UNIVERSIDADE FEDERAL DE ITAJUBÁ - UNIFEI
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Controlador *Fuzzy* Tipo 2 com Programação em Verilog das Funções de Pertinência

Aline Aires Teixeira

Itajubá, 29 de agosto de 2022

UNIVERSIDADE FEDERAL DE ITAJUBÁ - UNIFEI
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Aline Aires Teixeira

Controlador *Fuzzy* Tipo 2 com Programação em Verilog das Funções de Pertinência

Dissertação submetida ao Programa de Pós-Graduação em Engenharia Elétrica como parte dos requisitos para obtenção do Título de Mestre em Ciências em Engenharia Elétrica.

Área de Concentração: Microeletrônica

Orientador: Prof. Dr. Gabriel Antônio Fanelli de Souza
Coorientador: Prof. Dr. Robson Luiz Moreno

29 de agosto de 2022
Itajubá

UNIVERSIDADE FEDERAL DE ITAJUBÁ - UNIFEI
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Controlador *Fuzzy* Tipo 2 com Programação em Verilog das Funções de Pertinência

Aline Aires Teixeira

Dissertação aprovada por banca examinadora em 29 de Agosto de 2022, conferindo ao autor o título de **Mestre em Ciências em Engenharia Elétrica.**

Banca Examinadora:
Prof. Dr. Gabriel Antônio Fanelli de Souza
Prof. Dr. Robson Luiz Moreno
Prof. Dr. Tales Cleber Pimenta
Prof. Dr. Thiago Pouza Mussolini

Itajubá
2022

UNIVERSIDADE FEDERAL DE ITAJUBÁ - UNIFEI
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Controlador *Fuzzy* Tipo 2 com Programação em Verilog das Funções de Pertinência

Aline Aires Teixeira

Dissertação submetida ao Programa de Pós-Graduação em Engenharia Elétrica como parte dos requisitos para obtenção do Título de Mestre em Ciências em Engenharia Elétrica.

Trabalho aprovado. Itajubá, 29 de Agosto de 2022:

Prof. Dr. Gabriel Antônio Fanelli de Souza
Orientador

Prof. Dr. Robson Luiz Moreno
Coorientador

Prof. Dr. Tales Cleber Pimenta

Prof. Dr. Thiago Pouza Mussolini

Itajubá
29 de agosto de 2022

Agradecimentos

Gostaria de agradecer, primeiramente a Deus, por estar sempre ao meu lado.

Aos meus pais, Paulo Cezar e Lucimara por sempre me apoiarem, e as minhas irmãs Poliane e Ana Paula por me incentivarem. A toda minha família por sempre acreditaram em mim.

Aos meus amigos que sempre me ajudaram e torceram por essa conquista, em especial a Elvira e Fidel, por serem parte importante desse processo.

Aos meus colegas do mestrado, do grupo de microeletrônica, que se tornaram amigos, do café ao compartilhamento de dúvidas.

Aos meus orientadores Gabriel Fanelli e Robson Moreno, por sempre me ajudarem, motivarem e por todo conhecimento compartilhado.

Ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Itajubá.

A Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo apoio financeiro prestado na forma de bolsa de estudos.

"Tudo posso naquele que me fortalece."
(Filipenses 4:13)

Resumo

O Sistema de inferência *fuzzy* tipo-2 apresenta um desempenho superior em relação a outros métodos para aplicações em sistemas com incerteza. Este trabalho apresenta a arquitetura de um SIF (*System Inference Fuzzy*) tipo-2 que tem como objetivo utilizar a função de pertinência trapezoidal que garante circuitos mais rápidos e simples. A arquitetura implementada em FPGA (*Field Programmable Gate Array*) consiste em cinco blocos, fuzzificador, inferência, base de regras, tipo-redutor e defuzzificador. O hardware apresentado consiste em duas entradas de 8 bits com três funções de pertinência trapezoidais para cada entrada, nove regras e uma saída de 8 bits com três funções de pertinência. Os resultados da implementação em Verilog são comparados com a mesma arquitetura implementada no Matlab® utilizando a Toolbox para type-2 fuzzy.

Palavras-chaves: FPGA. Função de pertinência trapezoidal. Sistema de inferência *fuzzy* tipo-2.

Abstract

The *fuzzy* type-2 fuzzy inference system presents a superior performance in relation to other methods for applications in systems with uncertainty. This work presents the architecture of a SIF type-2 that aims to use the trapezoidal membership function that ensures faster and simpler circuits. The architecture implemented in FPGA consists of five blocks, fuzzifier, inference, rule base, type-reductor and defuzzifier. The hardware presented consists of two 8-bit inputs with three trapezoidal membership functions for each input, nine rules and an 8-bit output with three membership functions. The Verilog implementation results are compared with the same architecture implemented in Matlab® using the Toolbox for type-2 fuzzy.

Key-words: FPGA. Function membership trapezoidal. System inference *fuzzy* type-2 .

Lista de ilustrações

Lista de abreviaturas e siglas

FOU	*Footprint of Uncertainty*	17
FPGA	*Field Programmable Gate Array*	7
LMF	*Lower Membership Function*	9
MFs	*Membership Functions*	16
RTL	*Register Transfer Level*	33
SIF	*System Inference Fuzzy*	7
TR	*Type Reductor*	17
UMF	*Upper Membership Function*	9

Lista de símbolos

$+$	Soma	29
$=$	Igual	21
A_1	Conjunto *fuzzy* tipo - 1 A_1	27
A_2	Conjunto *fuzzy* tipo - 2 A_2	27
A	Conjunto *fuzzy* A	21
B_1	Conjunto *fuzzy* tipo - 1 B_1	27
B_2	Conjunto *fuzzy* tipo - 2 B_2	27
B	Conjunto *fuzzy* B	26
$C_1^{'}$	Conjunto *fuzzy* tipo - 1 $C_1^{'}$	27
$C_2^{'}$	Conjunto *fuzzy* tipo - 1 $C_2^{'}$	27
$J_x \subseteq [0,1]$	Pertinência primária de x contido entre 0 e 1	23
J_x	Pertinência primária	24
$LMF(\tilde{A})$	Função de pertinência do tipo-2 inferior	23
$UMF(\tilde{A})$	Função de pertinência do tipo-2 superior	23
X	Universo de discurso X	21
Y	Universo de discurso Y	26
Y_{NT}	Valor real crisp	29
$\cap$	Interseção	27
$\cup$	União	27
$\forall$	Para todo	23
$\in$	Pertence	23
μ	Grau de pertinência	24
$\mu_A(x)$	Grau de pertinência ou Grau de pertinência primário	21
$\mu_{\tilde{\mathrm{A}}}(x,\mu)$	Grau de pertinência secundário	22
$\mu_{\tilde{\mathrm{A}}}(x=x^{'},\mu)$	Amplitude da função de pertinência do tipo-2	23
$\overline{\mu}_{\tilde{A}}$	Função de pertinência do tipo-2 superior	23
$\overline{f^i}$	Funções de pertinência superior	29
$\overline{f}^n$	Funções de pertinência inferior	40
$\tilde{A}_2$	Conjunto *fuzzy* tipo - 2 $\tilde{A}_2$	29
$\tilde{B}_1$	Conjunto *fuzzy* tipo - 2 $\tilde{B}_1$	29
$\tilde{B}_2$	Conjunto *fuzzy* tipo - 2 $\tilde{B}_2$	29
$\tilde{C}_1^{'}$	Conjunto *fuzzy* tipo - 2 $\tilde{C}_1^{'}$	29
$\tilde{A}_1$	Conjunto *fuzzy* tipo - 2 $\tilde{A}_1$	29
$\tilde{C}_2^{'}$	Conjunto *fuzzy* tipo - 2 $\tilde{C}_2^{'}$	29
$\underline{\mu}_{\tilde{A}}$	Função de pertinência do tipo-2 inferior	23

$\underline{f}^i$	Funções de pertinência inferior	29
$\underline{f}^n$	Funções de pertinência supeior	40
a	Parâmetro da base do trapézio	21
b	Parâmetro do topo do trapézio	21
c	Parâmetro do topo do trapézio	21
d	Parâmetro da base do trapézio	21
n	Elemento n	40
x	Elemento x	21
$x(J_x)$	Pertinência primária de x	23
$x^{'} = 2$	x' igual a 2	9
x_1	Primeiro elemento de x	24
x_2	Segundo elemento de x	24
x_i	Posição da altura máxima da função de pertinência da i - ésima função de pertinência	29
y	Elemento y	26
yk_n	Posição do centro de cada função de pertinência "n" no universo de discurso	40
z	Saída real z	27
$x = x^{'}$	x igual a x'	23

Sumário

1 CONSIDERAÇÕES INICIAIS

1.1 Contextualização

A lógica *fuzzy* tipo-1 surgiu em 1965, desenvolvida por Lofti A. Zadeh com a apresentação do conceito de conjuntos *fuzzy* [1]. Em 1973, Zadeh propôs conjuntos *fuzzy* representando variáveis linguísticas para modelar fenômenos antes descritos de maneira qualitativa [2]. Trata-se de uma técnica de inteligência computacional que permite trabalhar com diferentes níveis de incerteza. Segundo este procedimento, nada pode ser afirmado como sendo inteiramente certo ou inteiramente errado [3, 4].

