensão proveniente da atualização anterior do valor de saída do controlador.

O universo de amostragem da saída é de [-0.1 a 0.1] e seus conjuntos de funções de pertinência são: Negativo Grande (NG), Negativo Pequeno (NP), Zero (Z), Positivo

Pequeno (PP) e Positivo Grande (PG), totalizando cinco funções. Este intervalo de amostragem foi definido com testes práticos no sistema, onde com o aumento da amplitude, haveria oscilação, o que poderia levar a uma instabilidade do sistema, e com a diminuição muito drástica, poderia deixar o controlador lento demais, demorando no processo de estabilização do sistema em geral. A Figura 18 representa as funções de pertinência da saída do controlador *fuzzy* Matlab®

Figura 18 - Funções de pertinência da saída do controlador



Fonte: Adaptado de Vieira (2017)

A base de regras é formada por estruturas do tipo:

SE <condição/premissa> **ENTÃO <** conclusão/decisão>

Estas regras são processadas pelo mecanismo de inferência, o qual infere as ações de controle de acordo com o estado do sistema, aplicando o operador de implicação, conforme o procedimento de inferência e seus variados métodos. Em um dado controlador *fuzzy*, é importante que existam tantas regras quantas forem necessárias para mapear totalmente as combinações dos termos das variáveis, ou seja, que a base seja completa, garantindo que exista sempre ao menos uma regra a ser disparada para qualquer entrada (CONFESSOR, 2014).

As premissas são relacionadas pelos conectivos lógicos, dados pelo operador de conjunção (*e*) e o operador de disjunção (*ou*). Em geral, as regras têm a forma de um sistema de múltiplas entradas e saídas, onde diversas combinações de entrada geram diversas combinações de saída, mas que pode ser transformado em vários sistemas com

múltiplas entradas e uma saída, onde poucas saídas determinadas podem ser obtidas através das diversas combinações das entradas do sistema (SANDRI & CORREA, 1999).

Com a base de regras formada, é possível processar estas regras representadas em funções de pertinência para a aplicação dos métodos de inferência, e através destes métodos, classificar a saída de acordo com funções e regras previamente projetadas pelo especialista.

As regras para a tomada de decisão do controlador dependem do estado das variáveis de entrada (*erro* e *derivada do erro*), então a saída do sistema será de acordo com as uniões e interseções dos valores provenientes destas variáveis de entrada. A Tabela 1 mostra o conjunto de regras para a tomada de decisão do controle, onde a primeira coluna desta tabela representa a *derivada do erro*, e a primeira linha representa o *erro*. O cruzamento da respectiva linha com a coluna das variáveis de entradas resulta na variável de saída correspondente.

Tabela 1 – Regras do conjunto *fuzzy*

	ENG	ENP	EZ	EPP	EPG
DENG	NG	NG	NG	PP	PP
DENP	NG	NP	NP	PP	PP
DEZ	NG	NG	Z	PP	PP
DEPP	PP	Z	PP	Z	PP
DEPG	Z	PG	PP	Z	PP

Fonte: adaptado de Vieira (2017)

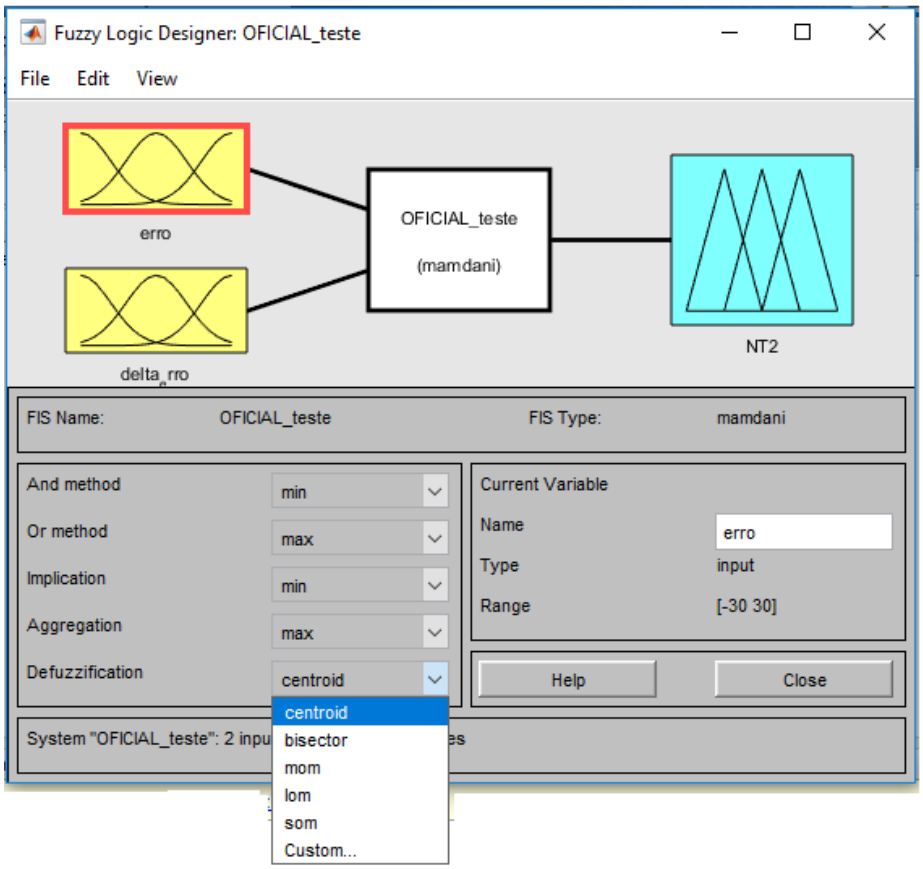
Este tipo de configuração foi feito para o controlador *fuzzy* ganhar adaptabilidade ao sistema no qual está sendo utilizado, podendo alterar o sistema em que está situado e o mesmo continuará com a mesma eficácia.

Concluídas as etapas de codificação e inferência, a última parte do processo é chamada de decodificação, onde o valor da variável linguística de saída inferida pelas regras *fuzzy* será traduzido num valor discreto, com a finalidade de se obter um único valor que melhor represente os valores *fuzzy* inferidos da variável linguística de saída, ou seja, a distribuição de possibilidades. Simões e Shaw (2007) definem que o processo de

decodificação é uma transformação inversa que traduz a saída do domínio *fuzzy* para o domínio discreto.

Enquanto são configurados os parâmetros *fuzzy*, o sistema permite a escolha de qual método de decodificação usar. O método de escolha pode ser visto em destaque na Figura 19.

Figura 19 – Escolha do método de decodificação no Matlab®



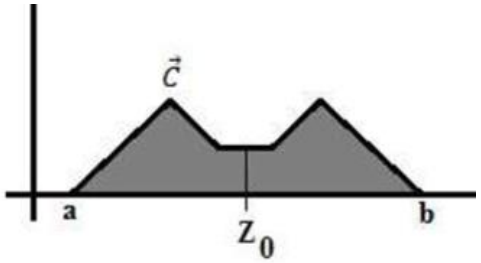
Fonte: adaptado de Vieira (2017)

Dentre os tipos de métodos de decodificação mostrados na Figura 19, dois deles são destacados: *Centroid (*método de decodificação Centro de Área), e *mom* (do termo *Middle of maximum*, que representa o método da decodificação Média dos Máximos). Simões e Shaw (2007) classificam estes dois métodos de decodificação citados e o método de Centro do Máximo como principais métodos de decodificação, cada um com suas características e equações particulares. Uma breve explanação sobre eles será feita a seguir:

- **Decodificação centro de área**

Este método, utilizado neste trabalho, é conhecido na literatura por centro de gravidade ou chamado apenas de centróide. Seu princípio se baseia na computação do centro da área da região sob uma curva definido por um conjunto *fuzzy* de acordo com seu intervalo, conforme pode ser observado na Figura 20.

Figura 20 - Decodificação pelo método do centro de área



Fonte: TEIXEIRA JÚNIOR (2014)

O centróide é um ponto que divide a área do conjunto C em duas partes, mostrando a contribuição das funções de pertinências e regras envolvidas no processo. Caso o intervalo de domínio de C seja finito, a Equação 8 é dada para valores discretos:

$$z_0 = \frac{\sum_{j=1}^{n} z_j C(z_j)}{\sum_{j=1}^{n} C z_j} \tag{8}$$

Onde:

z_0: saída;

z_j: valor de abcissa que varia até *n;*

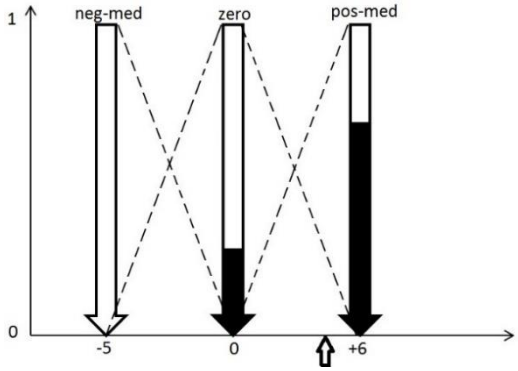
$C(z_j)$: valor da ordenada associada à abcissa z_j.

Segundo Simões e Shaw (2007), este método apresenta problemas quando as funções de pertinência não possuem sobreposição, onde o centro geométrico da figura na realidade não deveria ter um significado físico. Outro fator é que se mais de uma regra tiver a mesma saída *fuzzy*, há uma sobreposição de áreas que não é devidamente contabilizada, além disso, a necessidade de integração numérica toma esforço computacional para cálculo.

- **Decodificação centro do máximo**

Neste método, os picos das funções de pertinência representados no universo de discurso da variável de saída são usados, enquanto ignoram-se as áreas das funções de pertinência; as contribuições múltiplas de regras são consideradas por este método (SIMÕES & SHAW, 2007). O autor ainda comenta que o valor de saída é determinado achando-se o ponto de apoio onde ficam os pesos, que são representados pelo tamanho do comprimento do vetor associado ao valor de pertinência das regras associadas ao valor da saída. Esta representação pode ser vista na Figura 21.

Figura 21 - Decodificação pelo método centro do máximo



Fonte: Autoria própria (2016)

De acordo com a Figura 21, a saída discreta é calculada como uma média ponderada dos máximos (indicado pela seta no eixo horizontal), cujos pesos são os resultados da inferência. O Cálculo do valor decodificado é realizado através da Equação 9:

$$X_{CM} = \frac{\sum_{i=1}^{M} X_{máx}(i)\, \mu[X_{máx}(i)]}{\sum_{i=1}^{M} \mu[\, X_{máx}(i)]} \tag{9}$$

Onde:

X_{CM}: valor do centro do máximo;

$X_{máx}(i)$: valor de abcissa onde se encontra o ponto máximo da função;

$\mu[X_{máx}(i)]$: pontos em que ocorrem os máximos das funções de pertinência de saída.

