Análise e Implementação de um Sistema de Inferência Fuzzy tipo-2 intervalar em FPGA

1st Allyson Gonçalves Ramos
Graduando em Engenharia Eletrônica
Universidade Federal de Itajubá
Itajubá, Brasil
allyson.ramos@unifei.edu.br

2nd Wanderson Andrade Viana
Graduando em Engenharia Eletrônica
Universidade Federal de Itajubá
Itajubá, Brasil
av.wanderson@unifei.edu.br

3rd Gabriel Antonio Fanelli De Souza
Orientador do TFG
Universidade Federal de Itajubá
Itajubá, Brasil
fanelli@unifei.edu.br

4th Robson Luiz Moreno
Co-orientador do TFG
Universidade Federal de Itajubá
Itajubá, Brasil
moreno@unifei.edu.br

***Resumo*—Este trabalho visa a análise, a implementação e o aperfeiçoamento de um sistema de controle baseado no método de inferência fuzzy tipo-2 intervalar em FPGA. Para alcançar isso, a simulação do sistema digital (via Intel® Quartus® Prime e ModelSim) é comparada à simulação do sistema real (via Matlab® e Simulink).**

Palavras-chave: ModelSim, Simulink, Fuzzy tipo-2, FPGA

***Abstract*—This work aims the analysis, implementation and improvement of a control system based on the interval type-2 fuzzy inference method in FPGA. To reach this, the simulation of the digital system (using Intel® Quartus® Prime and ModelSim) is compared to the simulation of the real system (using Matlab® and Simulink).**

Keywords: ModelSim, Simulink, type-2 Fuzzy, FPGA

I. Introdução

O primeiro controlador por lógica Fuzzy foi desenvolvido por Mamdani e Assilian, em 1975, no Reino Unido, para controlar um gerador de vapor. O sistema de controle por lógica fuzzy consiste em um sistema não linear baseado em regras e funções de pertinência para obter seu valor de saída. Basicamente, essas regras e funções são obtidas a partir do estudo do sistema a ser controlado. A lógica fuzzy apresenta uma abordagem para tratamento de inferência parecida com a capacidade humana, e que geralmente é capaz de controlar sistemas do mundo real com uma performance superior aos sistemas de controle lineares (como o PID - Proporcional, Integral e Derivativo) [1]. Por exemplo, o ser humano controla a aceleração e/ou frenagem do carro de acordo com a proximidade de outro veículo à sua frente - se estiverem próximos, o carro é desacelerado ou freado e, se estiverem afastados, o carro mantém ou aumenta a aceleração.

Uma FPGA (Field-Programmable-Gate-Array) é um dispositivo lógico programável. Este dispositivo consiste na implementação de circuitos digitais por meio de linguagens de descrição de hardware (VHDL, Verilog HDL, entre outras). Assim, é possível descrever memórias, registradores, flip-flops, barramentos de comunicação, dentre outros tipos de circuitos. Estes circuitos são dedicados para a aplicação e, ao contrário de circuitos embarcados, uma FPGA apenas utiliza os periféricos descritos pela linguagem. Como consequência, FPGAs apresentam uma velocidade superior de processamento quando comparadas aos MCU's ARM [2][3].

Como o foco do trabalho é o aperfeiçoamento e a implementação de um sistema fuzzy tipo-2 em uma FPGA, um passo natural que deve ser dado é o desenvolvimento de modelos de simulação desse tipo de sistema utilizando a ferramenta de simulação ModelSim®, através da sua integração com o software Intel® Quartus® Prime. Com o Quartus, é possível escrever linhas de código para o chamado *testbench*, sendo interpretado pelo ModelSim, que entrega o comportamento digital do sistema, isto é, mostra o comportamento da saída do sistema para cada combinação possível entre ambas as entradas. Além disso, outro passo natural é a validação do sistema de forma comparativa com o mundo real.

II. Proposta

O trabalho proposto constitui-se em três etapas: o estudo de um sistema de controle fuzzy tipo-2 intervalar e sua estrutura, a análise e implementação de otimizações de um sistema fuzzy tipo-2 intervalar em Verilog HDL e a comparação dos resultados obtidos através do sistema em Verilog HDL com o sistema fuzzy tipo-2 intervalar teórico.

O estudo do sistema de controle fuzzy tipo-2 intervalar fornece o conhecimento do conceito da lógica fuzzy, assim como o funcionamento do sistema como um todo. Essa compreensão advém do estudo de cada bloco interno do sistema. Uma vez compreendido o funcionamento interno do controlador, torna-se possível a otimização do sistema fuzzy-2 previamente implementado em [4]. A validação do controlador otimizado (implementado em Verilog HDL) pode ser feita comparando os seus resultados com os resultados de um controlador fuzzy tipo-2 intervalar teórico contendo as mesmas características (regras, funções de pertinência e método de redução/defuzzificação).

Como resultado final é esperado obter um sistema de controle fuzzy tipo-2 intervalar implementado em Verilog HDL que possui duas entradas e uma saída, todas de 8 bits. Este sistema contém seis funções de pertinência trapezoidais (sendo três para cada entrada), opera com nove regras e é baseado no método de redução/defuzzificação de Nie-Tan [5]. Além disso, os parâmetros das funções de pertinência são ajustáveis, de modo que o sistema se torna adaptável para realizar o controle de diversas aplicações.

III. Desenvolvimento

O desenvolvimento do projeto contém cinco etapas: as três etapas apresentadas no capítulo anterior, além de uma etapa intermediária de estudo do sistema apresentado em [4]. Desse modo, este capítulo será dividido em quatro tópicos para enunciar e caracterizar as ações realizadas durante o desenvolvimento do projeto.

A. Estudo de Lógica e Controladores Fuzzy Tipo-1 e Tipo-2

A primeira etapa do projeto consistiu no estudo bibliográfico da lógica fuzzy e de sistemas de controle fuzzy apresentados em [1][4][6][7][8].