Em 1975, Mamdani implementou um controlador através da lógica *fuzzy* proposta por Zadeh para um pequeno motor a vapor [5]. Com o seu controlador, demonstrou a capacidade de transformar em algoritmo uma solução expressa linguisticamente para o problema. A partir desse momento, a criação de controladores *fuzzy* vem crescendo constantemente com desenvolvimento de diversas aplicações como sistemas de diagnóstico para monitoramento de saúde, controle de nitidez de imagem para câmeras de foco automático, controle para sistemas servomecânicos, e controlador para manipulador robótico [6].

Estes controladores *fuzzy* tipo-1 são baseados em um sistema de inferência *fuzzy* SIF tipo-1 para descrever certos padrões a partir de quatro componentes principais: Fuzzificador, que tem a função de transformar os valores escalares das variáveis de entrada em valores linguísticos; Base de regras, que compreende o comportamento do processo proposto por um especialista para descrever um conjunto de regras linguísticas, na forma *se* (um conjunto de condições são satisfeitas) e *então* (um conjunto de consequências pode ser concluído); Método de Inferência que é usado para mapear determinada entrada *fuzzy* para uma dada saída *fuzzy*, e por último o Defuzzificador que converte a saída *fuzzy* do bloco de inferência em um número real [7].

Entretanto, mesmo com o uso da lógica *fuzzy* tipo-1 em certas aplicações de controladores *fuzzy*, ainda existem muitos problemas em que a lógica do tipo-1 não é capaz de solucionar. Dessa forma, foram desenvolvidas novas técnicas para melhorar a precisão ou desempenho da incerteza, resultando na lógica *fuzzy* tipo-2 que é uma extensão da lógica *fuzzy* tipo-1 e possui a capacidade de modelar níveis mais altos de incerteza [8].

A lógica *fuzzy* tipo-2 foi apresentada em 1975 por Zadeh [9], através de conjuntos *fuzzy* tipo-2 e são constituídos por funções de pertinência MFs (*Membership Functions*) propostas para sistemas de controle [10]. Novos conceitos foram introduzidos por Mendel e Liang [11], permitindo a definição do conjunto *fuzzy* tipo-2 por uma função de pertinência superior e uma inferior. Estas duas funções podem ser representadas separadamente

por um conjunto de funções de pertinência do tipo-1, onde o intervalo entre elas define a mancha de incerteza FOU (*Footprint of Uncertainty*), utilizada para caracterizar um conjunto *fuzzy* tipo-2 [12].

No entanto, o elevado processamento computacional para modelar os conjuntos *fuzzy* tipo-2 dificultou as aplicações práticas. Somente a partir de 1998, a lógica *fuzzy* tipo-2 foi melhor compreendida com o trabalho de Karnik e Mendel [13]. Nesse estudo, foi detalhada a teoria do sistema de inferência *fuzzy* SIF tipo-2, que é fundamentada através do SIF tipo-1 com o acréscimo do circuito tipo redutor TR (*Type Reductor*) que tem por finalidade realizar o mapeamento de um conjunto *fuzzy* tipo-2 em um tipo-1, e assim procurar o conjunto *fuzzy* tipo-1 que melhor represente o conjunto *fuzzy* tipo-2 [14]. Algum tempo depois, Liang e Mendel desenvolveram a teoria do SIF tipo-2 intervalar [11]. O sistema de inferência *fuzzy* tipo-2 intervalar simplificou os cálculos das operações de entrada das funções de pertinência modeladas pelo SIF tipo-2 no intervalo estabelecido entre as funções [13].

Existem trabalhos sobre o sistema de inferência *fuzzy* tipo-2 intervalar, sistema *fuzzy* tipo-2 intervalar implementado em FPGA baseado no algoritmo de Nie-Tan [15], processadores *fuzzy* tipo-2 intervalar embutindo um controlador *fuzzy* tipo-2 intervalar de alta velocidade para uma planta real em um FPGA [16], implementação de hardware de um mecanismo de inferência para controle *fuzzy* tipo-2 intervalar em FPGA [17], construções de controladores lógicos tipo-2 intervalares para aplicações em tempo real [18], arquitetura para detecção de borda em tempo real baseada na lógica *fuzzy* tipo-2 intervalar [19] e arquitetura de hardware e implementação FPGA de um sistema *fuzzy* tipo-2 [14].

Neste trabalho, a arquitetura abordada consiste em cinco blocos: fuzzificador, inferência, base de regras, tipo redutor e defuzzificador. O valor aplicado à entrada do circuito, também conhecido como *crisp* é convertido em seu valor *fuzzy* correspondente. Nesta etapa, são atribuídos valores linguísticos às variáveis de entrada para que essas possam ser trabalhadas pelo sistema de inferência *fuzzy*.

O bloco fuzzificador, é responsável por duas entradas, cada uma possui três funções de pertinência do tipo-2 trapezoidais, sendo divididas em funções superiores e funções inferiores que formam a FOU. A configuração da quantidade do número das funções em cada entrada ocorre no bloco de fuzzificação. A base de regras identifica as regras ativas e transfere essa informação para o bloco de inferência, que realiza as operações entre as funções antecedentes e consequentes. As funções consequentes são direcionadas ao bloco tipo redutor que tem o objetivo de procurar o melhor conjunto *fuzzy* tipo-1 que represente o conjunto *fuzzy* tipo-2 e o defuzzificador aplica o algoritmo de Nie-Tan para calcular o valor de saída real também conhecida como *crisp*.

1.2 Motivação

Sistemas de inferência fuzzy implementados em hardware e com foco em aplicações portáteis (por exemplo dispositivos portáteis ou equipamentos biomédicos implantáveis) devem ter como característica velocidade de operação suficiente para trabalhar em tempo real, implementação simples ocupando pouca área e utilizando poucos recursos e, principalmente, baixo consumo de potência. Este trabalho consiste no desenvolvimento de um controlador *fuzzy* tipo-2, escolhido pelo fato de empregar o número de regras menor em relação a outras arquiteturas tipo-1 e uma função de pertinência trapezoidal de implementação mais simples que garante circuitos mais rápidos do que os adotados em outras soluções. A proposta apresentada por Maciel [15] sobre a lógica *fuzzy* tipo-2 é baseada em funções gaussianas que podem ser modeladas pela posição de subida e descida das funções, pelo desvio padrão ou utilizando diferentes valores para o máximo grau de ativação. A função tem formato de sino [15]. Visando aprimorar essa implementação e com o objetivo de criar uma nova funcionalidade e otimizar a implementação em hardware, é possível implementar um circuito fuzzificador capaz de gerar funções de pertinência trapezoidais. Esse tipo de função tem menor complexidade na implementação, mais facilidade de construção, ocupa menor área e seu uso torna o circuito mais simples [7, 16, 17].

1.3 Objetivos

1.3.1 Objetivo Geral

Projetar um circuito digital capaz de implementar um sistema de inferência *fuzzy* tipo-2 intervalar a ser utilizado como um controlador em uma aplicação portátil.

1.3.2 Objetivo Específicos

Investigar conceitos sobre a lógica *fuzzy* tipo-2 e implementar a função de pertinência trapezoidal para tornar o sistema mais simples. Elaborar regras para transformar uma informação lógica obtida por experiência em um formalismo matemático que permita implementar um algoritmo em software ou em hardware. Para validar o circuito proposto, o resultado obtido para a superfície de controle é comparado utilizando como métrica o erro em relação ao modelo teórico. A melhoria no sistema está na implementação das funções de pertinência trapezoidais que possuem forma linear com as vantagens já mencionadas.

1.4 Contribuições

O circuito proposto baseia-se na arquitetura apresentada em [15], dispondo de duas entradas de 8 bits com três funções de pertinência trapezoidais para cada uma, nove

regras e uma saída de 8 bits com três funções de pertinência. A utilização de funções de pertinência trapezoidais na etapa de fuzzificação garante uma implementação em hardware mais simples, com potencial para apresentar melhor desempenho e baixo consumo de potência, devido à menor quantidade de dispositivos sob chaveamento. A simplicidade da arquitetura também facilita uma potencial utilização em tempo real em uma estrutura de controle.