A diferença entre os dois métodos anteriores, segundo Simões e Shaw (2007), é que enquanto o método do centroide usa a área de cada função de pertinência, o centro

dos máximos utiliza apenas seus valores de pico, e naturalmente, os resultados utilizando estes métodos serão ligeiramente diferentes.

- **Decodificação média dos máximos**

Apesar de apresentar um baixo custo computacional, este método não é muito implementado em computação para fins de controle, pelo fato de resultar valores discretos descontínuos (SIMÕES & SHAW, 2001, apud Vale, 2008). A Equação 10 mostra o método de Decodificação utilizando a média dos máximos.

$$\mu * = \sum_{m=1}^{M} \frac{\mu_m}{M} \tag{10}$$

Onde:

μ_m: m-ésimo elemento do universo de discurso

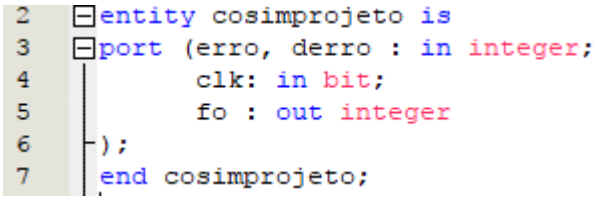
M: número total de elementos.

3.5.2 Implementação do controlador *fuzzy* em VHDL

Nesta seção, será mostrado como implementar o sistema *fuzzy* em questão em linguagem de descrição de hardware. O sistema foi dividido em três etapas: codificação, inferência e decodificação. Em cada etapa, será mostrado parte do programa para fins de esclarecimento da programação.

Na etapa de codificação, as variáveis erro e derivada do erro são convertidas em valores *fuzzy*. A declaração desta etapa em entidade em VHDL deve ser declarada – além de um sinal de *clock* – para fins de simplificação com valores inteiros, pois o fator conversão para um sinal digital é mais fácil, e uma possível implementação em um FPGA de ponto flutuante (*floating point)* pode ser inviável por causa do seu valor aquisitivo de mercado e sua complexidade de obtenção. A Figura 22 mostra a declaração das variáveis de entrada como entidade de maior nível (*Top-Level Entity*).

Figura 22 – Declaração das variáveis de entrada do sistema descrito em VHDL



```
2   entity cosimprojeto is
3   port (erro, derro : in integer;
4         clk: in bit;
5         fo : out integer
6   );
7   end cosimprojeto;
```

Fonte: Autoria própria (2017)

As funções de pertinência são escritas como equações de reta, obedecendo aos intervalos de cada função, ou aproximando ao valor inteiro superior mais próximo, caso o valor seja decimal. Para este trabalho, foi adotado um ganho no valor da derivada do erro e da saída de 100 e de 1000, respectivamente, para fins de se obter um valor inteiro desejado, além da saída ser de um ganho percentual, variando de 0 a 100%. Uma parte do código em VHDL que representa a equação da função de pertinência erro zero (EZ) pode ser encontrada na Figura 23.

Figura 23 - Equação da reta da função de pertinência Erro Zero (EZ)

```
--EZ erro zero
 ez <= ((42*erro) + 100)  when (erro >-2 and erro < 0) else
       ((-42*erro )+ 100) when (erro >0 and erro <2)
        else 0;
```

Fonte: Autoria própria (2017)

O mecanismo de inferência usado é o método de min-max de Mamdani para se obter a saída *fuzzy*. Antes de avaliar as regras, deve-se criar duas funções que retornem o valor mínimo e o valor máximo entre duas variáveis. O código descrito em VHDL pode ser visto na Figura 24.

Figura 24 – Funções de máximo e mínimo descritas em VHDL

```
10   function minimum(a, b: integer) return integer is
11   variable min: integer;begin
12   if    a < b   then min :=a;
13   else   min :=b;
14   end if;
15   return min;
16   end minimum;
17         |
18   function maximum(a, b, c, d, e, f, g, h, i, j, k: integer) return integer is
19   variable max: integer;
20   begin
21   max := a;
22
23   if b > max then max :=b; end if;
24   if c > max then max :=c; end if;
25   if d > max then max :=d; end if;
26   if e > max then max :=e; end if;
27   if f > max then max :=f; end if;
28   if g > max then max :=g; end if;
29   if h > max then max :=h; end if;
30   if i > max then max :=i; end if;
31   if j > max then max :=j; end if;
32   if k > max then max :=k; end if;
33   return max;
34   end maximum;
35   begin
```

Fonte: Autoria própria (2017)

A função *Minimum* só utilizou duas variáveis para comparação, enquanto que a função de *Maximum* foram utilizadas onze declarações, isto foi designado baseado nos conjuntos de regras em que determinada variável se repetia. Cada conjunto de regra corresponde a um operador de mínimo entre as condições, por exemplo, na regra:

*"Se erro for **EZ** e derro for **DEZ** Então Fo é **Z**"*

Esta regra pode ser representada por C= min[EZ, DEZ]. Todas as regras estão agrupadas pelo operador de máximo (D_n) para se obter as classes do sistema *fuzzy*, então os conjuntos D1, D2, D3, D4, D5 estão representadas pelas saídas NG, NP, Z, PP, PG, respectivamente, onde cada conjunto D está associado ao máximo valor entre os mínimos que envolvem as regras *fuzzy* daquela saída. A Figura 25 representa o agrupamento de regras representada pelo Conjunto D.

Figura 25 – Mecanismo de inferência *fuzzy* descrito em VHDL

```
37      -- d1 = NG; d2 = NP, d3 = Z, d4=PP , d5=PG
38
39  d1 <= maximum (minimum(eng,deng), minimum(enp,deng), minimum(ez,deng),minimum(eng,denp),
40                 minimum(eng,dez),minimum(enp,dez), minimum(enp,dez), minimum(enp,dez),
41                 minimum(enp,dez), minimum(enp,dez),minimum(enp,dez));
42
43  d2 <= maximum (minimum(enp,denp), minimum(ez,denp),minimum(ez,denp),minimum(ez,denp),
44                 minimum(ez,denp),minimum(ez,denp),minimum(ez,denp), minimum(ez,denp),
45                 minimum(ez,denp),minimum(ez,denp),minimum(ez,denp));
46
47  d3 <= maximum (minimum(eng,depg),minimum(enp,depp),minimum(ez,dez), minimum(epp,depp),
48                 minimum(epp,depg),minimum(epp,depg), minimum(epp,depg), minimum(epp,depg)
49                 minimum(epp,depg),minimum(epp,depg),minimum(epp,depg));
50
51  d4 <= maximum (minimum(epp,deng),minimum(epg,deng),minimum(epp,denp),minimum(epg,denp),
52                 minimum(epp,dez),minimum(epg,dez),minimum(eng,depp),minimum(ez,depp),
53                 minimum(epg,depp),minimum(ez,depg),minimum(epg,depg));
54
55  d5 <= minimum  (enp,depg);
```

Fonte: Autoria própria (2017)

Já no estágio de decodificação, é feita a conversão dos valores *fuzzy* em valores reais, através de métodos matemáticos. No caso deste trabalho, foi usado o método do centro de gravidade, e foi comparado com o trabalho de Vieira (2017) que também utilizou este método. O código foi gerado para que a cada evento de transição de borda de subida de *clock*, fosse calculado um novo valor de tensão para a bomba. Na Figura 26, é mostrado o processo de decodificação discretizado em VHDL.

Figura 26 – Método de decodificação descrito em VHDL

```
 1  entity cosimdefuzif is
 2  port (d1, d2, d3, d4, d5: in integer;
 3            clk: in bit;
 4            fo : out integer);
 5  end cosimdefuzif;
 6
 7  architecture xxx of cosimdefuzif is
 8  begin
 9  process (clk, d1, d2, d3, d4, d5)
10  variable prod,som,saida: integer;
11  begin
12  som:=(d1+d2+d3+d4+d5);
13  if (clk'event and clk = '1') then
14  prod:=(-50*d1)+(-25*d2)+(0*d3)+(25*d4)+(50*d5);
15     if som=0 then
16     saida:=0;
17     else
18     saida:=(prod/som);
19     end if;
20  end if;
21  fo<=saida;
22  end process;
23  end xxx;
```

Fonte: Autoria própria (2017)

Os blocos funcionais que foram modelados separadamente (codificação, inferência e decodificação) serão unidos através da declaração de componentes. A descrição em VHDL da declaração de componentes pode ser vista na Figura 27.

Figura 27 – Declaração de componentes descritas em VHDL

```
11  component cosim_fuzif is
12  port (erro, derro : in integer;
13        clk: in bit;
14        signal eng,enp,ez,epp,epg,deng,denp,dez,depp,depg: out integer);
15        end component;
16
17  component cosiminf is
18  port (signal eng,enp,ez,epp,epg,deng,denp,dez,depp,depg: in integer ;
19        clk: in bit;
20        signal d1, d2, d3, d4, d5 : out integer );
21  end component;
22
23  component cosimdefuzif is
24  port (signal d1, d2, d3, d4, d5: in integer;
25            clk: in bit;
26            fo : out integer);
27  end component;
```

Fonte: Autoria própria (2017)

Com os componentes devidamente declarados, é necessário a utilização de um comando chamado *port map*, ligando os sinais de entrada e saída de cada bloco de etapa *fuzzy* e mostrando em qual dos blocos o *clock* atuará. Na Figura 28, pode-se ver o uso do comando *port map* descrito em VHDL.

Figura 28 – Uso do comando *portmap* descrito em VHDL

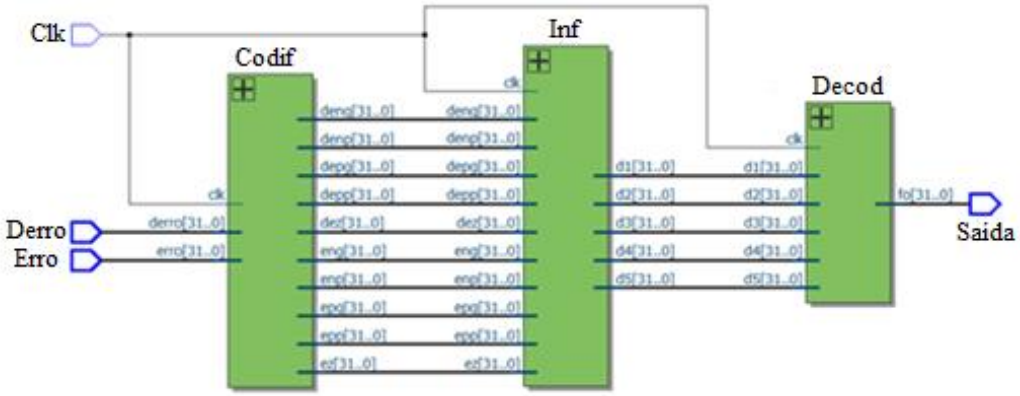
```
32  begin
33
34  Codif: cosim_fuzif
35  port map (erro, derro,clk,eng,enp,ez,epp,epg,
36            deng,denp,dez,depp,depg);
37
38  Inf: cosiminf
39  port map (eng,enp,ez,epp,epg,
40            deng,denp,dez,depp,depg,clk,
41            d1, d2, d3, d4 , d5);
42
43  Decod: cosimdefuzif
44  port map (d1,d2,d3,d4,d5,clk,fo);
```

Fonte: Autoria própria (2017)

Após o uso do *port map*, o Altera Quartus é usado para síntese e implementação do projeto em nível RTL (*Register Transfer level*), empregando primitivas disponíveis na ferramenta de síntese como comparadores, somadores, registradores e portas lógicas. O

circuito gerado nessa etapa não está associado a nenhuma tecnologia de fabricação em particular, e não está, necessariamente, otimizado (D'amore, 2005). O esquema do diagrama de blocos gerado pelo RTL *viewer* é mostrado na Figura 29.