Para um rápido entendimento da lógica fuzzy, considere uma paleta cores com tons de cinza. Essa paleta vai desde uma tonalidade totalmente clara (cor branca) até uma tonalidade totalmente escura (cor preta). A tonalidade - seja mais clara ou escura - vai aumentando gradualmente ao decorrer da paleta, o que não restringe a coloração ser somente branca ou preta. A lógica fuzzy permite obter essas diversas ”tonalidades”para um determinado valor. A imagem abaixo ilustra um comparativo entre a lógica booleana (branca ou preta) e a lógica fuzzy (diferentes tonalidades).

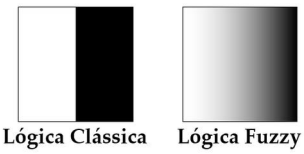

Figura 1. Diferente paletas entre lógica clássica e lógica fuzzy [9]

Os sistemas de controle fuzzy tipo-1 e tipo-2 possibilitam realizar o controle de determinada planta utilizando-se de ferramentas da lógica fuzzy - as diferenças entre sistemas do tipo-1 e do tipo-2 serão tratadas posteriormente. Basicamente, o controlador fuzzy apresenta quatro blocos: bloco de fuzzificação, inferência, regras e defuzzificação.

O bloco de fuzzificação ou fuzzificador é o bloco responsável pela determinação da ”tonalidade”citada anteriormente. Essa ”tonalidade”é chamada de grau de pertinência e indica o quanto um valor de entrada *crisp* qualquer pertence a um conjunto fuzzy específico. O grau de pertinência é obtido através da comparação do valor de entrada com as funções de pertinência do sistema.

Essas funções de pertinência possuem vários formatos e são definidas de acordo com o as necessidades do sistema de controle. Abaixo, segue uma imagem ilustrando alguns tipos de funções de pertinência.

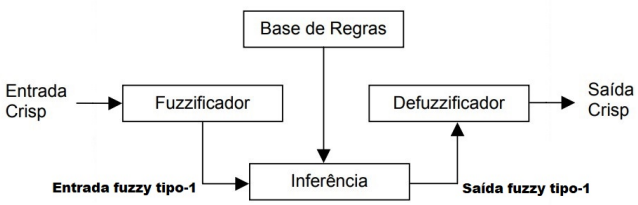

Figura 2. Blocos de um sistema de controle fuzzy tipo-1

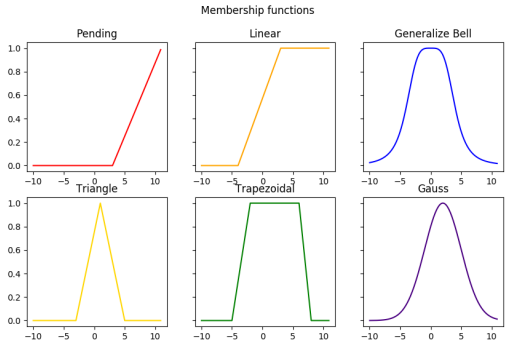

Figura 3. Tipos de funções de pertinência [10]

O grau de pertinência é a saída do bloco fuzzificador e transforma o valor de entrada *crisp* em um valor fuzzificado. Entretanto, na saída do controlador é esperado um valor de saída *crisp*. Logo, ainda são necessários mais alguns passos para obter um valor de saída *crisp* para o controlador.

O sistema de controle fuzzy também apresenta um bloco de regras. Essas regras determinam como será a saída do sistema baseada no grau de pertinência obtido no bloco fuzzificador. Um mesmo sistema de inferência pode ter diversas regras que são definidas através de dados obtidos ou do conhecimento de especialistas [7]. Além disso, são representadas por meio de variáveis linguísticas seguindo o seguinte exemplo:

***SE** antecedente (premissa) **ENTÃO** conclusão (consequência)*
***Se** x é A **então** y é B*

O antecedente é verificado e a consequência é a ação a ser tomada caso o antecedente seja verdadeiro.

O bloco de inferência é responsável por realizar a combinação e cálculo das regras obtendo resultado de saída fuzzy através dos valores dos antecedentes e consequentes. O método de inferência Mamdani dos mínimos e máximos [7] consiste em obter, através da operação de interseção, o mínimo entre os valores antecedentes e, posteriormente, o valor de máximo entre os consequentes através da operação de união. A Figura 4 ilustra a operação do método de inferência Mamdani para os antecedentes A11, A12, A21, A22 e seus consequentes B1 e B2 e o resultado da inferência, que é exibido no útimo gráfico da Figura 4.

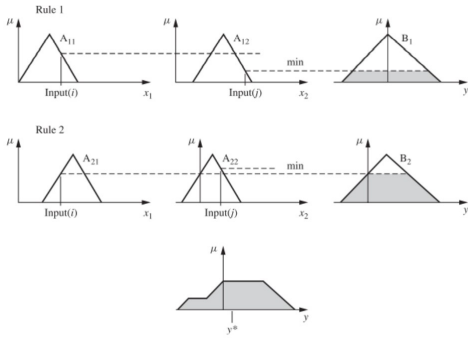

Figura 4. Método de inferência Mamdani [7]

O valor de saída do bloco de inferência é um valor de saída fuzzy, entretanto é necessário convertê-lo para o formato *crisp*. O bloco de defuzzificação é responsável por essa conversão, que é realizada através de alguns métodos de defuzzificação. Os métodos mais utilizados são: método da altura e método do centroide ou centro de área [7].

As descrições dos blocos realizadas anteriormente foram feitas baseadas em um sistema de inferência fuzzy do tipo-1. Para um sistema de inferência fuzzy tipo-2 os mesmos quatro blocos são mantidos, entretanto é necessário um bloco adicional para o tratamento de uma condição especial que será explicada a seguir.