1.5 Organização da Dissertação

A dissertação é desenvolvida em cinco capítulos e inclui alguns apêndices. Cada capítulo se divide em sessões para melhor organização do conteúdo. O capítulo 2 apresenta a fundamentação teórica e a situação atual da lógica *fuzzy* tipo-2 e descreve-se o diagrama em blocos de um SIF tipo-2 intervalar. O capítulo 3 é dedicado à descrição da arquitetura SIF tipo-2 intervalar adotada. As técnicas e métodos serão exibidos com o projeto e a montagem do circuito, acompanhados dos códigos em Verilog SIF tipo-2 intervalar. No Capítulo 4 são apresentados os resultados obtidos da arquitetura em FPGA e comparados com os resultados do modelo implementado no Matlab® utilizando o Interval Type-2 *fuzzy* Logic Toolbox [20]. O Capítulo 5 é dedicado às conclusões e aos comentários mais relavantes sobre o trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Revisão Bibilográfica

A lógica sobre conjuntos com diferentes graus de pertinência desenvolveu-se a partir de 1920 com os trabalhos pioneiros de Jan Lukasiewicz (1878-1956) [21]. Em 1965, Zadeh inventou a lógica *fuzzy* tipo-1, combinando os conceitos da lógica clássica e os conjuntos de Lukasiewicz. Em sua proposta, definiu graus de pertinência para suprir as necessidades relativas a problemas industriais, biológicos ou químicos [22].

Entre 1970 e 1980, apareceram diversas aplicações industriais. Um exemplo foi o emprego dos conceitos *fuzzy* no controle de uma máquina a vapor em 1975 por Ebrahim Mandani [23]. No início da década de 1980, pesquisadores japoneses implementaram as primeiras aplicações para o tratamento de água [7]. O projeto foi executado pela empresa Fuji Electric. Em 1987, Hitachi implementou um sistema automático de controle de operação de trem baseado na lógica *fuzzy*, em um sistema de metrô [24]. O crescimento das aplicações em processos industriais intensificou-se nos Estados Unidos a partir do ínicio da década de 1990 [7].

Após o primeiro trabalho de Mandani, em 1975 Zadeh apresentou o conceito sobre lógica *fuzzy* tipo-2 [9]. Consistiu em uma extensão da lógica *fuzzy* tradicional incentivada pela insuficiência dos primeiros processos em modelar as incertezas inerentes à definição das funções de pertinência [25]. Em 1998, Karnik e Mendel apresentaram conjuntos em situações em que há incerteza a respeito dos graus de pertinência, incerteza do formato das funções de pertinência ou em alguns parâmetros das funções de pertinência [26]. Em 1999, a mesma equipe definiu conjuntos *fuzzy* do tipo-2 cujos graus de pertinência são conjuntos *fuzzy* do tipo 1 e não mais um único valor [27].

A exigência de um elevado processamento computacional para modelagem desses conjuntos *fuzzy* tipo-2 dificultou as suas aplicações. Trabalhos mais aprofundados contribuíram na elaboração do modelo de SIF tipo-2 [26][28]. Em 2000, Liang e Mendel desenvolveram a teoria do SIF tipo-2 intervalar que simplificou os cálculos das operações de entrada das funções de pertinência. Utilizaram o conceito de funções de pertinência, superior e inferior e assim a incerteza pôde ser modelada pelo sistema no intervalo entre as funções [11].

2.2 Lógica *fuzzy* tipo-1

A lógica *fuzzy* tipo-1 foi desenvolvida por Zadeh e pode ser considerada como um conjunto de princípios matemáticos para a representação do conhecimento. Baseia-se no grau de pertinência dos termos, identificados como graus de verdade ou grau de pertinência primário. O grau de pertinência é definido conforme a expressão (2.1),

$$\mu_A = \begin{cases} 1 & x \in A \\ 0 & x \notin A \end{cases}, \tag{2.1}$$

onde A é um conjunto *fuzzy*, no universo de discurso X, que mostra o intervalo de pertinência entre 0 e 1, onde $\mu_A(x)$ é uma função de pertinência definida como um mapeamento, com graus de pertinência entre 0 e 1. O valor $\mu_A(x) = 0$ significa que um elemento x do universo X não pertence a determinado conjunto e $\mu_A(x) = 1$ significa o contrário [28] [29]. Costuma-se representar estas afirmações na forma.

$$A = \{x, \mu_A(x) | x \in X\}, \tag{2.2}$$

O conjunto *fuzzy* A relaciona-se com variáveis linguísticas interpretadas de maneira qualitativa, como alto, médio, baixo, quente, morno e frio, entre outros. Um exemplo é um processo de controle de temperatura em que as identificações baixas, média, e alta são definidas por conjuntos *fuzzy* que podem assumir diferentes graus de pertinência [9].

As funções de pertinência mais utilizadas possuem forma linear trapezoidal por causa de sua facilidade de construção [7][16][17]. A Figura 1 exibe uma função de pertinência do tipo-1 trapezoidal, sendo descrita pela equação (2.3),

$$\mu_(x) = \begin{cases} 0 & if\ x \leq a \\ \frac{x-b}{b-a} & if\ a < x \leq b \\ 1 & if\ b < x \leq c \\ \frac{d-x}{d-c} & if\ c < x \leq b \\ 0 & if\ x > d \end{cases}, \tag{2.3}$$

em que a, b, c e d são parâmetros a serem escolhidos na definição da função de pertinência. A função trapezoidal tem a forma de um triângulo truncado, sendo simétrico ou assimétrico. Os termos a e d são os valores de x correspondentes à base do trapézio, b e c são correspondentes ao topo, representando segmentos de reta que compõem a função trapezoidal [12].

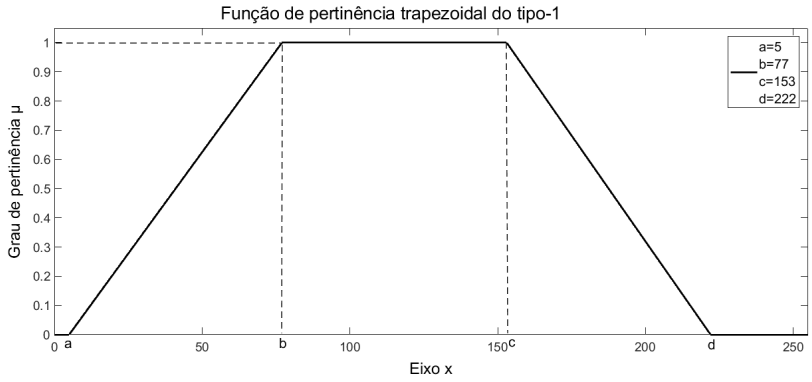

Figura 1 – Função de pertinência trapezoidal do tipo-1.

2.3 Lógica fuzzy tipo-2

A lógica *fuzzy* tipo-2 é uma extensão da lógica *fuzzy* tipo-1, representada por valores mapeados no universo de discurso dos conjuntos *fuzzy* tipo-1, por meio da função de pertinência. Tal lógica pode ser usada em situações em que existe incerteza sobre os graus de pertinência, incerteza do formato das funções de pertinência ou incerteza em alguns dos parâmetros das funções de pertinência [26].

A representação da lógica *fuzzy* tipo-2 com tais características é dada por um conjunto *fuzzy* tipo-2 A no universo de discurso X. Na função de pertinência do tipo-2 é definido um corte vertical $\mu_{\tilde{A}}(x, \mu)$, também chamado de grau de pertinência secundário, como apresentado na Figura 2.

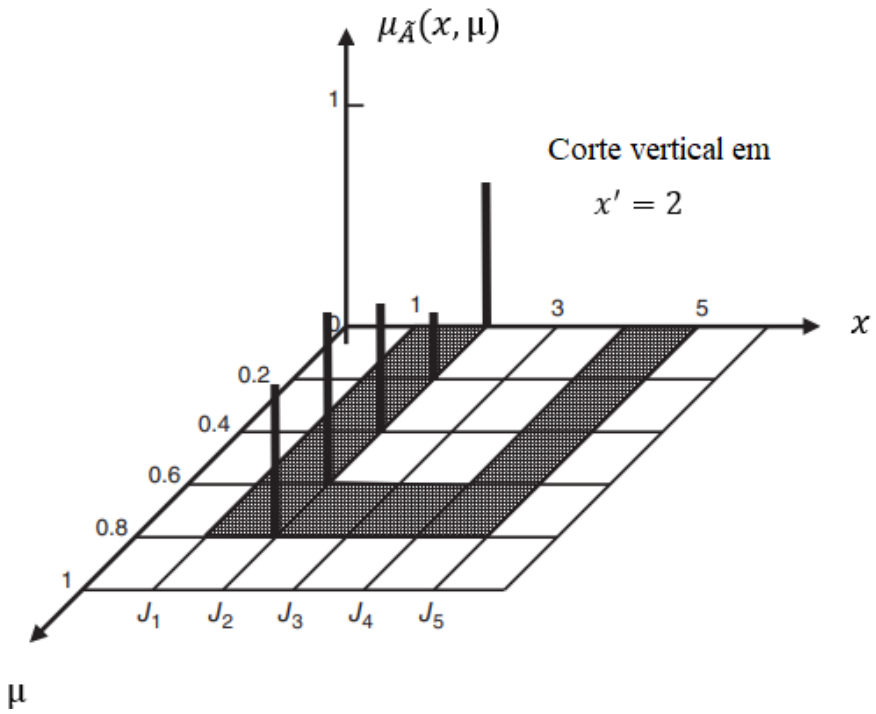

Figura 2 – Funções de pertinência do tipo-2, corte vertical distribuído para $x^{'} = 2$ [7].