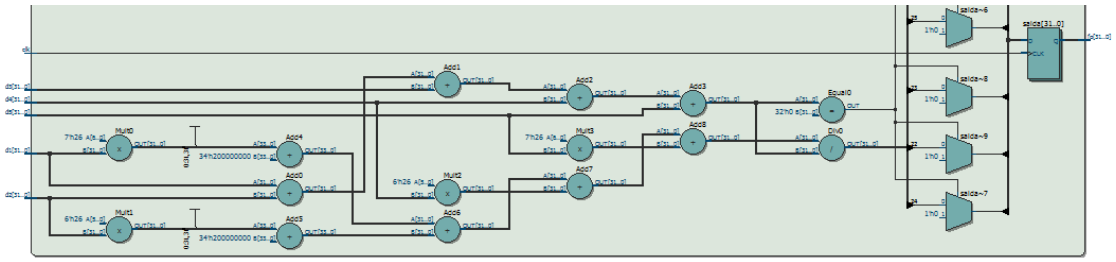
Figura 29 - Diagrama de blocos gerado pelo RTL *Viewer*



Fonte: Autoria própria (2017)

Cada bloco da Figura 29 pode ser expandido, a fim de averiguar o que há dentro de cada bloco, que contém desde somadores, Flip-flops e até ULA's (unidades lógicas aritméticas). Na Figura 30, é mostrada uma parte do bloco de decodificação em nível RTL.

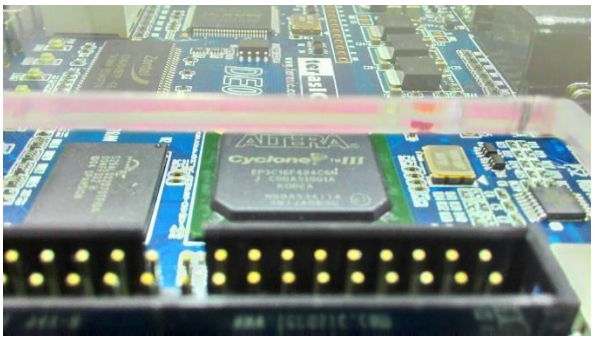
Figura 30 - Parte do bloco de decodificação



Fonte: Autoria própria (2017)

Com o código escrito, verificado e compilado, para fins de comparação de uso de memória, foi utilizado um *kit* de desenvolvimento educacional DE0 ALTERA da família Cyclone III EP3C16F484C6. Por se tratar de um módulo didático, seus recursos são limitados (acender lâmpadas LED ao pressionar um determinado botão etc.), e não aceitam entradas do tipo inteira (*integer)* inviabilizando uma possível utilização embarcada. Este modelo pode ser visto na Figura 31.

Figura 31 - FPGA ALTERA Cyclone III EP3C16F484C6



Fonte: Autoria própria (2017)

Com o FPGA ALTERA Cyclone III EP3C16F484C6 usado como parâmetro de memória, o *Software* ALTERA Quartus II gera uma tabela de recursos utilizados pelo FPGA escolhido. A tabela com os recursos utilizados para o FPGA DE0 Cyclone III pode ser vista na Tabela 2.

Tabela 2 – Recursos de memória para o projeto no FPGA

Recurso	Elementos lógicos requeridos
Elementos lógicos totais	7.707/15.408 (50%)
Registradores lógicos dedicados	32/15.408 (<1%)
Pinos totais	97/347 (28%)
Multiplicadores embutidos	24/112 (21%)

Fonte: Autoria própria (2017)

Assim, pode-se observar que toda a programação descrita em *hardware* para um controlador *fuzzy* feito em Matlab® pode ser embarcado em um FPGA de família similar ao usado como parâmetro.

Com o término deste capítulo, serão feitas as análises dos resultados envolvendo os controladores e seus parâmetros, como erro, erro quadrático, e seus respectivos sinais de controle.

4. RESULTADOS E DISCUSSÃO

Neste capítulo, serão mostrados e discutidos os resultados referentes às simulações no tanque de formato cilíndrico da Quanser e no tanque de modelo proposto, de formato trapezoidal.

A análise se dá por meio da utilização de um conjunto de valores de referência previamente ajustados e verificar a resposta de como o sistema de controle atua. De início, serão mostrados os resultados referentes a uma referência fixa de 25 centímetros para o modelo de tanque cilíndrico, e, posteriormente, será mudado a referência ao longo do tempo, para avaliar a adaptabilidade dos controles (*fuzzy* Matlab® e o *fuzzy* em VHDL).

Os parâmetros analisados são: a resposta do sistema, o erro, erro quadrático e o sinal de controle. Feitas as primeiras discussões, o presente capítulo segue para a avaliação do sistema com o tanque proposto, de formato trapezoidal, sendo as respostas dos controladores comparadas entre si, com o intuito de verificar a eficácia da implementação *fuzzy* em VHDL para este caso, medindo, assim, a precisão do mesmo.

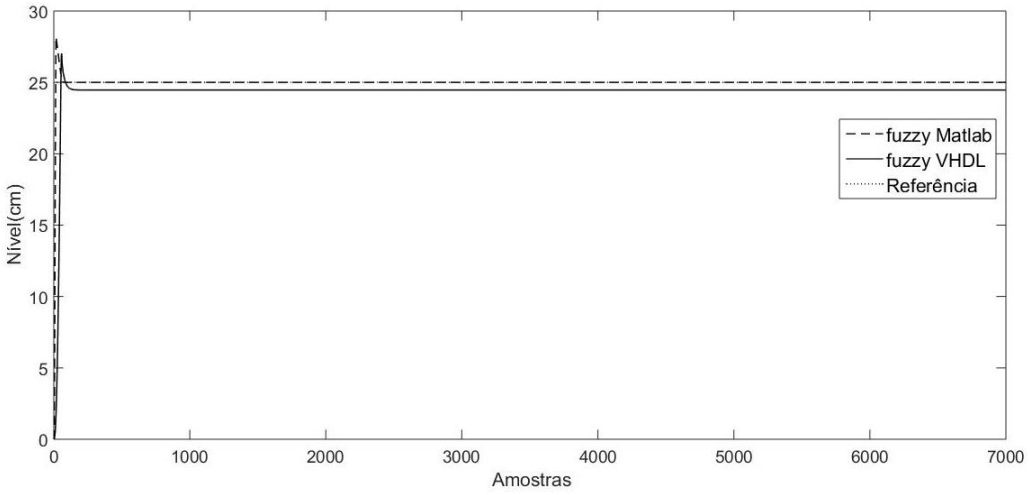
4.1 RESULTADOS REFERENTES AO TANQUE DA QUANSER

Neste tópico, serão mostrados todos os resultados provenientes da simulação do sistema com o tanque de forma cilíndrica da Quanser, avaliando a resposta do controle *fuzzy* Matlab® e a do sistema *fuzzy* em VHDL.

Iniciando com *Set-point* de 25 centímetros, o gráfico de saída do controle *fuzzy* Matlab® mostra uma rápida estabilização, precedido de um sobressinal de aproximadamente 3 centímetros, com um erro de regime permanente nulo.

Com relação a resposta do controlador *fuzzy* em VHDL, ocorreu um pequeno sobressinal, porém de valor menor em relação ao controle *fuzzy* Matlab®, e erro de regime permanente de aproximadamente 0,4 centímetros. O amortecimento da curva que vai do sobressinal até a estabilização da resposta foi um pouco mais lenta, devido aos valores de saída serem discretos, então os valores de passo são fixos, diferente do controle *fuzzy*, onde há uma gama maior de valores que podem ser tomados como saída. Na Figura 32, pode-se ver as respostas dos controladores *fuzzy* VHDL e *fuzzy* Matlab® em relação a um valor de referência fixo.

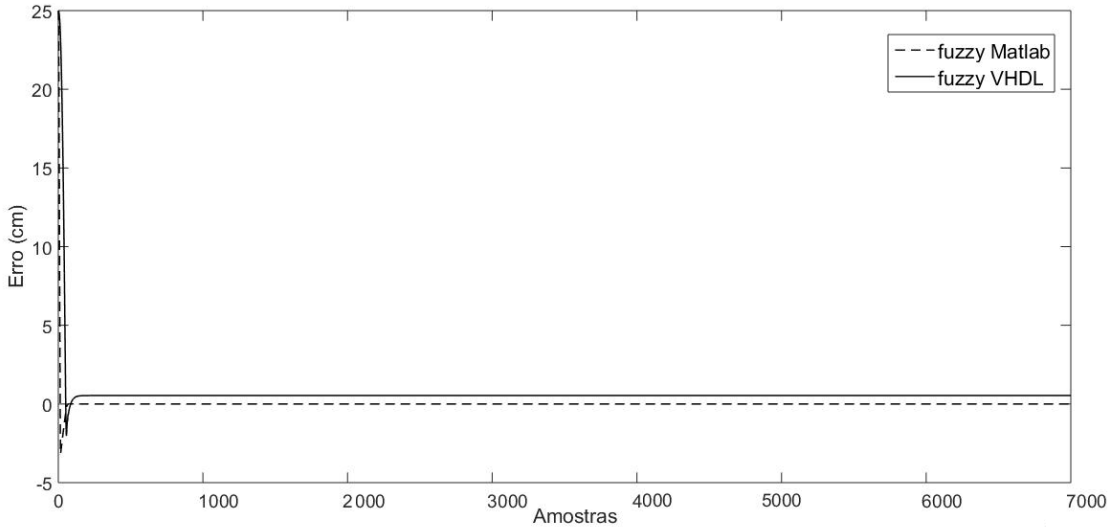
Figura 32 - Saída dos controladores para valor de referência de 25 cm. Tanque cilíndrico



Fonte: Autoria própria (2017)

Como os valores medidos atingiram do valor de referência designado, o erro do sistema *fuzzy* Matlab® foi praticamente nulo, existindo apenas a presença de erro em regime transitório. Após estabilizada, a curva atingiu erro nulo, mostrando que o sistema convergiu para o valor desejado. Houve também um erro transitório no sistema *fuzzy* VHDL, além de ocorrer um pequeno erro de regime permanente. Estas considerações podem ser vistas através da Figura 33.