Cada função de pertinência do sistema fuzzy tipo-2 intervalar é composta pelo equivalente à duas funções de pertinência para sistemas do tipo-1. Uma dessas funções é chamada de função de pertinência inferior (FPI) e a outra chamada de função de pertinência superior (FPS). Essas duas funções definem os limites que formam uma mancha de incerteza entre elas, conhecida como *Footprint of Uncertainty* (FOU).

A FOU é uma característica de um conjunto fuzzy tipo-2, que garante uma robustez para aplicações em que há uma incerteza presente na determinação das funções de pertinência, seja no seu formato ou nos seus parâmetros [7]. Desse modo, um mesmo valor de entrada pode assumir diferentes graus de pertinência. A Figura 5 ilustra isso: para o mesmo valor de entrada x = 200 o seu grau de pertinência pode assumir qualquer valor entre 150 e 179,6.

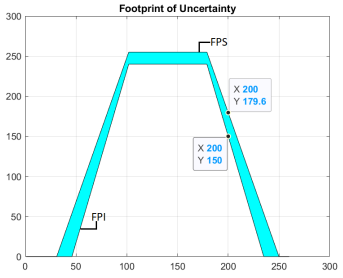

Figura 5. FOU para função de pertinência fuzzy tipo-2

Cada grau de pertinência atrelado a um valor de entrada possui um peso na terceira dimensão, mediante isso é obtida uma função de pertinência secundária no terceiro eixo. Entretanto, visto que as funções utilizadas nesse projeto são da forma intervalar, todos os pesos na terceira dimensão possuem valor unitário. Devido a isso, a função de pertinência pode ser tratada apenas com base na sua FOU [1][7]. A Figura 6 exibe a visão em três dimensões de um sistema fuzzy tipo-2.

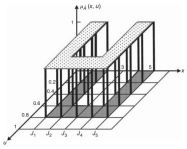

Figura 6. Função de pertinência fuzzy tipo-2 em 3D [1]

Mediante essa condição especial, o bloco adicional de um sistema de controle fuzzy tipo-2 é chamado de bloco redução de tipo. Esse bloco é responsável por transformar os conjuntos de saída fuzzy tipo-2 (com duas possibilidades de valores) em conjunto de saída fuzzy tipo-1 (com apenas um valor de saída). A redução de tipo é realizada mediante alguns métodos [1][8], porém o algoritmo estudado e aplicado nesse projeto é o algoritmo de Nie-Tan [8]. Esse algoritmo realiza o cálculo de redução ao mesmo tempo em que realiza a defuzzificação do valor pelo método da Altura [1][7]. A operação realizada pelo método de Nie-Tan é apresentada na seguinte equação:

$$Y_{NT} = \frac{\sum_{i=1}^{M} y^i (f_{up}^i + f_{low}^i)}{\sum_{i=1}^{M} (f_{up}^i + f_{low}^i)} \quad (1)$$

Sendo Y_{NT} a saida *crisp*, y^i o grau de ativação da regra e f_{up} e f_{low} representam FPI e FPS, respectivamente. A imagem abaixo ilustra um sistema de inferência fuzzy tipo-2.

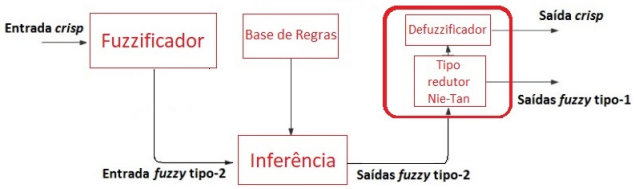

Figura 7. Diagrama de blocos do sistema fuzzy tipo-2

Além disso, para sistemas fuzzy tipo-2, o bloco de fuzzificação gera dois valores de saída para cada entrada *crisp* recebida: um valor referente à FPI e outro referente à FPS. Além desses dois pontos os outros blocos funcionam da mesma maneira para sistemas do tipo-1 e do tipo-2 [7].

B. Interação com o sistema fuzzy tipo-2 implementado em [4]

A descrição de hardware apresentada em [4] consiste em um sistema de inferência fuzzy tipo-2 com duas entradas e uma saída *crisp* de 8 bits. As funções de pertinência superiores e inferiores são do formato trapezoidal, a base de regras conta com nove regras, contém o método de inferência de Mamdani e o método de redução de Nie-Tan. Conta com os seguintes

sinais externos: dois sinais de *clock*, um sinal de *reset* e um de *enable*. O sistema também apresenta 4 módulos principais: FOU, Unidade_controle_regras, inferencia e TR_defuzzy.

O bloco FOU é responsável por realizar a fuzzificação do valor de cada entrada *crisp* e apresenta três submódulos: ffds (flip-flop tipo D para receber o valor de entrada), Trapezio (contém as funções e módulos necessários para a fuzzificação do valor de entrada, além também de blocos multiplicadores e divisores) e geraSativo (responsável por indicar quais funções de pertinência estão ativas para cada uma das entradas). As saídas desse módulo são o valor fuzzificado para cada função de pertinência (FOU_0x_UP e FOU_0x_LOW) e o sinal Ativo_UP, como exibido na Figura 8.

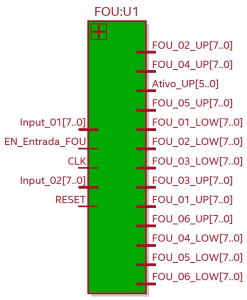

Figura 8. Bloco FOU desenvolvido em [4]

O módulo Unidade_controle_regras apresenta uma máquina de estados responsável pelo controle do restante dos blocos. Essa máquina de estados também controla o acionamento das regras presentes no sistema de inferência. Vale adicionar que a quantidade de regras ativas ao mesmo tempo pode ser: uma, duas ou quatro regras.

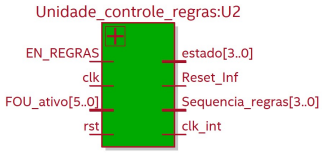

Figura 9. Bloco Unidade_controle_regras desenvolvido em [4]

A base de regras do sistema de inferência fuzzy tipo-2 é apresentada a seguir.