Nessa figura, certo valor de $x = x^{'}$, para cada ponto $x \in X$, o grau de pertinência $\mu_{\tilde{A}}(x = x^{'}, \mu)$ é a amplitude da função de pertinência do tipo-2. Esta variável assumirá valores entre 0 e 1, com as propriedades resumidas na equação (2.4)

$$A = \{((x, \mu), \mu_{\tilde{A}}(x)) \mid \forall\, x \in X, \forall\, \mu \in J_x \subseteq [0, 1]\}\,, \tag{2.4}$$

na qual a pertinência primária de $x(J_x)$ corresponde ao domínio da função de pertinência secundária para um valor de x com $J_x \subseteq [0, 1]$ [7].

A função de pertinência do tipo-2 compreende uma função de pertinência do tipo-1 superior e outra inferior UMF e LMF. O intervalo entre as funções UMF e LMF formam a área chamada de mancha de incerteza FOU [30]. Quando a incerteza desaparece, o conjunto *fuzzy* tipo-2 torna-se um conjunto *fuzzy* tipo-1. As funções $\overline{\mu}_{\tilde{A}}$ e $\underline{\mu}_{\tilde{A}}$ são definidas por,

$$\overline{\mu}_{\tilde{A}} = UMF(\tilde{A}) = sup\,\{\mu | \mu \in [0, 1], \mu_{\tilde{A}}(x, \mu) > 0 \mid \forall\, x \in X\}\,, \tag{2.5}$$

$$\underline{\mu}_{\tilde{A}} = LMF(\tilde{A}) = inf\,\{\mu | \mu \in [0, 1], \mu_{\tilde{A}}(x, \mu) > 0 \mid \forall\, x \in X\}\,, \tag{2.6}$$

onde $\overline{\mu}_{\tilde{A}} = UMF(\tilde{A})$ e $\underline{\mu}_{\tilde{A}} = LMF(\tilde{A})$ representam os limites superior e inferior de uma função de pertinência do tipo-1. O valor $\overline{\mu}_{\tilde{A}}$ para $\forall\, x \in X$ é associado à função de pertinência superior do tipo-1 e $\underline{\mu}_{\tilde{A}}$ para $\forall\, x \in X$ à função inferior. Este conjunto de propriedades delimitam a FOU. A Figura 3. ilustra a área de incerteza, as funções de pertinência inferior e superior.

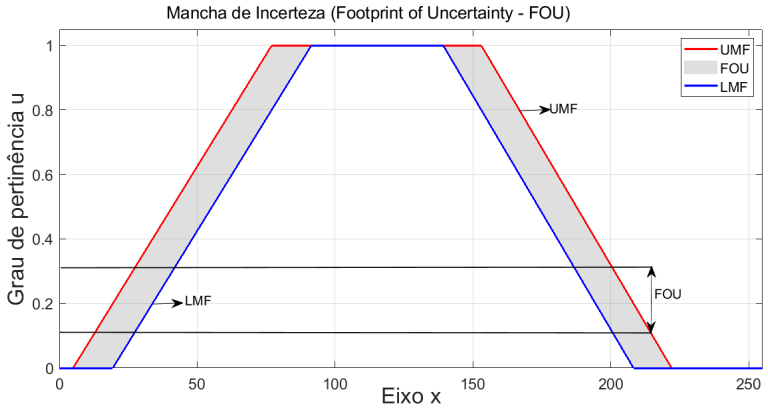

Figura 3 – Representação da mancha de incerteza, constituída pelas funções de pertinência superior UMF e inferior LMF.

A função de pertinência tipo-2 possui a amplitude definida como grau de pertinência secundário. Quando este for igual a 1 dentro da mancha de incerteza, $\mu_{\tilde{A}}(x, \mu) =$

1, $\forall\ x \in X$ e $\forall\ \mu \in J_x$, entre a UMF e LMF, não há prejuízos à lógica *fuzzy* tipo-2. Isto dá origem ao conjunto *fuzzy* tipo-2 intervalar [31]. As funções de pertinências do tipo-2 intervalares refletem uma incerteza uniforme sobre a pertinência primária de x e são as funções mais comumente usadas em sistemas *fuzzy* tipo 2 [32]. A Figura 4. mostra funções de pertinência do tipo-2 intervalar.

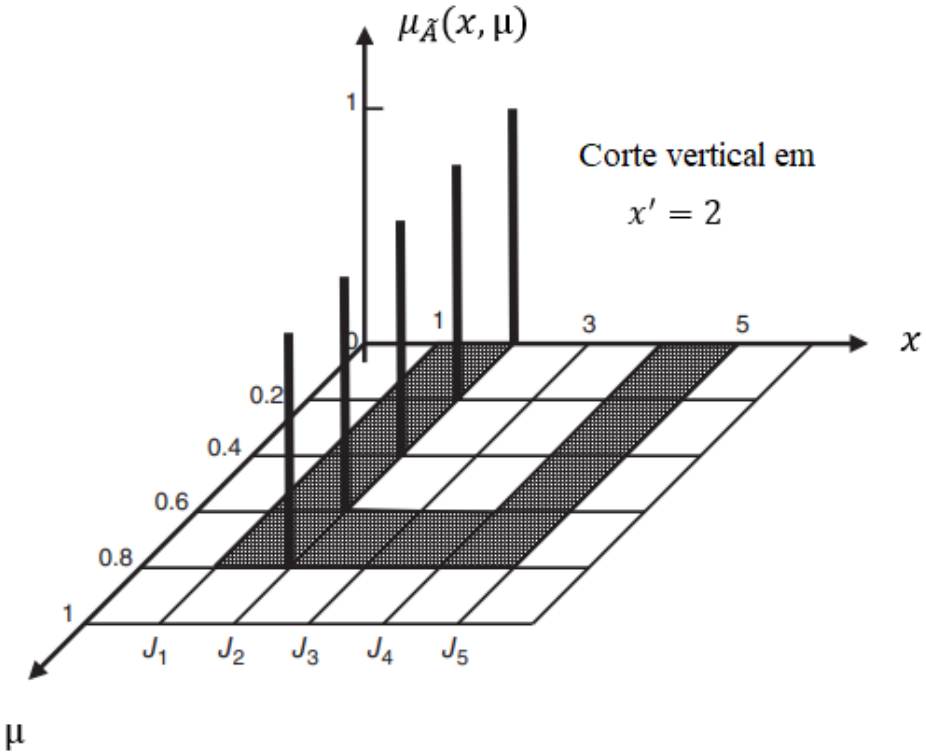

Figura 4 – Funções de pertinência tipo-2, corte vertical uniforme para $x^{'} = 2$ [7].

A seguir a região cinza da Figura 5 representa a FOU de um conjunto *fuzzy* tipo-2 intervalar.

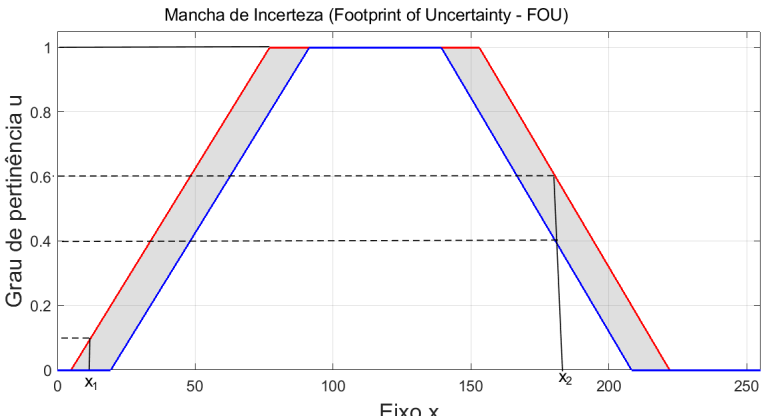

Figura 5 – Conjunto fuzzy tipo-2 intervalar.

O grau de pertinência primário de x_1 é um intervalo de valores de 0 a 0,1. Cada elemento deste intervalo possui grau de pertinência secundário igual a 1. Para x_2, as

pertinências primárias pertencem ao intervalo $[0,4; 0,6]$ e, da mesma forma, cada um destes valores possui pertinência igual a um. Na Figura 6 (a) e (b) são ilustradas as funções de pertinência secundárias referentes a x_1 e x_2.