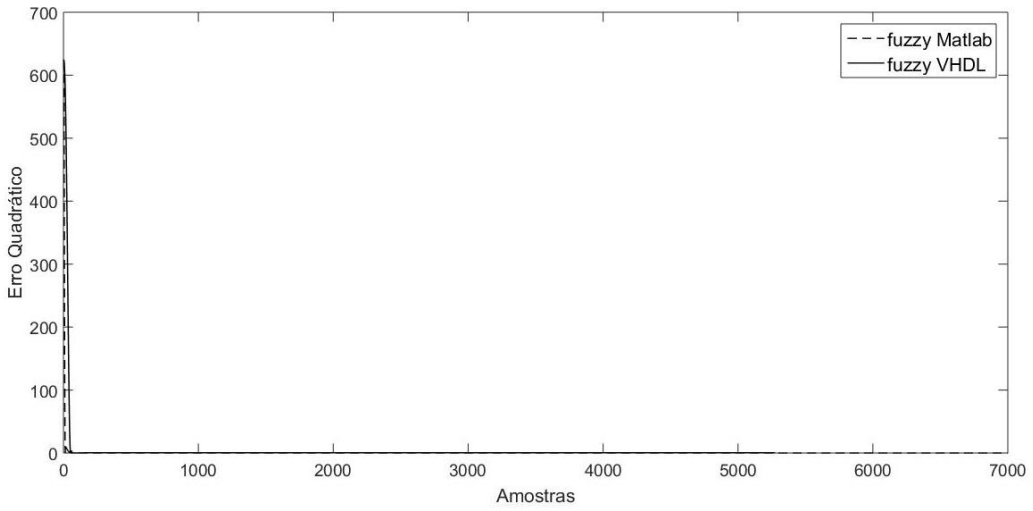
Figura 33 - Erro dos controladores para referência fixa de 25 cm. Tanque cilíndrico



Fonte: Autoria própria (2017)

Outro parâmetro a ser observado é o erro quadrático médio, sendo este um bom indicador de precisão do controle do sistema, determinando em quais períodos de amostragem o modelo não atingiu o valor esperado. O erro quadrático indica os valores iniciais fornecidos pelos controladores que não convergiram para a referência no início da simulação. No controlador *fuzzy* em VHDL, como houve erro em regime permanente, o valor medido é próximo de zero devido à aproximação do *set-point* e segue constante devido ao baixo valor de erro associado a esta implementação. Na figura 34, pode-se observar a curva do erro quadrático para ambos os controladores.

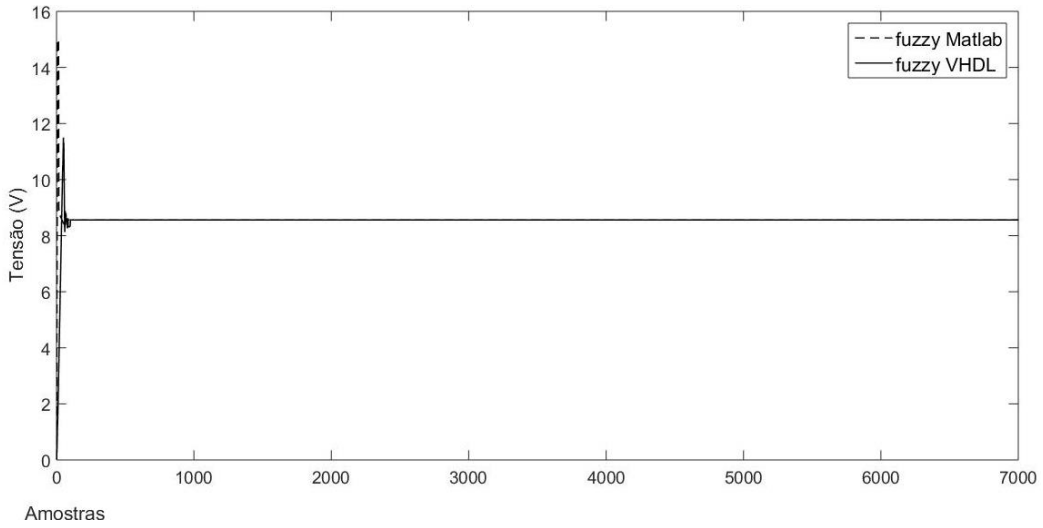
Figura 34 - Erro quadrático dos controladores para uma referência fixa. Tanque cilíndrico



Fonte: Autoria própria (2017)

O último parâmetro analisado é o sinal de controle, onde é mostrado em seu gráfico o nível de tensão que será enviado a bomba. Este parâmetro é importante para averiguar se o atuador em questão não está sofrendo sobrecarga, ou se a oscilação de tensão é alta, comprometendo a vida útil do equipamento. O sinal de controle dos controladores *fuzzy* Matlab® e *fuzzy* VHDL Pode ser visto na Figura 35.

Figura 35 - Sinal de controle dos controladores enviado ao atuador. Tanque cilíndrico



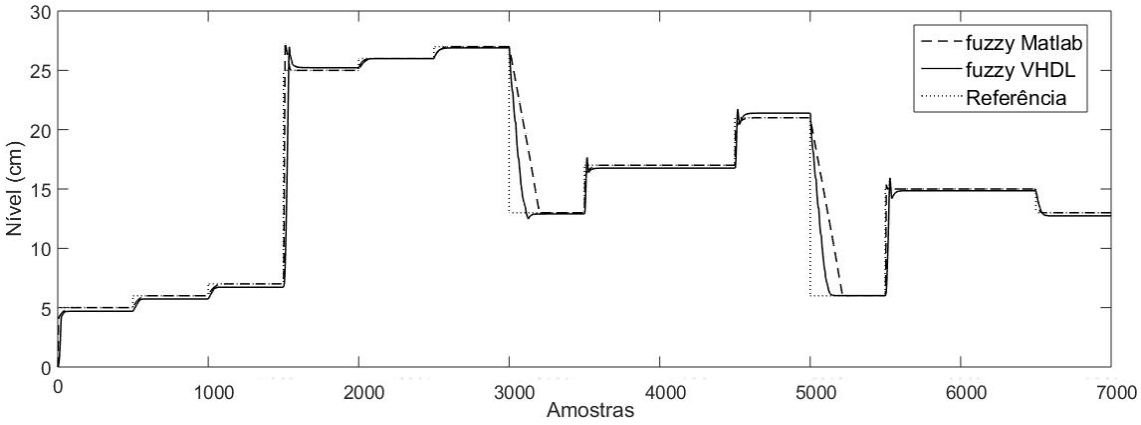
Fonte: Autoria própria (2017)

O nível de tensão do controlador *fuzzy* Matlab® foi crescendo gradativamente ao longo da amostragem, onde indica uma rampa de tensão até 9 amostras, aproximadamente. Houve um período de tensão constante de 15 Volts, e após um tempo, este valor foi reduzido para 9, onde se manteve praticamente constante ao atingir a estabilidade.

No sinal de controle *fuzzy* VHDL, o valor de pico de tensão menor, atingindo um máximo de 11Volts. Como pode-se observar, o gráfico do sinal de controle teve algumas oscilações devido a digitalização do sistema, onde os valores discretos são tratados e enviados ao atuador. Houve pequenas oscilações, e a curva foi semelhante ao do sistema *fuzzy* Matlab®, porém com uma velocidade menor para atingir um nível de tensão constante.

Com a análise dos resultados *Set-point* fixo concluída, serão atribuídos valores de *Set-points* variáveis (faixas de 5 a 27 cm) com o intuito de observar a capacidade de adaptação do controlador. As respostas de saída dos sistemas *fuzzy* Matlab® e *fuzzy VHDL* podem ser vistas na Figura 36.

Figura 36 - Saída dos controladores em resposta a vários sinais de referência. Tanque cilíndrico



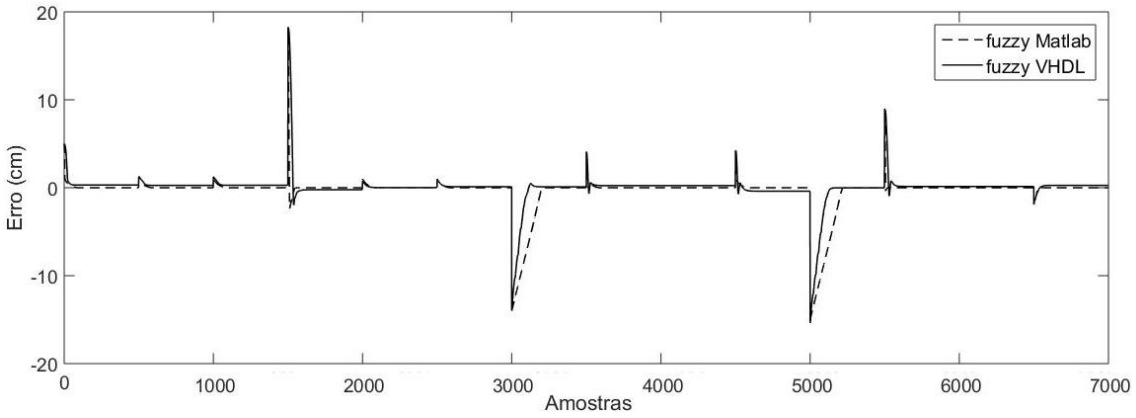
Fonte: Autoria própria (2017)

Para este caso, o controlador *fuzzy* Matlab® obteve uma boa resposta, gerando um erro quando houve variações bruscas de sinal de referência (de 27 para 13 cm, por exemplo), isso se deve às características de sintonização do controlador. Ocorreu um pequeno sobressinal, porém houve uma resposta rápida ao tempo de estabilização.

Em relação ao *fuzzy* VHDL, houve um sobressinal na maior variação do *set-point* e um leve atraso para conseguir atingir um sinal de nível menor que o atual entre as amostras entre 3000 e 4000. Este sistema é um pouco mais lento para atingir o valor de sobressinal máximo, devido ao uso de blocos discretos (*Delay)* na entrada das variáveis *Erro* e *Derivada do erro*, que causam um certo atraso no processamento do sinal. Houve também mais três pontos de sobressinais nas mudanças de um valor de referência menor para um maior.