Regra 0:	Se	Entrada 1	é	N1	e	Entrada 2	é	N2	então a saída é	N
Regra 1:	Se	Entrada 1	é	N1	e	Entrada 3	é	Z2	então a saída é	N
Regra 2:	Se	Entrada 1	é	N1	e	Entrada 4	é	P2	então a saída é	Z
Regra 3:	Se	Entrada 1	é	Z1	e	Entrada 5	é	N2	então a saída é	N
Regra 4:	Se	Entrada 1	é	Z1	e	Entrada 6	é	Z2	então a saída é	Z
Regra 5:	Se	Entrada 1	é	Z1	e	Entrada 7	é	P2	então a saída é	P
Regra 6:	Se	Entrada 1	é	P1	e	Entrada 8	é	N2	então a saída é	Z
Regra 7:	Se	Entrada 1	é	P1	e	Entrada 9	é	Z2	então a saída é	P
Regra 8:	Se	Entrada 1	é	P1	e	Entrada 10	é	P2	então a saída é	P

Figura 10. Base de regras do sistema

O módulo Inferencia é responsável pela combinação e cálculo das regras, gerando assim uma saída fuzzy tipo-2. Esse módulo recebe como entrada os valores fuzzificados de cada função de pertinência, um sinal de *clock* e dois sinais advindo do módulo Unidade_controle_regras: um sinal de *reset* e um sinal que indica as regras ativas. Internamente, há dois submódulos: codificador e Unidade_regras. O módulo codificador é responsável por analisar o sinal de regras ativas e codificá-lo de modo a executar corretamente o cálculo das regras no módulo Unidade_regras. A imagem abaixo exibe o módulo Inferencia.

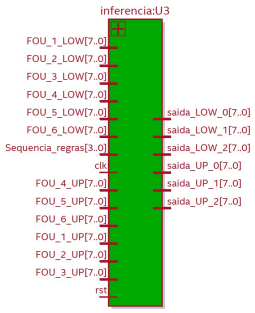

Figura 11. Bloco Inferencia desenvolvido em [4]

O módulo TR_defuzzy recebe como entrada um sinal de *enable*, um sinal de *clock* proveniente de Unidade_controle_regras, um sinal de *reset* e os valores de saída do módulo de inferência. A saída desse módulo consiste no valor *crisp* de 8 bits. Esse módulo é responsável por realizar a redução de tipo e defuzzificação do sistema através da implementação em Verilog HDL do método de Nie-Tan. A figura a seguir apresenta o bloco TR_defuzzy.

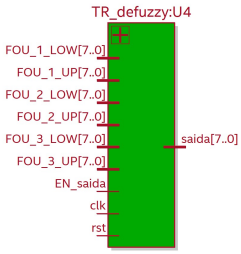

Figura 12. Bloco TR_defuzzy desenvolvido em [4]

C. Levantamento e implementação de melhorias no sistema

No sistema implementado anteriormente, os valores de saída do bloco fuzzificador variam de 0 a 100, para entradas que variam de 0 a 255 (8 bits). Portanto, o primeiro ponto identificado como passível de alteração foi a escala de saída das funções de pertinência, para 0 a 255, a fim de aproveitar toda a faixa dos valores possíveis de entrada.

Além disso, a implementação do fuzzificador no sistema anterior utilizava muitos recursos da FPGA, com uma descrição muito complexa e dividida em vários pequenos blocos. Dessa forma, foi implementado um novo bloco fuzzificador de forma mais simples, reduzindo a quantidade de blocos e de lógicas internas, como multiplicadores e divisores. No sistema anterior, os trapézios são descritos de forma que, mesmo que

tal função de pertinência não seja ativada pela entrada, ela é executada no circuito. No bloco atual, isso foi feito de forma que, caso a função de pertinência não seja ativada, não é realizado o cálculo de grau de pertinência referente à ela. Essa melhoria foi implementada da seguinte forma:

```
if(entrada<=A)
    MF<=8'd0;
else if(entrada<B)
    MF<=255*(entrada-A)/(B-A);
else if(entrada<=C)
    MF<=8'd255;
else if(entrada<D)
    MF<=255*(D-entrada)/(D-C);
else if(entrada>=D)
    MF<=8'd0;
```

Optou-se por este tipo de implementação visto que a construção do trapézio pode ser separada em funções lineares por partes [7], da seguinte forma:

$$\mu(x) = \begin{cases} \frac{x-a}{b-a}, & a < x < b \\ 1, & b \geq x \leq c \\ \frac{d-x}{d-c}, & c < x < d \\ 0, & a \leq x \geq d \end{cases} \quad (2)$$

Ainda no bloco fuzzificador, foi feita a organização do código. Para tal, agrupou-se, no próprio bloco fuzzificador, o sinal que indica quais funções de pertinências estão ativas para posterior tratamento no circuito de inferência. No sistema anterior, são gerados 6 ”geraSativo” dentro do bloco ”FOU”, um para cada função de pertinência. Agora, a divisão está por entrada, sendo três saídas de ”ativo” para a primeira entrada e três para a segunda, além da alteração do seu nome para ”Ativo_UP”. Também foram incluídás legendas de cada bloco interno do fuzzificador, e os parâmetros das funções de pertinência foram externalizados, para que o usuário consiga adaptar o controlador ao sistema a ser aplicado.

Para exemplificar, caso o usuário necessite alterar os parâmetros da função de pertinência ”MF_5_UP”, por exemplo, basta apenas alterar os valores dos parâmetros A, B, C e D, referentes à função ”MF_5_UP”, indicado no código. Esses parâmetros correspondem aos vértices do trapézio, definidos de acordo com a Figura 13.