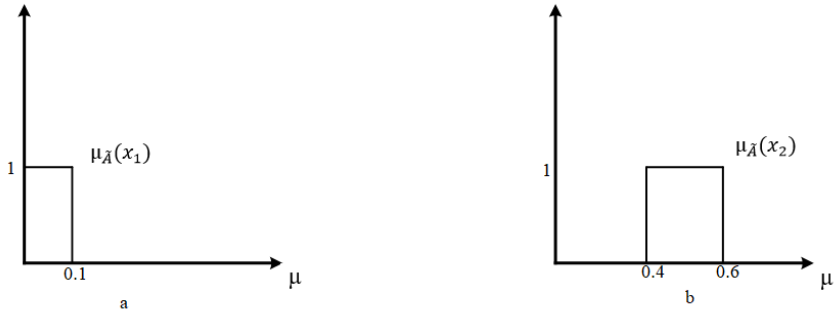

Figura 6 – Funções de pertinência secundária para o conjunto *fuzzy* tipo - 2 intervalar.

A lógica *fuzzy* tipo-2 modela controladores *fuzzy* em que os conjuntos *fuzzy* tipo-2 determinam as funções UMF e LMF com grau de pertinência no qual o intervalo demonstra a margem de erro FOU associada à incerteza. Por este motivo, o controlador *fuzzy* tipo-2 apresenta melhor desempenho comparado ao controlador *fuzzy* tipo-1. Por estas propriedades, este tipo de controlador tem a habilidade de calcular o erro da saída do circuito em relação ao valor teórico previsto [30][31].

2.4 Controlador *fuzzy*

O controlador *fuzzy* fornece um novo modelo para sistemas não lineares que permite executar um conjunto de ações a partir de dados de entrada. Esse dispositivo utiliza essas informações de maneira qualitativa e quantitativa com a definição de regras e comandos [7]. Emprega variáveis linguísticas desenvolvida por Zadeh, como estratégia de controle. Utiliza os conjuntos *fuzzy* tipo-1 e tipo-2 para atribuir um grau de pertinência entre 0 e 1 a diferentes conceitos subjetivos, como muito jovem, jovem e velho ou frio, morno, quente, etc. [2][33].

O controlador *fuzzy* oferece diversos benefícios em relação a outros operadores. Entre suas características vantajosas, pode operar com ruídos de diversas naturezas, é mais barato do que a criação de outro controlador com a mesma funcionalidade, são adaptáveis a diferentes especificações. Adiciona-se a estas características a maior facilidade de entender e modificar suas regras, que utilizam estratégias de um operador humano expressas em termos linguísticos naturais. A maneira de operar, projetar e aplicar um controlador *fuzzy* em um sistema real é mais fácil do que em outros sistemas [19].

2.4.1 Controlador *fuzzy* Tipo-1

O controlador *fuzzy* tipo-1 desenvolvido por meio da lógica *fuzzy* tornou-se um sucesso para tecnologias mais sofisticadas, permitindo a implementação de parâmetros complexos em controladores [7]. O sistema de controle *fuzzy* tipo-1 apresentado na Figura 7, é composto por quatro componentes principais: fuzzificador, base de regras, inferência e defuzzificador [28].

O bloco fuzzificador mapeia os pontos do universo de discurso X, em um conjunto *fuzzy*. Este processo obtém o grau de pertinência com que cada entrada pertencente a determinado conjunto, e atribui valores linguísticos, definidos por funções de pertinência associadas aos valores de entrada [28][34]. A equação (2.3) exibe um exemplo de expressão matemática para a função de pertinência trapezoidal do tipo-1. Através deste mapeamento ocorre a ativação das regras.

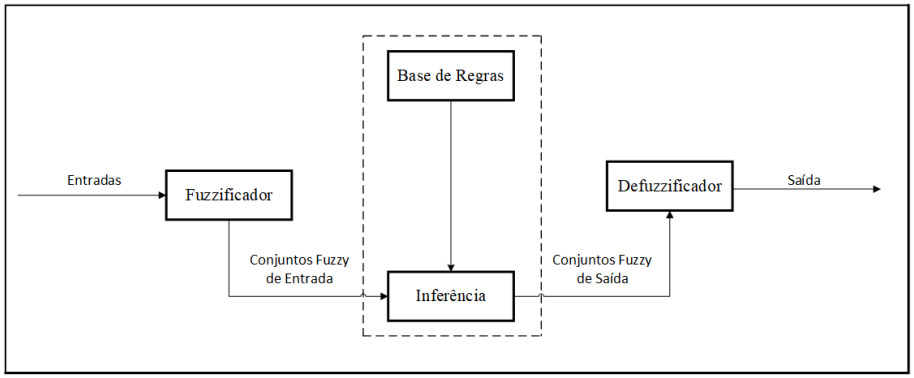

Figura 7 – Sistema de controle *fuzzy* tipo-1 [13].

A base de regras é definida por um especialista em forma de sentenças linguísticas para resolver determinado problema. Este bloco descreve relações entre as variáveis linguísticas que são processadas pela inferência. A base de regras *fuzzy se-então* assume a forma [35]:

$$\textbf{Se } x \text{ é } A \textbf{ Então } y \text{ é } B \tag{2.7}$$

onde A e B são valores linguísticos definidos por um conjunto *fuzzy* no universo de discurso X e Y, respectivamente. Em geral, x é chamado de antecedente e y é chamado de consequente. Exemplos de regras *fuzzy* comuns em expressões linguísticas do dia a dia são [35]:

- **Se** a **estrada** estiver **escorregadia**, **então dirigir** é **perigoso**.
- **Se** a **velocidade** for **alta**, **então** use um **pouco** do **freio**.

O bloco de inferência calcula quais regras da base de regras são disparadas e quais são seus graus de pertinência, usando os conjuntos *fuzzy* fornecidos a ele pelo fuzzificador.

O mecanismo de inferência pode também combinar o grau de pertinência de cada regra com o conjunto *fuzzy* consequente dessa regra para produzir o conjunto *fuzzy* de saída da regra, e em seguida combinar todos esses conjuntos em todas as regras disparadas para fornecer um conjunto de saída *fuzzy* agregado usando a matemática dos conjuntos *fuzzy*. De outro modo, ele pode também enviar o grau de pertinência de cada regra diretamente para o defuzzificador, onde todos são agregados de uma maneira diferente [29].

Existem diversos procedimentos para executar as operações de inferência, proporcionando mapeamentos dos conjuntos antecedentes para os consequentes. Neste trabalho é adotado o método de inferência de Mamdani, que tem o intuito de representar experiências da vida real, através do desenvolvimento de sistemas *fuzzy* baseando-se em regras de conjuntos *fuzzy*. *Se* "antecedente" *Então* "consequente" é determinado pelo produto cartesiano *fuzzy* dos conjuntos *fuzzy* que constituem o antecedente e o consequente da regra. No método de Mandani as regras são agregadas através do operador OU, que é modelado pelo operador máximo, e em cada regra, o operador lógico E é modelado pelo operador mínimo. As regras a seguir apresentam o método de Mandani [35],

$$Regra\ 1 : Se\ x \text{ é } A_1\ e\ y \text{ é } B_1\ Então\ z \text{ é } C_1, \tag{2.8}$$

$$Regra\ 2 : Se\ x \text{ é } A_2\ e\ y \text{ é } B_2\ Então\ z \text{ é } C_2, \tag{2.9}$$

A Figura 8 apresenta um exemplo do processo de inferência de Mamdani, com uma saída real z gerada a partir das entradas e reais e a regra de composição max - min. Onde o menor grau de pertinência é determinado pelo min entre as funções antecedentes através da operação de interseção, $A_1 \cap B_1$ e $A_2 \cap B_2$, e o maior grau de pertinência max entre as funções consequentes é determinado pela operação de união $C_1' \cup C_2'$. A saída z pertence aos números reais, e é obtida pela defuzzificação do conjunto de saída $C_1' \cup C_2'$ [3].

A defuzzificação gera uma única saída real, através dos conjuntos *fuzzy* resultante da saída do mecanismo de inferência. A técnica de defuzzificação que será utilizada nesta dissertação é a da altura. O método da altura também é conhecido como centro dos máximos e é uma opção de cálculo mais simples por causa do menor número de operações [36].