Como os valores medidos de ambos os controladores foram próximos ao sinal de referência, o erro foi muito próximo de zero e houve sobressinal apenas quando o valor de referência mudou de 7 para 25 centímetros. A Figura 37 mostram os valores de erro nos controladores

Figura 37 - Erro dos controladores *fuzzy* Matlab® no tanque cilíndrico para *Set-points* variados



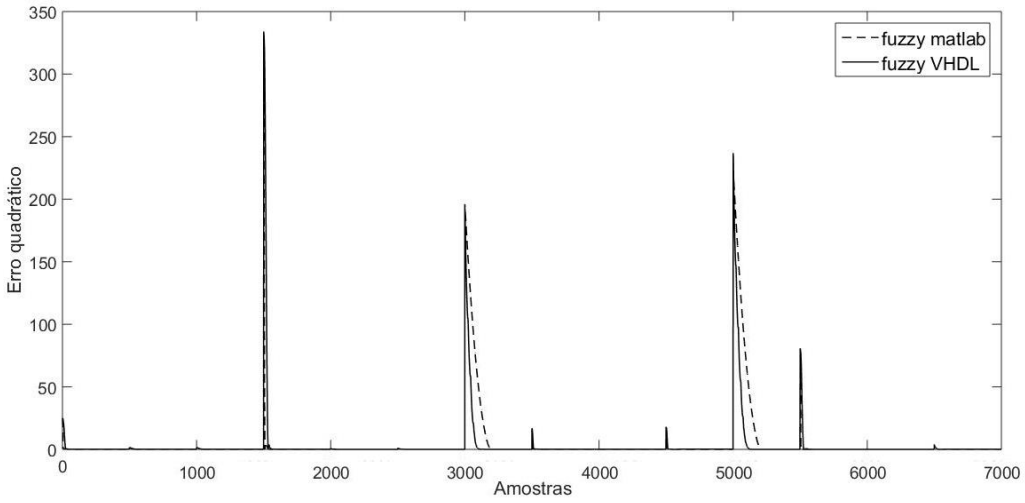
Fonte: Autoria própria (2017)

Em relação ao erro do controlador implementado em VHDL, o sinal também foi semelhante em relação ao sinal do controlador *fuzzy* Matlab®. Como houve pequenos erros de aproximação e de conversão de números decimais em inteiros (para o sistema implementado poder reconhecer o tipo de variável), o sistema implementado herda o erro do controlador *fuzzy* Matlab®, e ainda é acrescido este problema de conversão.

Como pode-se observar, próximo à amostra 2000, houve o maior valor de erro, que corresponde ao sobressinal da saída do sistema, e nas amostras 3000 e 5000 houveram valores de erro negativo, correspondente à diminuição brusca de valor de *set-point*, e com o aumento desta amostragem, o sistema foi se estabilizando e o erro tendendo a zero.

Com o gráfico do erro, é possível calcular o erro quadrático médio, onde os valores serão normalizados e serão representados todos sobre o mesmo eixo vertical, sem valores negativos. A Figura 38 mostra o erro quadrático médio dos sistemas de controle *fuzzy*

Figura 38 - Erro quadrático do controlador *fuzzy* Matlab® para *Set-points* variados. Tanque cilíndrico



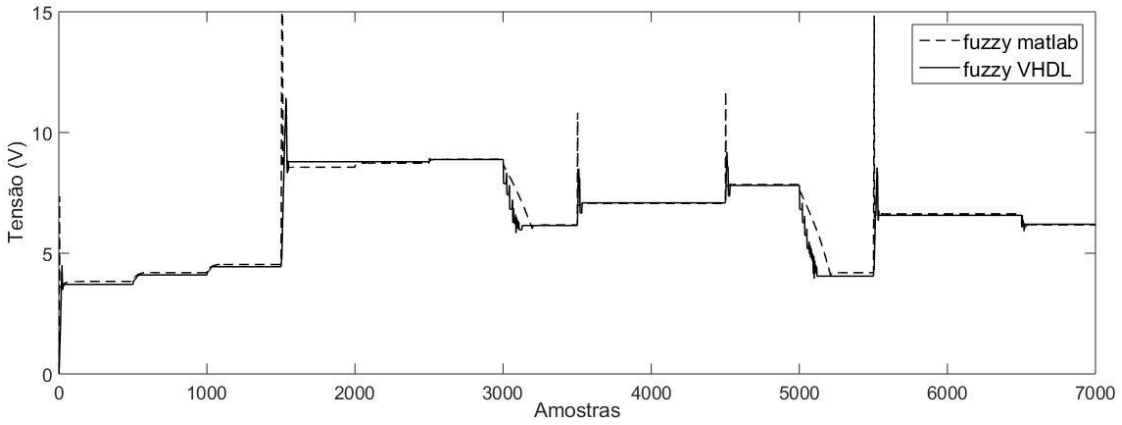
Fonte: Autoria própria (2017)

Considerando que o erro quadrático serve de parâmetro para mostrar zonas de maior divergência, que estão situados onde houve uma maior separação do valor medido com o valor esperado, que são nas mudanças bruscas de *set-point*.

Como o resultado da implementação foi bem fiel ao comportamento do controlador *fuzzy* Matlab®, o erro quadrático também foi semelhante. Na amostra de número 1500, houve o maior valor de erro, seguido das amostras 3000 e 5000. O diferencial deste resultado em relação ao *fuzzy* Matlab® foi o tempo de convergência para o erro quadrático nulo, que foi menor nesta implementação. Outra observação foi que na implementação VHDL, não houve pico na amostra de 6600, ao contrário do erro quadrático no *fuzzy* Matlab®.

Analisado o sinal de controle, nota-se que o maior pico de tensão ocorreu quando houveram variações bruscas de *set-point* ou sobressinal. Na implementação em VHDL do controlador, os resultados foram um pouco diferentes do *fuzzy* Matlab® em relação ao nível de tensão, mas o comportamento do sinal de uma forma geral foi o mesmo. A Figura 39 mostra o sinal de controle do sistema *fuzzy* implementado em VHDL.

Figura 39 - Sinal de controle do sistema *fuzzy* implementado em VHDL para *Set-points* variados. Tanque cilíndrico



Fonte: Autoria Própria (2017)

Analisando o sinal de controle, pode-se observar que o nível máximo de tensão na bomba do sistema *fuzzy* Matlab® atingiu 15 Volts, onde no *fuzzy* VHDL não ultrapassou 12 Volts. A partir da amostra de número 3000, há oscilações quando o sinal de referência varia para um nível menor devido aos valores discretos que a bomba recebia do controlador, diferentemente do sistema *fuzzy* Matlab® que fornecia valores contínuos. Nas amostras de número 3500, 4000, 5600 e 6600 (aproximadamente) houve uma redução significativa da tensão de pico na bomba, de 15(*fuzzy* Matlab®) para o máximo de 9 Volts (*fuzzy* VHDL) dentre as amostras citadas.

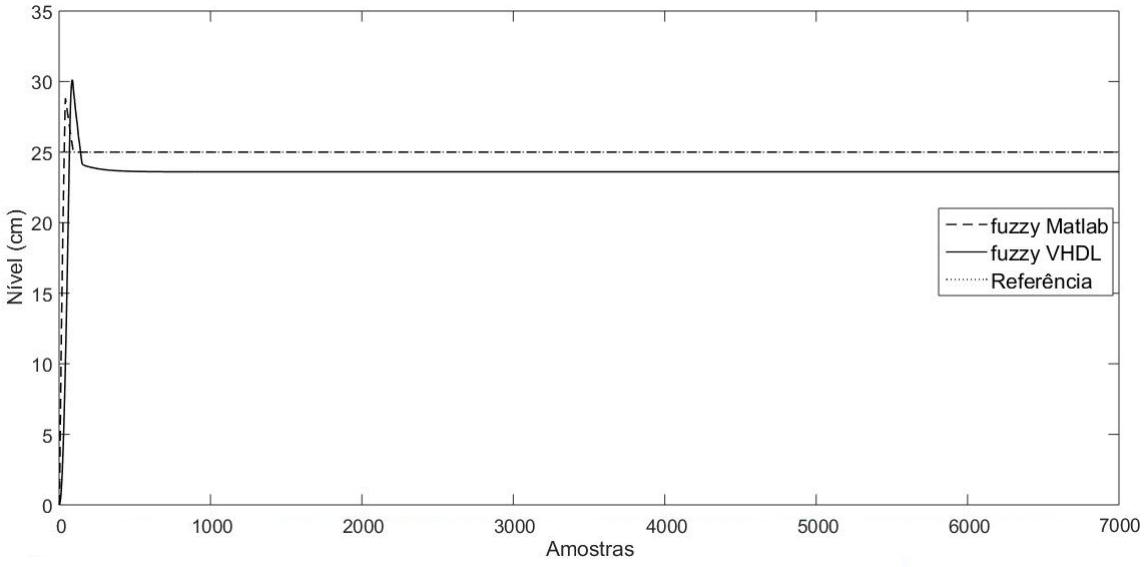
Com os resultados do tanque de formato cilíndrico da Quanser definidos, a próxima etapa será a de avaliar o tanque proposto de formato trapezoidal, onde serão avaliados os mesmos parâmetros e controladores deste tópico.

4.2 RESULTADOS REFERENTES AO TANQUE PROPOSTO DE FORMATO TRAPEZOIDAL

De maneira análoga ao tópico 4.1, será observado a resposta dos controladores estudados (*fuzzy* Matlab® e o *fuzzy VHDL*) para o tanque proposto de formato não linear e geometria trapezoidal. Primeiramente, serão avaliados os parâmetros com o *Set-point* fixo de 25 centímetros, e após finalizada as discussões sobre tais resultados, será avaliado o mesmo sistema para valores de referência variados com o intuito de testar sua adaptabilidade

Em relação a saída do sistema, o comportamento do controlador *fuzzy* Matlab® foi semelhante ao modelo com tanque linear da Quanser, com rápida resposta, pequeno erro de regime transitório, erro de regime permanente nulo e com presença de sobressinal de aproximadamente 3 centímetros antes de atingir a estabilidade. Na saída do controlador *fuzzy* VHDL, sua resposta também se mostrou rápida, com a presença de um sobressinal de aproximadamente 5 centímetros e um erro de regime permanente de aproximadamente 0,8 centímetros. Do valor de pico oriundo do sobressinal até atingir a estabilidade, percebe-se que há um amortecimento da curva de forma mais lenta que a do controlador *fuzzy* Matlab®. As curvas de resposta podem ser vistas na Figura 40.