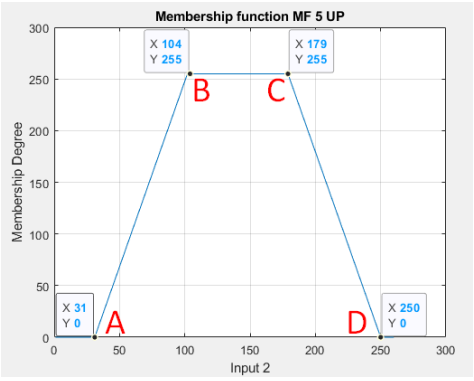

Figura 13. Identificação dos parâmetros das funções de pertinência

Além da organização do bloco fuzzificador, também foram organizados os blocos Fuzzy_1, Unidade_controle_regras e TR_defuzzy da mesma forma como no bloco fuzzificador, a fim de facilitar o entendimento do código.

O sistema implementado em [4] possui dois sinais de *clocks* diferentes para os módulos: o sinal clk_0 para o módulo Unidade_controle_regras e o sinal clk_1 para o módulo inferencia. O sinal clk_1 é utilizado pelo bloco inferencia e é defasado com relação ao sinal clk_0 que, por sua vez é utilizado no bloco Unidade_controle_regras. Entretanto, essa disposição de clocks foi alterada no sistema desenvolvido. Essa alteração foi realizada, primeiramente, para que esses dois sinais de *clock* fossem unificados em apenas um único sinal (nesse caso clk_0). Assim, a defasagem entre eles foi implementada de modo que o bloco inferencia utilizasse a borda de descida do sinal clk_0 e o bloco Unidade_controle_regras utilizasse a borda de subida do clk_0. Após a realização dessas alterações, os sinais de saída do sistema de inferência fuzzy foram analisados e nenhum impacto negativo foi notado.

Entretanto, outra alteração no sinal de *clock* do sistema foi necessário: o sinal de *clock* do bloco inferencia foi alterado para trabalhar na borda de subida. Dessa maneira, os blocos de inferencia e Unidade_controle_regras tornaram-se sincronizados na borda de subida de clk_0. Essa alteração foi realizada de modo a aumentar a frequência máxima de operação do sistema (vide sessão de Resultados).

Após a análise dos sinais de saída do bloco inferencia, notou-se que houve uma pequena alteração: com o sinal de *clock* do bloco na borda de subida o resultado de saída do sistema fuzzy é calculado com um grau de ativação intermediário ao invés de utilizar o valor esperado. Na imagem a seguir, para entradas de 160 e 80, o grau de ativação para o sinal Ssaida_LOW_1 acaba sendo de 154 ao invés 165 (que é o valor esperado). Porém, como o valor dos cálculos acaba sendo truncado, a diferença entre 154 e 165 acaba acarretando em um valor com baixo erro no resultado de saída.

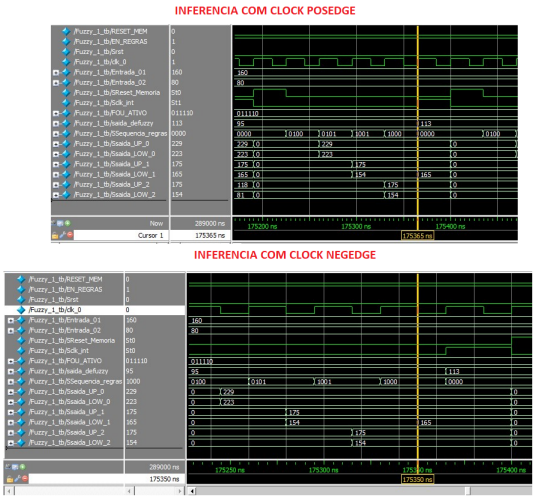

Figura 14. Comparativo do bloco inferencia com *posedge* e *negedge clock*

Por fim, um último ponto de alteração foi a redefinição

da quantidade de bits dos sinais intermediários do bloco TR_defuzzy. O principal motivo foi reduzir a quantidade de bits reservados para os sinais de entrada e saída, visto que não havia necessidade de manter a mesma quantidade implementada anteriormente em [4].

D. Validação das melhorias implementadas

Durante o processo de validação do sistema, foram descobertos pequenos erros que acarretavam o seu malfuncionamento. Utilizando o ModelSim para realizar a simulação a nivel RTL (*Register Transfer Level)*, com um *testbench* criado para variar as entradas em todos os valores possíveis (0 a 255 em cada uma delas), notou-se que o sistema não percorria todo o ciclo esperado. Ao avaliar os caminhos intermediários do sistema, percebeu-se que haviam dois problemas quanto às regras. Na máquina de estados, seguindo pela tabela de regras de MacVicar-Whelan [8], mostrada na Figura 15, quando a Regra 4 era ativada pela condição onde o sinal de "Ativo_UP" equivalia a 010_011 (funções de pertinência MF2, MF5 e MF6 ativas), o estado deveria passar da Regra 4 para a Regra 5, porém se mantinha na Regra 4. Nessa condição, a máquina de estados nunca saía desse estado, travando o sistema.

		Entrada 2		
		MF4	**MF5**	**MF6**
Entrada 1	**MF1**	R0 (N)	R1 (N)	R2 (Z)
	MF2	R3 (N)	R4 (Z)	R5 (P)
	MF3	R6 (Z)	R7 (P)	R8 (P)

Figura 15. Base de regras de MacVicar-Whelan

Devido ao erro descrito, a máquina de estados foi revisada por completo. Após isso, foi observada outra condição, parecida com a descrita acima, referente às regras ativadas pela condição de "Ativo_UP" equivalente a 011_010. Neste caso, as regras ativadas deveriam ser as Regras 4 e 7, porém, o código ativava as Regras 4 e 6. Dessa forma, quando o sistema percorresse estes pontos, o cálculo de inferência seria realizado de maneira incorreta, devido à ativação de regras indesejadas.

Durante o percorrer do trabalho, foram desenvolvidas algumas ferramentas para auxiliar o processo de validação do sistema. Uma ferramenta que auxiliou no rastreamento dos problemas encontrados na base de regras foi uma planilha desenvolvida para visualizar todas as condições de regras possíveis que o sistema ative (uma, duas ou quatro simultâneas), de acordo com cada entrada possível.