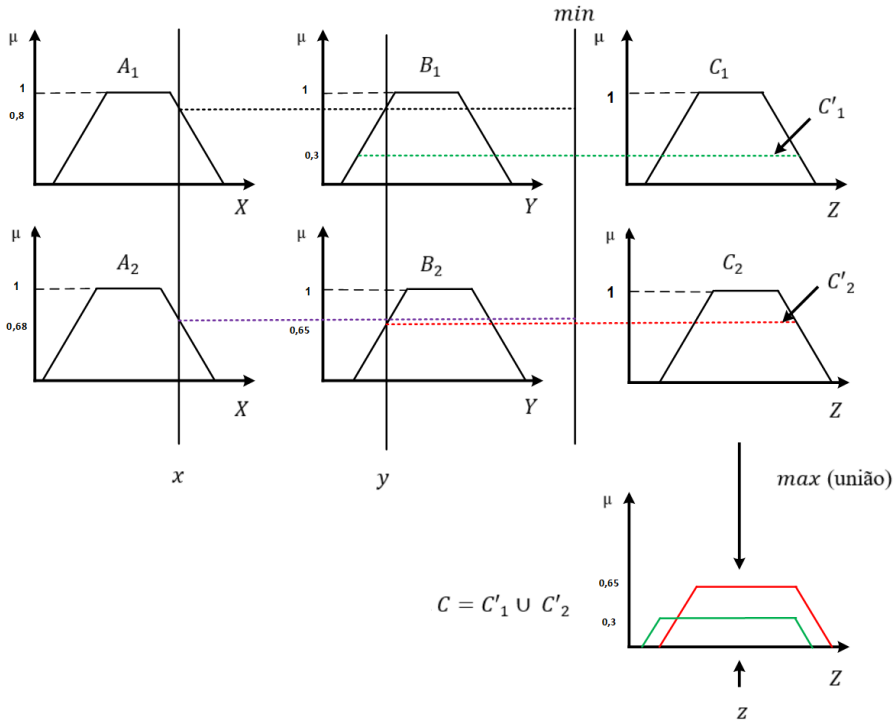

Figura 8 – Processo de inferência de Mamdani, para controlador fuzzy tipo-1 [37].

2.4.2 Controlador *fuzzy* Tipo-2

O controlador *fuzzy* tipo-2 é semelhante ao controlador *fuzzy* tipo-1, a diferença predominante é a adição do bloco tipo-redutor que coleta todas as saídas *fuzzy* tipo-2, aplicando um método de redução que permite converter o conjunto *fuzzy* tipo-2 em conjunto *fuzzy* tipo-1. Por fim, o conjunto reduzido é direcionado ao defuzzificador que então gera o valor crisp de saída [21]. O diagrama em blocos do sistema de inferência *fuzzy* tipo-2, é abordado na Figura 9, este sistema é composto por cinco blocos: o fuzzificador, base de regras, inferência, tipo-redutor e o defuzzificador.

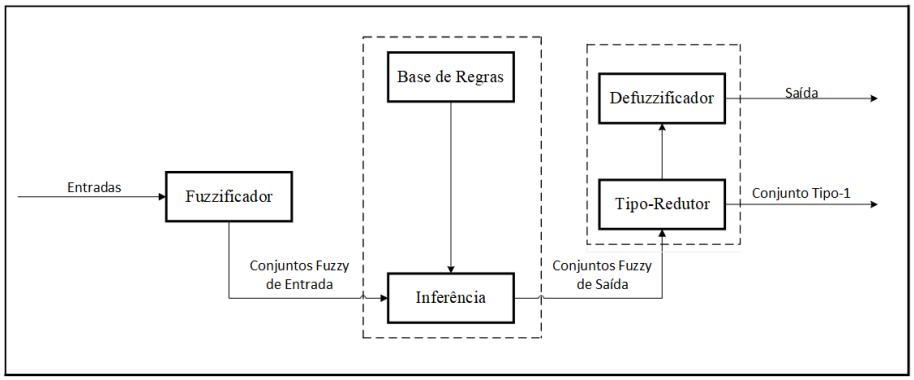

Figura 9 – Sistema de controle *fuzzy* tipo-2 [13].

O fuzzificador recebe os valores reais de entrada e transforma-os em um conjunto *fuzzy* tipo-2, através da associação entre a entrada e os graus de pertinência formado pelas funções de pertinência de entrada. Nesse caso para os conjuntos *fuzzy* tipo-2 intervalar o

grau de pertinência de cada entrada representada pelo universo de discurso X, é constituída pela mancha de incerteza que é formada por duas funções de pertinência superior e inferior.

A base regras do controlador *fuzzy* tipo-2 tem a mesma forma do controlador *fuzzy* tipo-1, tratado na seção 2.4.1. A mudança está na natureza das funções de pertinência. Como por exemplo,

$$Regra\ 1 : Se\ x \text{ é } \tilde{A}_1\ e\ y \text{ é } \tilde{B}_1\ Então\ z \text{ é } \tilde{C}_1, \tag{2.10}$$

$$Regra\ 2 : Se\ x \text{ é } \tilde{A}_2\ e\ y \text{ é } \tilde{B}_2\ Então\ z \text{ é } \tilde{C}_2, \tag{2.11}$$

O mecanismo de inferência estabelece a combinação e o cálculo das regras, por meio do mapeamento dos conjuntos antecedentes *fuzzy* para os conjuntos consequentes *fuzzy*. O modelo de inferência utilizado neste trabalho é o de Mamdani, exibido na Figura 10. O menor grau de pertinência é determinado pelo min entre as funções antecedentes através da operação de interseção, $\tilde{A}_1 \cap \tilde{B}_1$ e $\tilde{A}_2 \cap \tilde{B}_2$, e o grau de pertinência *max* entre as funções consequentes é determinado pela operação de união $\tilde{C}'_1 \cup \tilde{C}'_2$. A saída z pertence aos números reais, e é obtida pela defuzzificação do conjunto de saída $\tilde{C}'_1 \cup \tilde{C}'_2$.

O tipo-redutor tem como objetivo a transformação de um conjunto *fuzzy* tipo-2 em conjunto *fuzzy* tipo-1. Os métodos mais utilizados são, Karnik-Mendel (KM) onde o tipo-redutor é um algoritmo em que a iteração visa alcançar o menor e o maior centroide dos conjuntos *fuzzy* tipo-1 agregados e considera-se a média de ambos os valores como a saída crisp e defuzzificada [37]. No método Wu-Mendel (WM) os limites de incerteza min-max definidos pelo tipo-redutor, reduzem os conjuntos *fuzzy* tipo-2 em tipo-1 e é utilizado no método da altura para realizar a defuzzificação [37]. O algoritmo de Nie-Tan (NT) usado neste trabalho para calcular o conjunto *fuzzy* tipo-1 através das funções consequentes provenientes da saída do bloco de inferência, é definido através da operação matemática de média entre os graus de pertinência superior e inferior das funções de pertinência, apresentado na Equação (12).

$$Y_{NT} = \frac{\sum_{i=1}^{M} y^i(\overline{f^i} + \underline{f^i})}{\sum_{i=1}^{M}(\overline{f^i} + \underline{f^i})} \tag{2.12}$$

Onde temos que, $\overline{f^i} + \underline{f^i}$ são as funções de pertinência superior e inferior, o Y_{NT} é o valor real crisp e x_i é a posição da altura máxima da função de pertinência da i-ésima função de pertinência [15].

A defuzzificação gera uma única saída real, através dos conjuntos *fuzzy* resultante da saída do mecanismo de inferência. A técnica de defuzzificação que será utilizada nesta

dissertação é a da altura. O método da altura também é conhecido como centro dos máximos e é uma opção de cálculo mais simples por causa do menor número de operações [37].

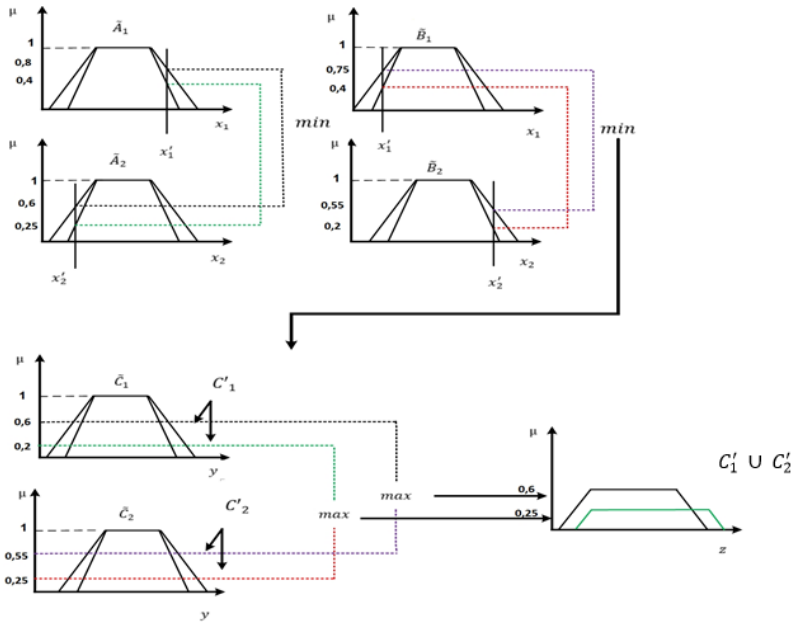

Figura 10 – Processo de inferência de Mamdani, para controlador *fuzzy* tipo-2.