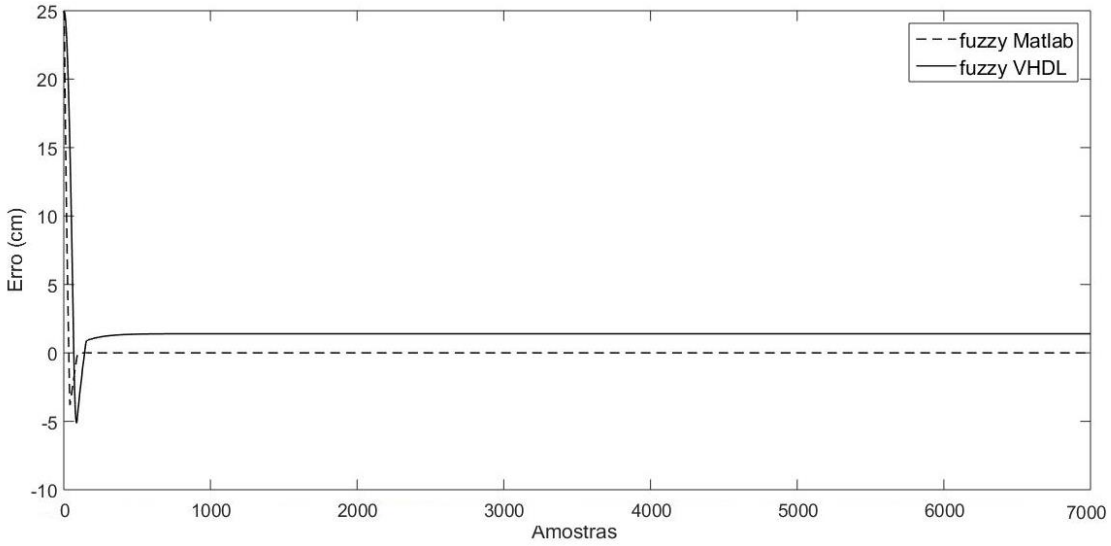
Figura 40 - Saída dos controladores *fuzzy* para um valor de referência fixo. Tanque trapezoidal



Fonte: Autoria própria (2017)

Como a resposta foi satisfatória em relação à saída do controle para ambos os casos, o gráfico do erro, consequentemente, também foi satisfatório, mesmo com oscilações no início da simulação devido ao tempo de subida da curva de resposta e do sobressinal. A Figura 41 mostra a curva de erro dos controladores *fuzzy*.

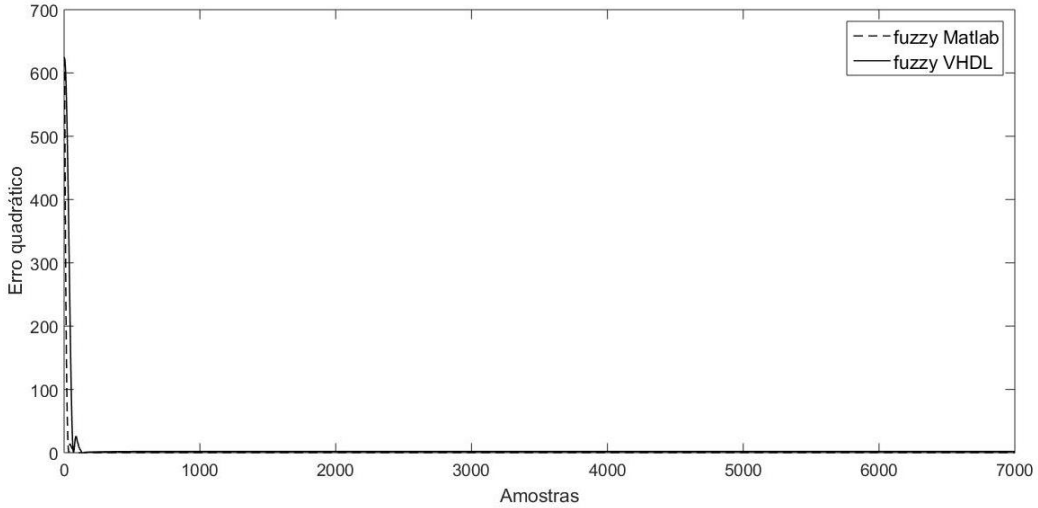
Figura 41 - Erro dos controladores para um valor de referência fixo. Tanque proposto



Fonte: Autoria própria (2017)

Como o erro foi pequeno e quase tende a zero, o erro quadrado também segue este padrão, pois com valores pequenos, as curvas do erro e do erro quadrático se aproximam e tendem a ser iguais. Com valor alto no início e um valor nulo no fim, o erro quadrático mostrou-se estabilizado, e o sistema está totalmente controlado. A Figura 42 ilustra a curva característica do erro quadrático para os dois tipos de controladores.

Figura 42 - Erro quadrático dos controladores *fuzzy* para uma referência fixa. Tanque proposto

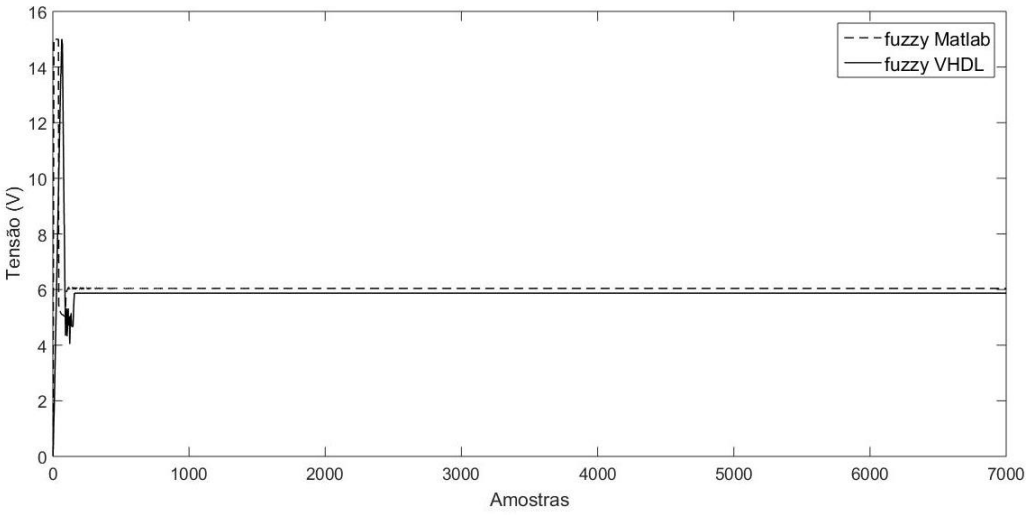


Fonte: Autoria própria (2017)

Para o sistema descrito em VHDL, o valor de erro quadrático começa elevado e diminui ao longo do tempo de amostragem. Ainda há um pequeno aumento em torno da amostra 80, mas rapidamente volta para valores próximos a zero, região esta que é caracterizada pelo regime permanente do sistema.

Com a convergência rápida do controle *fuzzy* Matlab®, os níveis de tensão sob a bomba tendem a ser constantes, variando apenas quando o valor atual está distante do valor de referência, que, para este caso, só se caracteriza no início da simulação, tendo um pico de tensão no momento do sobressinal e amenizando no momento em que o sistema opera em regime permanente e com erro nulo, como mostrado na Figura 43.

Figura 43 - Sinal de controle dos controladores para referência fixa. Tanque proposto

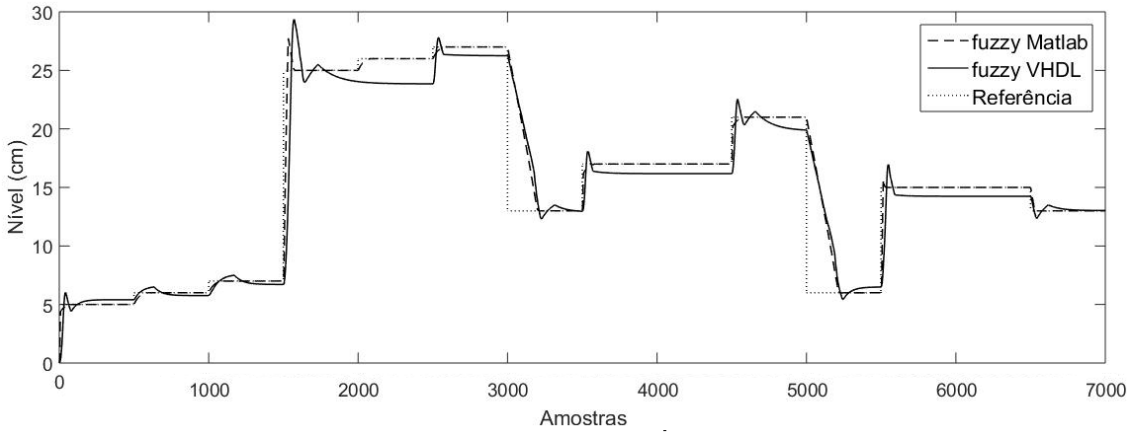


Fonte: Autoria própria (2017)

Os valores de sinal de controle do sistema *fuzzy* VHDL oscilam ao longo do regime transitório, onde só há uma estabilidade de tensão no atuador quando o erro tende a zero. Entre as amostras de número 100 e 150, aproximadamente, há uma grande variação de valores e o controle tenta buscar uma resposta ideal para aquele tipo de situação.

Com os resultados devidamente apresentados com o valor de referência fixo, serão mostrados os resultados gerados através do teste de adaptabilidade dos controladores em relação ao sistema de nível, que além do *set-point* variável, a área do tanque também varia de acordo com o nível da coluna d'água. As saídas dos controladores *fuzzy* podem ser vistas na Figura 44.

Figura 44 - Saída dos controladores *fuzzy* para referências variadas. Tanque proposto

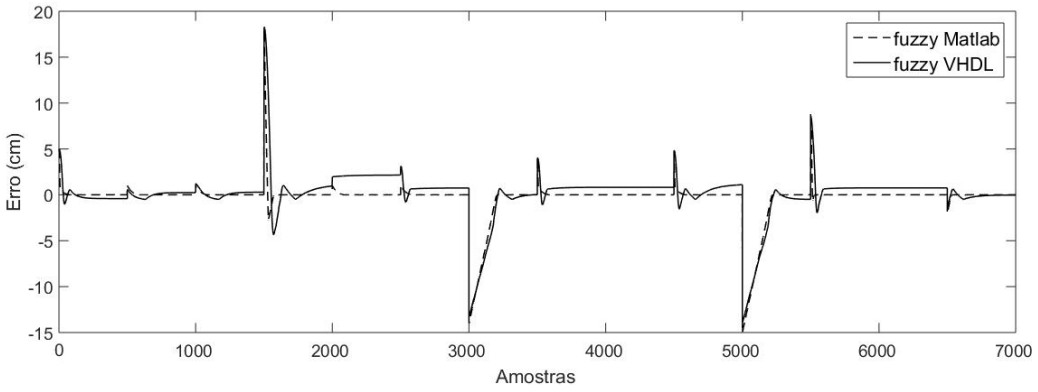


Fonte: Autoria própria (2017)

O controlador implementado em VHDL tentou seguir as mudanças impostas pelo sistema, porém com duas variáveis mudando ao longo do tempo (*set-point* e nível baseado na geometria do tanque), e além do emprego de valores inteiros no controlador previamente configurado, não foi possível atingir determinados valores de referência. O controlador gerou alguns sobressinais e quase não gerou valores abaixo do valor de referência quando havia uma diminuição do *set-point.* A resposta do controlador *fuzzy* Matlab® se mostrou consistente, com dois pequenos sobressinais e estabilização rápida. Contudo, houveram determinadas faixas de amostragem (entre 3000 e 3500, e 5000 e 5500) que o controle não foi eficaz, porém é considerado satisfatório levando em consideração a ação de controle feita como um todo.

Em relação ao erro, os valores são semelhantes ao modelo com tanque da Quanser, referentes aos valores em amplitude e períodos de amostragem. Os erros do sistema gerado pelos controladores podem ser vistos na Figura 45.