Além disso, foi criado um *testbench* que armazena todos os pontos de saída do sistema digital em um arquivo de texto, para posterior tratamento. Ainda, foi desenvolvido um *script* para o Matlab® que gera uma curva percorrendo todos os pontos possíveis para ambas as entradas, gerando um total de 65025 pontos. Atrelado a esse *script*, foi criado outro *script* com a finalidade de fazer a comparação entre a curva de saída do Matlab®, que simula o sistema real (gerada com valores teóricos), e a curva de saída do sistema digital, gerada no Matlab® após tratamento interno do arquivo de texto com os dados armazenados pela simulação do sistema dgital utilizando o Intel® Quartus® e o ModelSim.

IV. Resultados

Os resultados obtidos neste trabalho serão separados por tópicos.

A. Resultados das otimizações realizadas no sistema

Apenas com a implementação do bloco fuzzificador atualizado, o sistema tornou-se notoriamente mais simples, como pode ser visto na comparação do diagrama de blocos do fuzzificador anterior com o atual.

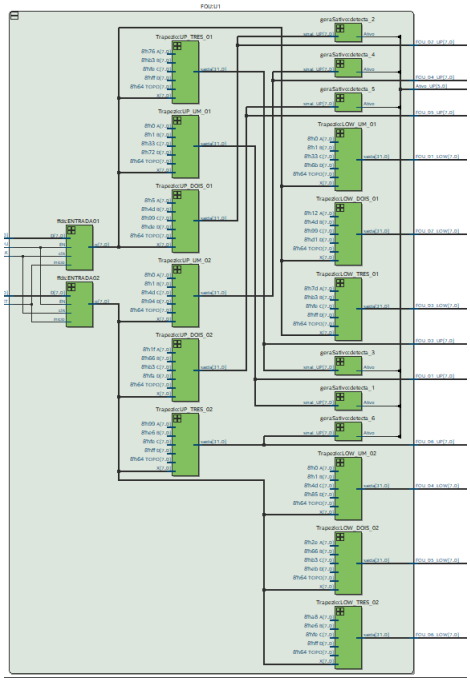

Figura 16. Bloco fuzzificador antigo

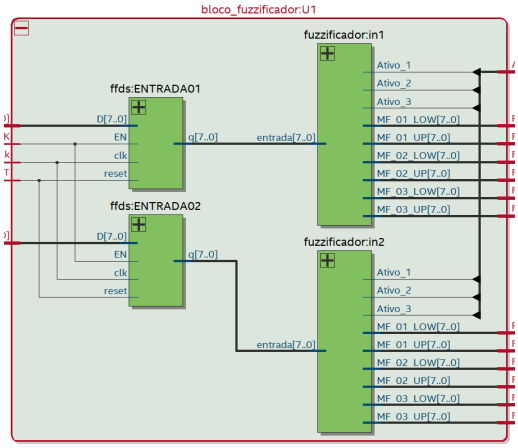

Figura 17. Bloco fuzzificador atual

A alteração da faixa dos valores de saída das funções de pertinência também foi implementada, como pode-se observar na comparação das funções do sistema anterior com o atual através da simulação pelo ModelSim, nas Figuras 18 e 19:

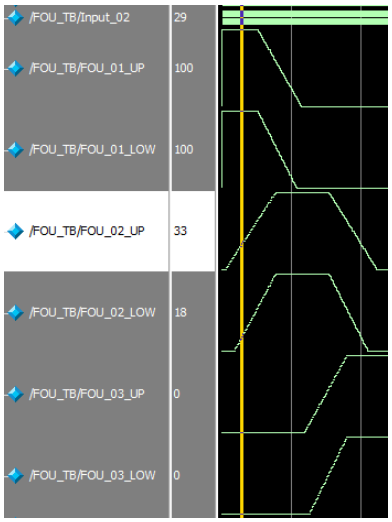

Figura 18. Valores das funções de pertinências da entrada 1 no sistema inicial

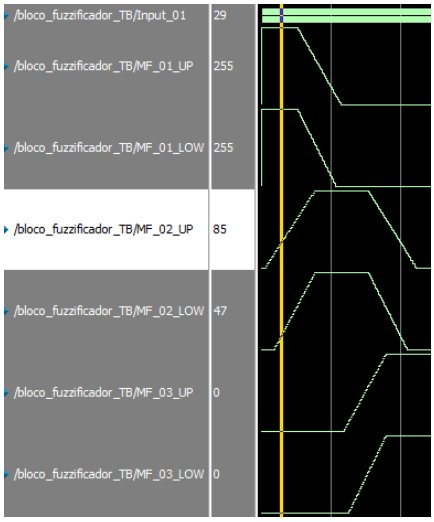

Figura 19. Valores das funções de pertinências da entrada 1 no sistema atual

O ponto mostrado é referente ao valor de entrada igual a 29. Observa-se que, no sistema implementado em [4], o valor de saída para a função de pertinência ”FOU_02_UP” é 33, visto na Figura 18. Essa função corresponde à função ”MF_02_UP” na Figura 19, que possui 85 como valor correspondente.