2.5 Vantagens de *fuzzy* tipo-2 com relação ao tipo-1

As vantagens do *fuzzy* tipo-2 em relação ao tipo-1 estão associadas à capacidade do controlador *fuzzy* tipo-2 de lidar com as incertezas do sistema. As principais vantagens que podem ser citadas são [9] [38]:

- Incertezas linguísticas que podem ser modeladas pelo sistema *fuzzy* tipo-2 intervalar, enquanto o sistema *fuzzy* tipo-1 não é capaz de modelar;

- O uso das funções de pertinência *fuzzy* do tipo-2 para representar as entradas e saídas do controlador lógico *fuzzy* resulta na redução do número de regras da base de regras quando comparado com o sistema *fuzzy* tipo-1;

- As incertezas nas funções de pertinência de entrada e saída do controlador *fuzzy* podem ser trabalhadas com a função de pertinência *fuzzy* tipo-2 que contém a mancha de incerteza (FOU).

2.6 Considerações finais

Neste capítulo foram apresentados os conceitos sobre a lógica *fuzzy* tipo-1 e tipo-2, as estruturas dos controladores tipo-1 e tipo-2 intervalar e a vantagem do *fuzzy* tipo-2 em relação ao tipo-1. No próximo capítulo será apresentada a arquitetura proposta para implementar o controlador *fuzzy* tipo-2 intervalar em hardware digital e o projeto dos seus blocos constituintes.

3 Projeto do Controlador *Fuzzy* tipo-2 em FPGA

3.1 Introdução

Neste capítulo é apresentado o hardware digital implementado em FPGA para um Sistema de Inferência *Fuzzy* SIF tipo-2 intervalar. Utiliza-se a linguagem de descrição de hardware Verilog para descrever o circuito. A Figura 11 mostra o sistema constituído por duas entradas de 8 bits, cada uma associada a 3 funções de pertinência trapezoidais do tipo-2 que formam o fuzzificador tipo-2. O mecanismo de inferência é baseado no método de Mamdani que tem como entrada informações do grau de ativação das funções antecedentes, provenientes do bloco de fuzzificação. Cada uma das duas entradas está associada a três funções de pertinência do tipo-2 intervalares, sendo compostas pela função superior e inferior. Cada bloco fuzzificador possui um bit "Ativo_UP" para identificar a ativação da função. Esta informação é direcionada à base de regras que identifica as regras ativas e gerencia o bloco de inferência. A arquitetura pode operar com até 9 regras, dependendo da configuração estabelecida no bloco fuzzificador, e possui três funções consequentes na saída do bloco de inferência. Estas são enviadas para o tipo-redutor/defuzzificador baseado no algoritmo de Nie-Tan, que transforma a função trapezoidal do tipo-2 em um valor crisp de saída.

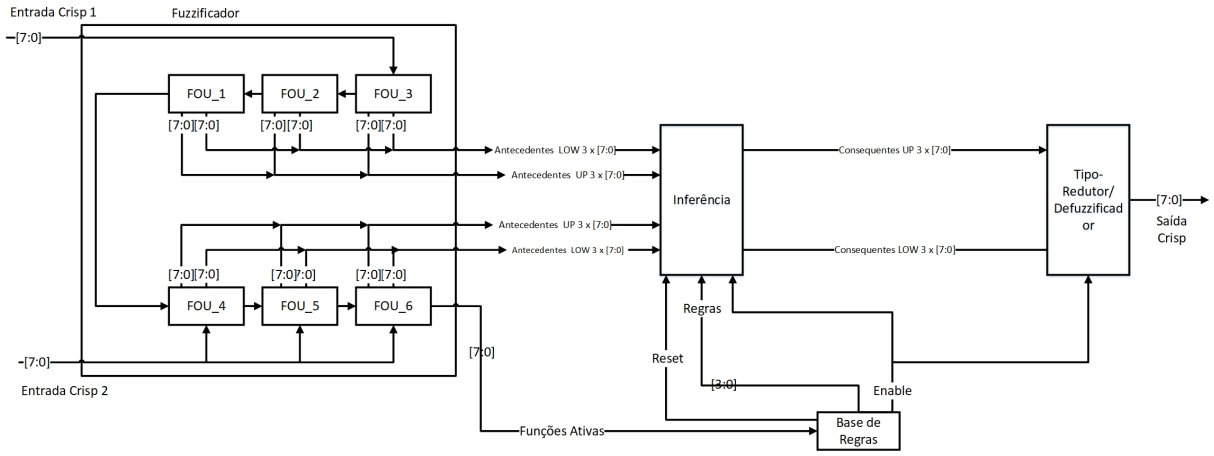

Figura 11 – Arquitetura do SIF tipo-2 intervalar.

3.1.1 Fuzzificador

O bloco do fuzzificador do tipo-2 é mostrado na Figura 12. Seus parâmetros para um SIF tipo-2 intervalar modelam os níveis superior e inferior da mancha de incerteza especificada por um especialista e que podem ser otimizadas no ajuste do controlador. A

Figura 13 mostra a forma trapezoidal, a partir da qual é possível gerar as demais formas, e os seus parâmetros de configuração de entrada A, B, C e D.

Para implementar a FOU são utilizadas duas funções de pertinência *fuzzy* tipo-1, uma que determina a parte superior do FOU e a outra a parte inferior. A Figura 13 apresenta o diagrama em RTL (*Register Transfer Level*) Viewer dos circuitos duplicados. A saída nomeada como "Ativo_UP" é destinada a identificar a ativação do FOU, informação utilizada pela base de regras para gerenciar as regras ativas. A identificação é feita por comparadores "geraSativo" conectados somente nos blocos das funções de pertinência superior. A justificativa é o fato de estas sempre possuírem um valor maior que a função inferior, o que é suficiente para detectar a ativação. Além disto, a função inferior pode ocorrer uma vez que a superior está ativa. Por este motivo, o comparador utiliza a saída da função de pertinência superior "sinal_UP" com relação ao valor zero. Se o valor for maior que zero tem-se para a saída "sinal_UP" a saída "Ativo_UP = 1".

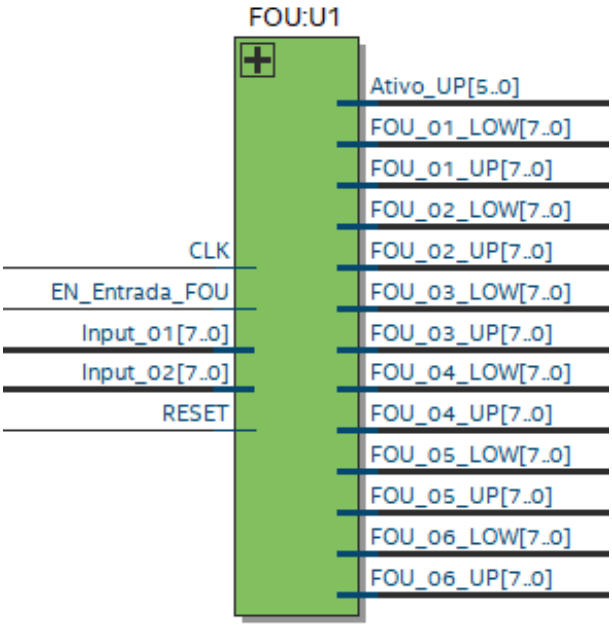

Figura 12 – Bloco Fuzzificador completo com as entradas e saídas definidas.

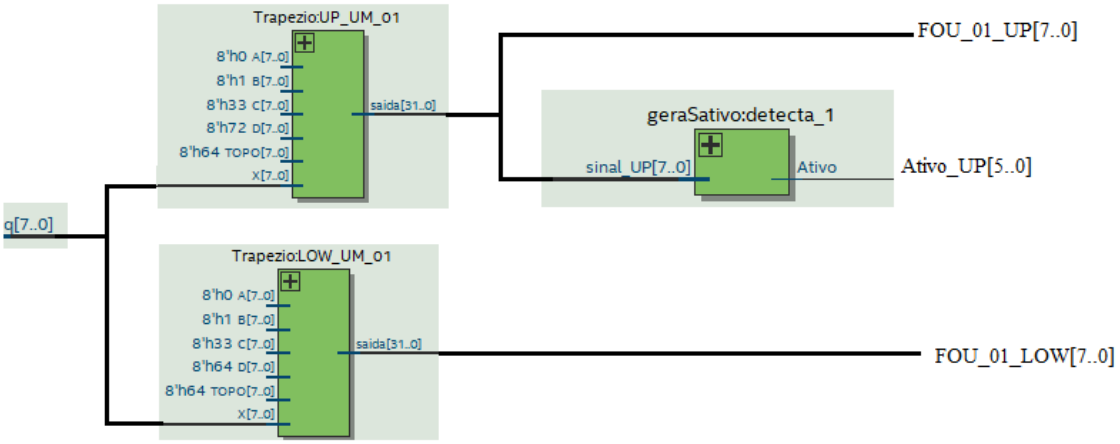

Figura 13 – RTL *Viewer* dos circuitos duplicados do FOU_1.

Os blocos que geram a função de pertinência trapezoidal do tipo-1 (que irá representar as funções superior ou inferior do conjunto tipo-2) são denominados:

- range_x_trapezio,
- Num_denom_trapezio,
- Numerador_Multiplicado_trapezio,
- Elementos_trapezio,
- Elemento_menor_trapezio,
- descarte_topo_trapezio.