Figura 45 - Erro dos controladores *fuzzy* para referências variadas. Tanque proposto

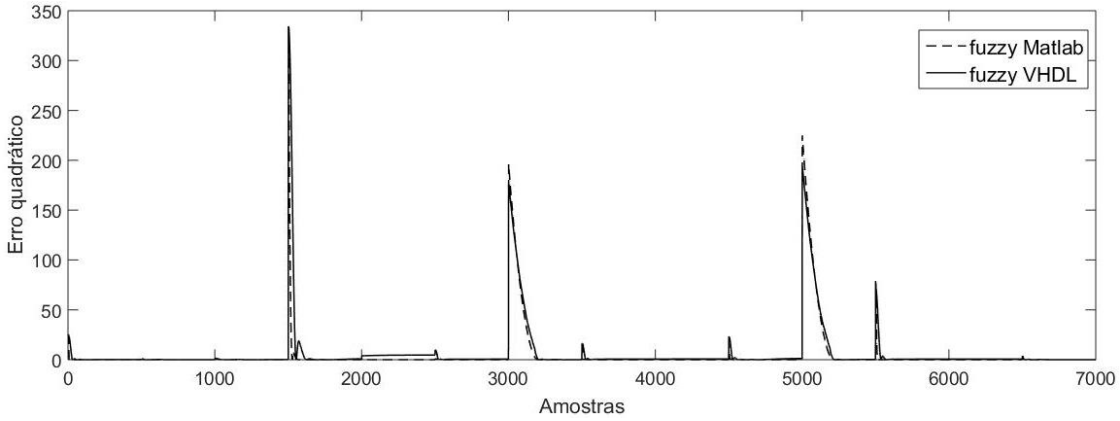


Fonte: Adaptado de Vieira (2017)

Como houveram muitos valores que não atingiram a referência no controlador *fuzzy* VHDL, o gráfico do erro mostrou as oscilações ocorridas ao longo das amostragens.

O Erro quadrático indica de forma normalizada as regiões onde seu valor foi mais acentuado. A Figura 46 mostra o erro quadrático dos controladores *fuzzy* aplicados ao tanque proposto, de formato cilíndrico.

Figura 46 - Erro quadrático do controlador *fuzzy* Matlab® para variadas referências. Tanque proposto

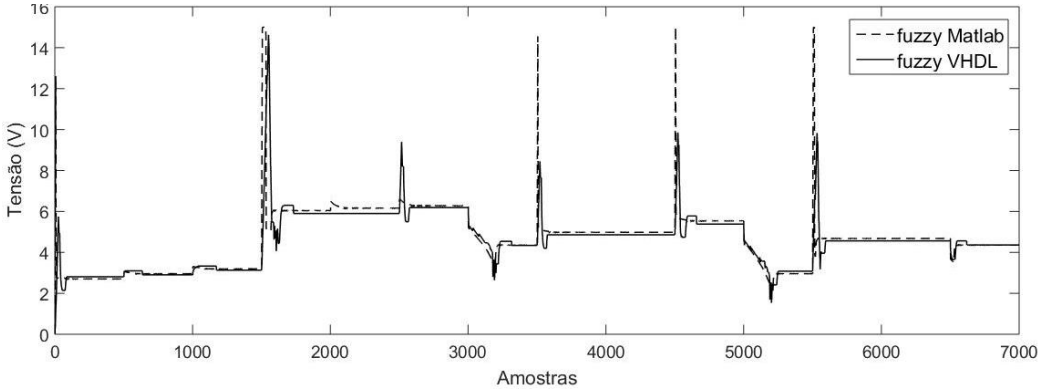


Fonte: Adaptado de Vieira (2017)

Em relação ao erro quadrático do sistema implementado em VHDL, os elevados valores de erro quadrático no início da simulação ocorrem devido ao fato de haver intervalos em que o *set-point* não foi alcançado. Houve também mais três picos ao longo do período sob análise (amostras de número 1500,3000 e 5000), e nos demais pontos, o valor se aproxima de zero, mostrando em quais intervalos se deve trabalhar na

implementação para melhorar a resposta do sistema. A Figura 47 mostra os níveis de tensão aplicados à bomba como forma de sinal de controle fornecido pelo controlador *fuzzy* Matlab®.

Figura 47 - Sinal de controle do controlador *fuzzy* Matlab® para variados valores de referência Tanque proposto



Fonte: Adaptado de Vieira (2017)

O Sinal de controle mostra que ocorreram grandes mudanças de tensão em quatro regiões do gráfico (amostras1500, 3500, 4500 e 5500), sendo estas as regiões que coincidem os maiores valores de erro, pois é enviado um sinal proporcional a bomba para que o erro seja corrigido rapidamente. Houve também três regiões nas quais a bomba teve seu valor de tensão reduzido para se adaptar ao *set-point* do sistema (amostras 1800, 3200 e 5200).

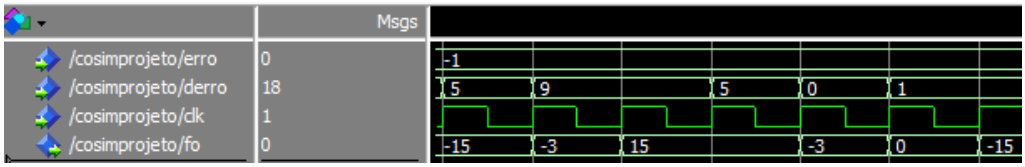
No controlador implementado em VHDL, ocorreu picos de tensão semelhantes ao do sistema *fuzzy* Matlab®, contudo, houve redução significativa após a metade do tempo de simulação, onde os picos de tensão que poderiam chegar a faixa de 15 Volts, foram reduzidos para aproximadamente 9 Volts. Este valor médio de tensão no controlador *fuzzy* VHDL em relação ao *fuzzy* Matlab® pode ser consequência de os valores medidos não atingirem a referência, que refletem diretamente na tensão elétrica sobre o atuador.

Para averiguar se a implementação obteve sucesso, deve-se comparar os valores obtidos pela simulação em Matlab® e pelo simulador no Modelsim/Altera, para calcular o erro percentual e seu erro quadrático médio, como consta na literatura (OLIVEIRA ET AL (2010), UPPALAPATI E KAUR (2009) e AGUILAR ET AL (2014)).

4.3 EFICÁCIA DO CONTROLADOR

Para testar a eficácia da implementação do controlador *fuzzy* descrito em VHDL, os valores de saída deste devem ser comparados com valores de saída do controlador *fuzzy* Matlab®. Primeiramente, valores de *erro, derivada do erro* e saída (*fo*) foram adquiridos através do *Software* Modelsim/Altera Quartus, onde é feito o processamento de sinal da descrição em VHDL. A Figura 48 ilustra os valores de uma parte da simulação no Modelsim/Altera Quartus.

Figura 48 – Resposta da saída *fo* para valores das variáveis Erro e Derro



Fonte: Autoria própria (2017)

Com os valores obtidos no Modelsim/Altera Quartus, deve-se inserir os mesmos valores correspondentes às entradas do sistema (*erro* e *derivada do erro*) no controlador *fuzzy* Matlab® e obter valores de saída. Assim, foi feita a Tabela 3, que mostra os valores de saída de ambos os controladores (*fuzzy* Matlab® e *fuzzy* VHDL) para os mesmos valores de entrada.

Tabela 3 – Resultados da simulação comparando os valors de saída do bloco *fuzzy* e da implementação do sistema *fuzzy* em VHDL.

ERRO	DERRO	Saída Modelsim/Altera	Saída Matlab	Erro (%)
-1	9	-3	-3,26	7,97
4	-42	25	23,1	8,22
1	9	14	13,8	1,45
8	-27	25	23,4	6,83
1	12	15	13,8	8,69
24	-15	25	23,4	6,83
-1	1	-3	-3,5	14,28
-13	26	0	0	0
-10	3	-50	-44,6	12,10
3	-28	25	23,4	6,83

Fonte: Autoria própria (2017)

Com estes resultados, pode-se calcular o erro quadrático médio percentual, que foi de aproximadamente 8,3%, valor este que foi abaixo do valor de 10,46% encontrado em Aguilar et al (2014) e acima dos valores encontrados na literatura (Oliveira et al (2010) com 0,84% e Uppalapati (2009) com 0,8%). Este valor percentual calculado teve contribuição principalmente de valores próximos a zero, tanto do erro quanto da derivada do erro, pois exigia de um grau de precisão maior para representá-los no processo de codificação.

Com o término deste capítulo, seguem as conclusões e recomendações para trabalhos futuros, onde é feita uma síntese de toda a pesquisa e avaliação da implementação de um controlador *fuzzy* em VHDL.

5. CONCLUSÕES E SUGESTÕES PARA TRABALHOS FUTUROS

Neste capítulo, será feita a síntese final do trabalho de acordo com os resultados, levando em consideração as variáveis de controle discutidas, sendo estas: Saídas do controlador, erro, erro quadrático e sinal de controle. Primeiramente, serão realizadas as conclusões para o tanque da Quanser de formato cilíndrico, seguido do sistema de nível feito com tanque proposto, de formato trapezoidal.

Para o tanque cilíndrico da Quanser utilizando *Set-point* fixo, o controlador descrito em VHDL atingiu a estabilidade um pouco depois em relação ao *fuzzy* Matlab®. Contudo, seu sobressinal e o amortecimento do valor de pico foram menores, indicando um valor mais baixo de tensão no atuador para executar o processo. Houve também um pequeno valor de erro de regime permanente em comparação com o *fuzzy* Matlab®, que foi nulo.

Com valores de referência variáveis ao longo do tempo para o tanque da Quanser, a implementação do sistema *fuzzy* em VHDL foi desenvolvida com sucesso, onde a resposta deste controlador assume características semelhantes às do controlador *fuzzy* Matlab®, com ressalva em pequenos valores de sobressinal originados das mudanças de valores de referência. Mesmo com erros de aproximação e de conversão de sinal, o controlador reagiu de maneira satisfatória para o que foi proposto, além de diminuir o valor de pico de tensão na bomba quando observado o sinal de controle do sistema descrito em VHDL.

Para o *Set-point* fixo no sistema de tanque trapezoidal, que varia sua área à medida que o nível também varia, o controlador *fuzzy* VHDL mostrou uma boa velocidade de convergência. Como neste trabalho os valores de entrada são de formato inteiro para a implementação em VHDL, houve uma perda de sensibilidade da parte do controlador, ocasionando um erro de regime permanente em sua saída, onde no controlador *fuzzy* Matlab® não ocorreu.

Ainda com o foco no sistema com tanque trapezoidal, mas agora utilizando *Set-point* variável, a resposta do controlador *fuzzy* VHDL foi insatisfatória, não conseguindo acompanhar devidamente o *set-point*. O formato geométrico do tanque também dificultou o desempenho do controlador, que em alguns momentos não acompanhou a mudança de *Set-point*.