B. Resultados das validações do sistema final

O gráfico de comparação das curvas de saída dos sistemas teórico com o sistema trabalhado, feito através do Matlab®, ilustra bem a diferença entre o sistema digital e o mundo real. Como se trata de um gráfico em três eixos, não é tão visível colocando no papel e, por isso, serão apresentadas duas vistas das saídas dos sistemas sobrepostas uma à outra nas duas imagens a seguir:

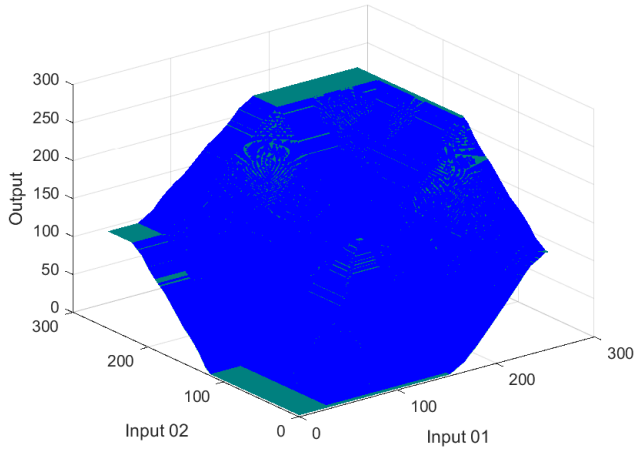

Figura 20. Vista frontal da comparação do sistema real (azul) e o sistema digital (verde)

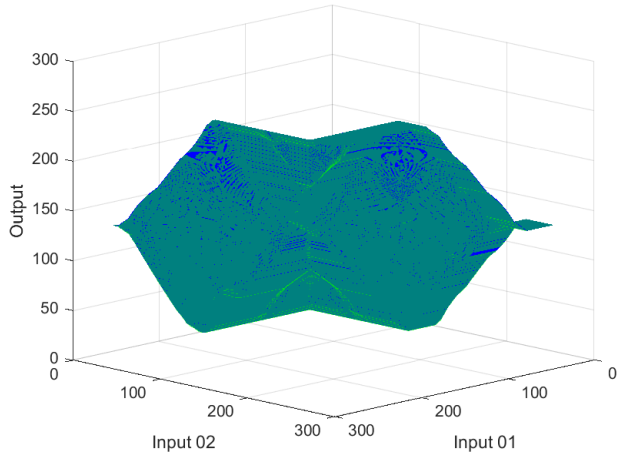

Figura 21. Vista traseira da comparação do sistema real (azul) e o sistema digital (verde)

No gráfico, têm-se ambas as entradas nos eixos x e y, variadas de 1 a 255 (com o intuito de varrer toda a faixa de valores possíveis), e em Z a saída dos sistemas. Pode-se observar que as superfícies estão muito próximas uma da outra, sendo este um indício de que o sistema em Verilog HDL possui uma resposta satisfatória, bem próxima do esperado.

Com a finalidade de comprovar se o sistema implementado é de fato satisfatório, foi gerada a curva de subtração entre as saídas do sistema digital e real, que representa o erro entre ambos os sistemas. A curva é apresentada a seguir:

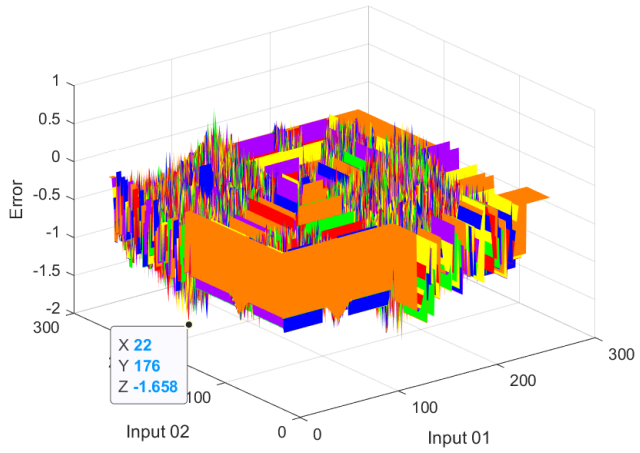

Figura 22. Erro do sistema final em valores numéricos

Na Figura 22, foi destacado o ponto de maior erro do sistema. Neste ponto, os valores referentes às entradas 1 e 2, respectivamente, são 21 e 175. Na imagem, os valores marcados são 22 e 176, dado que o primeiro índice válido do Matlab® é 1 e, portanto, todos os valores estão deslocados em 1. Outra forma de representação do erro é em porcentagem, tornando a escala mais visual. O gráfico do erro em porcentagem é demonstrado na Figura 23:

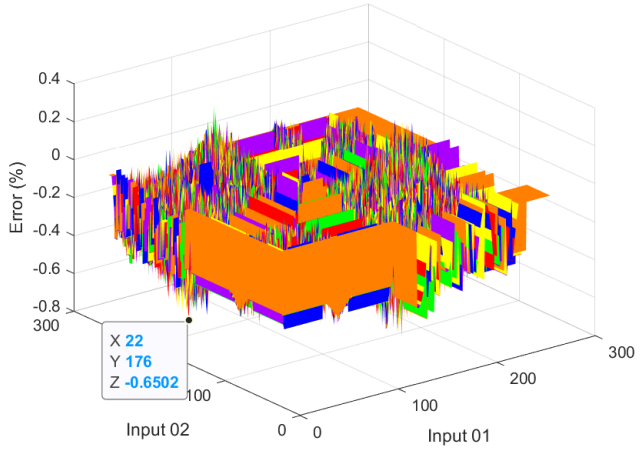

Figura 23. Erro do sistema final em porcentagem

Dessa forma, conclui-se que o maior erro do sistema é de 0,65%.

Após realizar a comparação do sistema inicial com o final, nota-se uma simplificação expressiva, de modo que o total de elementos lógicos do sistema foi reduzido em aproximadamente 75%. A quantidade de elementos lógicos anterior era próxima a vinte e cinco mil e agora está próxima a seis mil, como apresentado nas Figuras 24 e 25.