Em relação a eficácia do controlador, que obteve 8,3% de erro quadrático médio, pode-se afirmar que este valor percentual é gerado pelo uso de variáveis inteiras na entrada do controlador *fuzzy* VHDL, não atingindo completamente a sensibilidade que o controlador *fuzzy* Matlab® pode contemplar.

Baseando-se nos resultados discutidos, conclui-se que o controlador *fuzzy* VHDL foi simulado, projetado e modelado com sucesso, e que este converge com erro aceitável para valores de *Set-point* fixos e variáveis para o tanque cilíndrico da Quanser. Já no sistema com tanque de forma trapezoidal, a resposta do controlador *fuzzy* VHDL não foi satisfatória, mesmo com apenas um erro de 8% na eficácia de implementação. Avaliando os sinais de controle, pode-se afirmar que os valores médios fornecidos do controlador em questão à bomba foram menores que do controlador *fuzzy* Matlab®.

Esta pesquisa gerou alguns pontos que podem ser desenvolvidos e gerar trabalhos futuros, sendo estes: Aprimorar o método de implementação *fuzzy* em VHDL, usando as variáveis de entrada como do tipo *Floating point;* inserir a implementação desenvolvida em um FPGA para verificar outro tipo de simulação (*hardware-in-the-loop)* para posteriormente implementar o sistema real embarcado; modificar o tipo de decodificação; aplicar a implementação *fuzzy* em VHDL para um sistema de tanques acoplados; e por último, desenvolver um controlador PI implementado em VHDL e comparar sua eficácia.

REFERÊNCIAS

ABBES, H. **Implementation of a maximum power point tracking *fuzzy* controller on FPGA circuit for a photovoltaic system.** In: INTELLIGENT SYSTEMS DESIGN AND APPLICATIONS (ISDA), 2015, Marraquexe, Marrocos. P.386 – 391.

AGUILAR, A. et al. **Efficient design and implementation of a multivariate Takagi-Sugeno *fuzzy* controller on an FPGA.** In: IEEE INTERNACIONAL CONFERENCE ON MECHATRONICS, ELETRONICS AND AUTOMOTIVE ENGINEERING (ICMEAE), 2014, Cuernavaca, México, p. 152 – 157.

ALMEIDA, J. D. de. **Uma introdução ao estudo da lógica *fuzzy*.** In: Revista de humanidades e ciências sociais aplicadas. [S.l.]: Hórus, 2004.

ANDRÉ H. M. et al. Otimização de controle *fuzzy* usando algoritmo genético. In: **Simpósio Brasileiro de Automação Inteligente (SBAI)**. 2013. Fortaleza, Ceará, Brasil.

BARROS, L. S. et al Métodos Matriciais para Linearização e Representação no Espaço de Estados de Sistemas Elétricos de Potência Contendo Máquinas de Indução Duplamente Alimentadas Operando como Geradores Eólicos, **Anais…** SIMPÓSIO BRASILEIRO DE SISTEMAS ELÉTRICOS, Campina Grande, Paraíba, Brasil. 2006.

BLACHUTA, M. et al. High Performance Single Tank Level Control as an Example for Control Teaching. In:**IEEE MEDITERRANEAN CONFERENCE ON CONTROL AND AUTOMATION (MED)**, 2017,Valleta, Malta. V. 25. P.1053-1058.

BOUSELHAM, L. et al. Hardware implementation of *fuzzy* logic MPPT controller on a FPGA platform**.** In: **IEEE RENEWABLE AND SUSTAINABLE ENERGY CONFERENCE (IRSEC)**, 2015, Marraquexe, Marrocos, V.3. 6p.

CALDEIRA, André Machado et al. **Inteligência Computacional Aplicada À Administração, Economia e Engenharia em Matlab.** São Paulo: Thomson Learning, 2007. 370 p.

CAMPOS, Mario Cesar M.; TEIXEIRA, Herbert C. G. **Controles típicos de equipamentos e processos industriais.** 2ª Edição. São Paulo: Blucher, 2010.

CAPELLI, Alexandre. **Automação industrial:** controle do movimento e processos contínuos. 2ª Edição, São Paulo: Érica, 2008.

CONFESSOR, Sâmya Lorena de Medeiros. **Análise comparativa de controladores MPPT aplicados a um sistema fotovoltaico**. 2014. 72f. Dissertação (mestrado) – Curso de sistemas de comunicação e automação, Universidade Federal Rural do Semiárido, Mossoró, 2014.

CRUZ, A. G. A. A Hybrid System Based on *Fuzzy* Logic to Failure Diagnosis in Induction Motors. In: **IEEE Latin America Transactions.** 2017. V.15. N.8. p 1480 - 1489

D'AMORE, Roberto. **VHDL:** descrição e síntese de circuitos digitais. Rio de Janeiro: LTC, 2005.

DEEPA, P. SIVAKUMAR, R. Synthesis of Heuristic Control Strategies for Liquid Level Control in Spherical Tank**. IEEE Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB).** 3rd. Fev.2017. p. 1-4.

FIGUEIREDO, L. e JOTA, F. (2004). Implementação de técnicas de controle adaptativo ao resfriamento de tiras a quente. **Anais…** XV Congresso Brasileiro de Automática. p 1-6, 2004.

GHOSH, S. et al. A *Fuzzy* Logic System to Analyze a Student's Lifestyle. In: **IEEE Internacional Conference on Advanced Computational Intelligence (ICACI).** 2017. 9th. Doha, Qatar. p. 231-236.

GUARNIZO, J.G., AVENDANO. Liquid Level System as a Pedagogical Tool to Teach *Fuzzy* Control. **IEEE Electronics, Communications and Computers** (CONIELECOMP). Fev. 2017.

JIAN-JUN, Z. Design of *fuzzy* control system for tank liquid level based on wincc and matlab. **IEEE symposium on distributed computing and apllications to business, engineering and Science.** 13 Ed. Nov. 2014

LEE, C. C., ***Fuzzy* logic in control systems:** *Fuzzy* logic controller - part i and part ii. IEEE Transactions on Systems, Man and Cybernetics, 20:404–435, 1990.

MELLO, M. S. **Armazenamento em tanques.** Trabalho complementar para o curso de inspeção de equipamentos. 2015. Disponível

em:<https://pt.slideshare.net/MrioSrgioMello/trabalho-de-armazenamento-em-tanques> Acesso em 02 Ago. 2017.

NASIBOV, E. et al. A *Fuzzy* Logic Approach to Predict the Best Fitted Apparel Size in Online Marketing. In: **IEEE Application of Information and Communication Technologies (AICT).** 2016. 10th. Baku, Azerbaijão. p.1-4.

OLIVEIRA, D. et al. **Design and implementation of a mamdani *fuzzy* inference system on an FPGA using VHDL.** In: IEEE ANNUAL MEETING OF THE NORTH AMERICAN, 2010, Toronto, CA. *Fuzzy* Information Processing Society (NAFIPS).p. 1-6.

QUANSER. **Coupled water tanks –** User manual. Número do documento: 557, rev.5.1. 28p. 2017.

RAMOS, A. P.; WENSE, G. L. B., 2008. **Sistema Didático de Nível de Líquidos**. Monografia - Curso de Engenharia de Controle e Automação, Publicação FT.TG-nº 016/2008, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 78 p.

RODRIGUES, L. M. ; DIMURO, G.P. **Utilizando Lógica *Fuzzy* para Avaliar a Qualidade de uma Compra Via Internet.** In: G.P. Dimuro, L. Foss, M.S. Aguiar, S. Costa, A.C.R. Costa (Eds.), Anais do WEIT 2011 – Workshop-Escola de Informática Teórica, 2011, Pelotas. Rio Grande: FURG, 2011. p.254 – 265. Disponível em: <http://www.gracalizdimuro.com/papers/utilizando-logica-*fuzzy*-para-avaliar-a-qualidade-de-uma-compra-via-internet/>. Acesso em 10 Mai. 2016.

SANDRI S.; CORREA C. **Lógica Nebulosa**. V Escola de Redes Neurais, Promoção: Conselho Nacional de Redes Neurais, São José dos Campos, ITA, pp. c073-c090, jul. 1999. Disponível em: < http://www.gta.ufrj.br/ensino/cpe717-2011/curso_ERN99_*fuzzy*.pdf>. Acesso em: 04 Abr 2016.

SIMÕES, Marcelo Godoy; SHAW, Ian S. **Controle e Modelagem *Fuzzy***. 2. ed. São Paulo: Edgard Blucher, Fapesp, 2007. 186 p.

SINGH,S.; RATTAN, K. Implementation of a *Fuzzy* Logic Controller on an FPGA using VHDL. **IEEE *Fuzzy* Information Processing Society** (NAFIPS). 22 ed. Jul. 2003.

TEIXEIRA JUNIOR, Carlos Alberto. **Análise comparativa entre os controladores *fuzzy* pd+i e pid convencional.** 2014.101 f. Dissertação (mestrado) – Curso de sistemas de comunicação e automação, Universidade Federal Rural do Semiárido, Mossoró, 2014.

THOMAZINI, Daniel; ALBUQUERQUE, Pedro U. B. de. **Sensores industriais**: Fundamentos e aplicações, 4ª Edição (revisada). São Paulo: Érica, 2007.

TODINCA, D.; BUTOIANU, D. VHDL Framework for Modeling *Fuzzy* Automata. **IEEE International Symposium on Symbolic and Numeric Algorithms for Scientific Computing** (SYNASC). 14 ed. Set. 2012.

UPPALAPATI, S.; KAUR, D. **Design and implementation of a mamdani *fuzzy* inference system on an FPGA.** In: NORTH AMERICAN *FUZZY* INFORMATION PROCESSING SOCIETY ANNUAL CONFERENCE (NAFIPS), 2009. Ohio, Estados Unidos. P.1 – 6.

VALE, Marcelo Roberto Bastos Guerra. **Análise Comparativa do Desempenho de um Controlador *Fuzzy* Acoplado a um PID Neural Sintonizado por um Algoritmo Genético com Controladores Inteligentes Convencionais.** 2007.73 f. Dissertação (mestrado) – Curso em Engenharia Elétrica, Universidade Federal do Rio Grande do Norte, Natal, 2007.

Várkonyi-Kóczy, A. R. et al. *Fuzzy* Logic Supported 3D Modeling Based Orthodontcs. In: **IEEE Medical Measurements and Applications (MeMeA).** 2017. Rochester, MN, USA, p.1-6.

VIEIRA, Felipe Bezerra. **ESTUDO COMPARATIVO ENTRE CONTROLADORES *FUZZY* E PI PARA UM SISTEMA DE TANQUE**. 2017.79f. Dissertação (mestrado em sistemas de comunicação e automação) - Universidade Federal Rural do Semi-Árido, Mossoró.

VUONG, P.T. **VHDL IMPLEMENTATION FOR A *FUZZY* LOGIC CONTROLLER.** In: IEEE WORLD AUTOMATION CONGRESS (WAC), 2006, Budapeste, Hungria, p. 1-8.