Total logic elements	25,473 / 114,480 (22 %)
Total registers	76
Total pins	28 / 529 (5 %)
Total virtual pins	0
Total memory bits	0 / 3,981,312 (0 %)
Embedded Multiplier 9-bit elements	64 / 532 (12 %)
Total PLLs	0 / 4 (0 %)

Figura 24. Elementos lógicos do sistema inicial

Total logic elements	6,262 / 114,480 (5 %)
Total registers	76
Total pins	39 / 529 (7 %)
Total virtual pins	0
Total memory bits	0 / 3,981,312 (0 %)
Embedded Multiplier 9-bit elements	0 / 532 (0 %)
Total PLLs	0 / 4 (0 %)

Figura 25. Elementos lógicos do sistema final

É possível notar que o número de multiplicadores embarcados de 9 bits foram retirados do sistema, tornando-o bem simplificado. Isso também foi percebido observando o tempo de compilação do código, que foi reduzido de 6 minutos e 30 segundos para 1 minuto e 10 segundos, isto é, uma redução próxima de 80%.

C. Resultados do TimeQuest Timing Analyzer

Dadas as alterações e validações implementadas no Quartus® e Matlab® foi realizada a verificação do *TimeQuest Timing Analyzer* para obter a frequência máxima de operação do sistema. Considerando a FPGA Ciclone IV EP4CE115F29C7, condições de operação do modelo *Slow* 1200 mV 85C e utilizando o Quartus® como ferramenta, foi realizado o procedimento para a *constraint* padrão de 1 *ns* presente no software citado. Enquanto o bloco de inferencia operava na borda de descida do sinal clk_0 o valor de frequência obtido fica por volta de 51 MHz. Realizada a alteração para que o bloco inferencia opere na borda de subida, o valor de frequência máxima de operação obtido fica por volta de 91 MHz. A imagem a seguir indica o resultado do *TimeQuest Timing Analyzer*:

	Fmax	Restricted Fmax	Clock Name	Note
1	90.86 MHz	90.86 MHz	clk_0	

Figura 26. Frequência de operação máxima do sistema

V. Conclusão

A. Conclusão do trabalho desenvolvido

Este trabalho cumpriu o principal objetivo proposto, visto que foi obtido um modelo de sistema de inferência fuzzy tipo-2 intervalar simplificado implementado em Verilog HDL que permite ao usuário adaptá-lo ao seu sistema, com parâmetros ajustáveis. Além disso, percebeu-se que o maior erro do valor de saída do sistema quando comparado com o sistema teórico foi de 0,65%. Apesar das alterações realizadas de

modo a simplificar a descrição do circuito, para facilitar o entendimento do mesmo, a frequência máxima de operação obtida foi de 90 MHz. Entretanto, com a otimização do código, o tempo de compilação teve uma drástica redução, assim como a quantidade de recursos lógicos utilizados.

Outro objetivo atingidido foi a aprendizagem do grupo sobre o tópico de sistemas de inferência fuzzy tipo-2, visto que esse assunto não é tratado no curso de graduação em Engenharia Eletrônica. Essa experiência foi obtida através da realização de cursos, revisão bibliográfica e exercícios práticos realizados. Unindo esse novo conhecimento ao tópico de FPGAs/Verilog HDL, juntamente com os trabalhos realizados anteriormente, foi possível a implementação e otimização desse sistema.

B. Sugestões para trabalhos futuros

O valor de frequência máxima do circuito pode ser melhorado na mesma medida da simplificação do circuito e pode seguir como sugestão para trabalhos futuros.

No sistema trabalhado, todas as funções de pertinência possuem seu limite máximo em 255. Com a finalidade de otimizar o sistema para o usuário, pode ser estudado uma forma de transformar o topo das funções de pertinência do sistema ajustável, visto que as funções de pertinência inferiores, para sistemas fuzzy tipo-2, podem ter seu limite máximo menor que as funções superiores.

Para os sinais que indicam quais funções de pertinência estão ativas, os "Ativo_UPs", é interessante que seja implementado um sistema de travamento para que o sinal não seja enviado sempre, apenas quando solicitado, a fim de não influenciar na conta do próximo valor de entrada.

Por fim, é interessante que seja feito o *deployment* do código em uma FPGA, de preferência do mesmo modelo utilizado neste trabalho, a Cyclone IV EP4CE115F29C7, para validar não só a linguagem de descrição Verilog HDL como o Hardware de fato, além de se tornar uma ferramenta para o uso em aplicações futuras a serem desenvolvidas.

REFERÊNCIAS

[1] J. M. Mendel *et al.*, Introduction to Type-2 Fuzzy Logic Control: Theory and Applications, *Piscataway: IEEE Press*, 2014.

[2] F. Vahid, Sistemas Digitais, *Grupo A - Bookman*, 2009.

[3] Arm, "What Is an FPGA?". [Online]. Disponível em: https://www.arm.com/glossary/fpga/

[4] TEIXEIRA, Aline Aires. Controlador Fuzzy Tipo 2 com Programação em Verilog das Funções de Pertinência. 2022. 57p. Mestrado em Engenharia Elétrica - IESTI.

[5] M. Nie and W. W. Tan, "Towards an efficient type-reduction method for interval type-2 fuzzy logic systems", *IEEE International Conference Fuzzy System*, 2008.

[6] MathWorks®, "What Is Fuzzy Logic?". [Online]. Disponível em: https://www.mathworks.com/help/fuzzy/what-is-fuzzy-logic.html/

[7] MACIEL, Regimar. Sistema fuzzy tipo-2 intervalar implementado em FPGA baseado no algoritmo de Nie-Tan. 2020. 94p. Mestrado em Engenharia Elétrica - IESTI.

[8] SOUZA, Gabriel Antonio Fanelli de. A Novel Power-Reducing Architecture For a Current Mode Analog Interval Type-2 Fuzzy Logic Inference System. 2019. 119f. Thesis of Doctor of Science in Electronic Engineering and Computer Science, Field of Electronic Systems and Devices – Instituto Tecnológico de Aeronáutica, São José dos Campos.

[9] KOHAGURA, Tiago. Lógica Fuzzy e Suas Aplicações. 2007. 61p. Graduação em Ciência da Computação - Universidade Estadual de Londrina.

[10] M. Santos, "Membership functions", [Online]. Disponível em: https://github.com/VManuelSM/Membership-functions