O bloco range_x_trapezio identifica se a entrada X está entre os valores de A e D. O bloco Num_denom_trapezio gera os termos do numerador e do denominador correspondentes à equação 2.3. O bloco Numerador_Multiplicado_trapezio multiplica o Numerador_1 e Numerador_2 por 100. A multiplicação é necessária para o resultado final do grau de pertinência ser expresso em representação de valores inteiros entre 0 e 100 (correspondendo ao grau de pertinência entre 0 e 1). O bloco Elementos_trapezio realiza a divisão do Numerador_1 com o Denominador_1 e do Numerador_2 com o Denominador_2, resultando no Elemento_1 e no Elemento_2. Cada um dos elementos corresponde à rampa de subida e de descida do trapézio, então o bloco Elemento_menor_trapezio compara o Elemento_1 com o Elemento_2, seleciona o menor entre ambos, e o bloco descarte_topo_trapezio é utilizado para limitar a saída no valor máximo 100, formando assim a parte superior do trapézio.

3.1.2 Circuito de Inferência

O circuito de inferência opera com 9 regras e processa funções *fuzzy* tipo-2 intervalar. Utiliza dois circuitos, um para as funções de pertinência tipo-1 da parte superior do FOU e o outro para as mesmas funções da parte inferior do FOU, como na Figura 14. A parte interna das instâncias superior ou inferior do bloco de inferência é apresentada na Figura 15 pela entrada de dois multiplexadores de 4 canais, MUX1 e MUX2. O multiplexador MUX1 recebe as funções de pertinência tipo-1 referente aos FOU_1, FOU_2 e FOU_3. O multiplexador MUX2 recebe as funções de pertinência tipo-1 referente aos FOU_4, FOU_5 e FOU_6. Para realizar as operações de inferência, as entradas de seleção dos multiplexadores MUX1 e MUX2 são endereçadas com as regras ativas. As saídas do circuito de inferência utilizam três registradores de 8 bits para armazenar as funções consequentes, resultantes da operação de inferência.

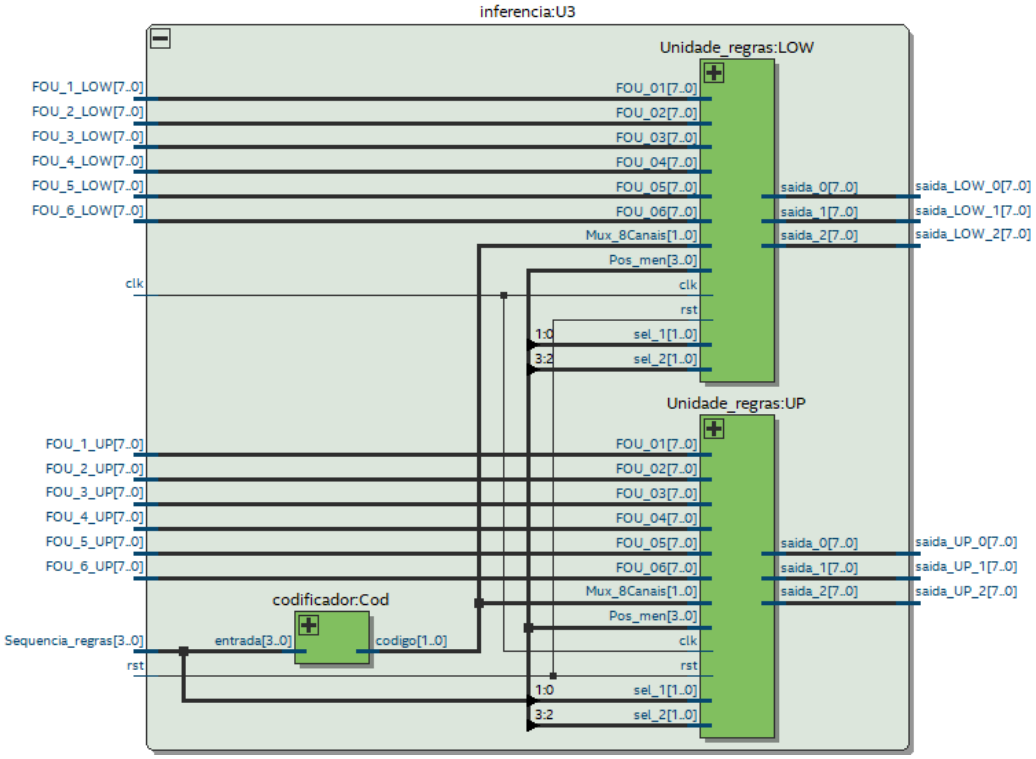

Figura 14 – O circuito de inferência.

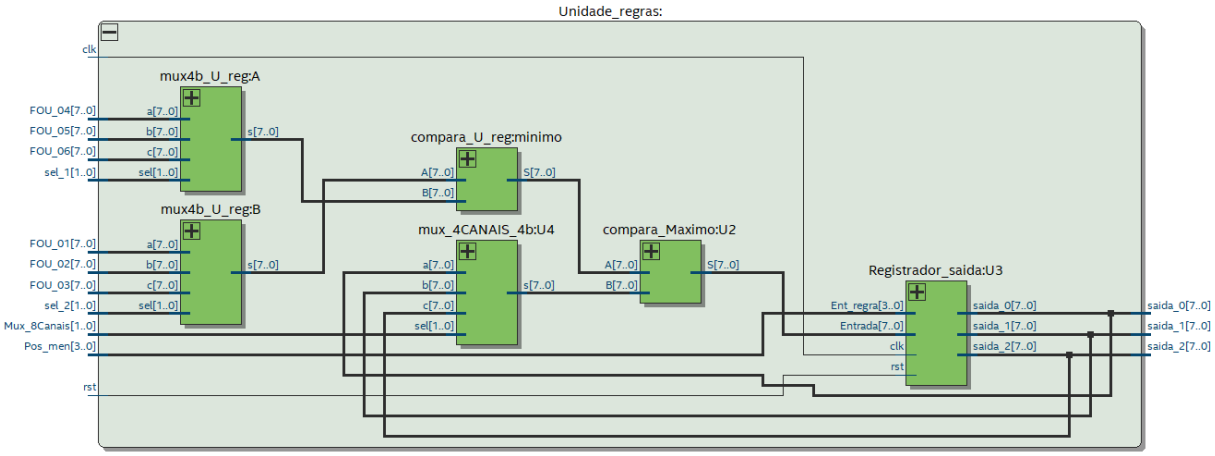

Figura 15 – Parte interna das instâncias superior ou inferior do bloco de inferência.

3.1.2.1 Circuito de Mínimo

O circuito de mínimo define o mínimo grau de pertinência entre duas funções selecionadas por uma regra. Possui dois blocos, um para função trapezoidal superior e outro para a inferior. Os multiplexadores utilizados em cada bloco possuem três entradas de dados de 8 bits e uma entrada de seleção de 2 bits. A sequência de regras ativas, de 4 bits, é a informação utilizada na entrada de seleção. Os multiplexadores que recebem as funções de pertinência de 1 a 3 empregam em sua entrada de seleção os bits de posição 0 e 1 da sequência de regras ativas. Os multiplexadores que recebem as funções de pertinência de 4 a 6 utilizam os bits de posição 2 e 3 da sequência de regras ativas. Um exemplo da operação de mínimo é apresentado na seção 2.4.2.

3.1.2.2 Circuito de Máximo

O circuito de máximo determina o maior grau de ativação entre as regras que possuem o mesmo consequente, para cada um dos três consequentes. A Figura 15 apresenta a determinação do valor de cada função consequente, armazenado em registradores de 8 bits. Como o mecanismo de inferência utiliza processamento sequencial, somente uma regra é processada a cada ciclo de clock. O grau de pertinência resultante da regra ativa é disponibilizado na entrada de todos os registradores, porém, somente o registrador referente à função consequente é habilitado para atualização.

O operador Máximo determina o maior valor entre o grau de pertinência da função consequente, armazenado no registrador de destino, e o grau de pertinência determinado pelo circuito mínimo para a regra ativa. O operador Máximo compara a relação do mínimo grau de pertinência das funções antecedentes, relacionadas à regra ativa, com o valor da função consequente. Caso a informação de saída do operador Mínimo seja maior que a informação armazenada na função consequente, o operador Máximo atualiza a informação da função consequente com o valor da saída do operador Mínimo. Caso contrário, a

informação da função consequente mantém-se com o mesmo valor.

O registrador apresentado na Figura 16 é selecionado na entrada do circuito Máximo e usa o multiplexador de 8 canais. A entrada de seleção é determinada pelo bloco codificador, que utiliza a informação de regra ativa para determinar o canal de entrada a ser atualizado na saída do multiplexador. Dependendo da regra ativa, o codificador recebe uma informação binária correspondente e ativa um dos canais do multiplexador. Este canal possui como entrada o registrador referente à função consequente da regra ativa.

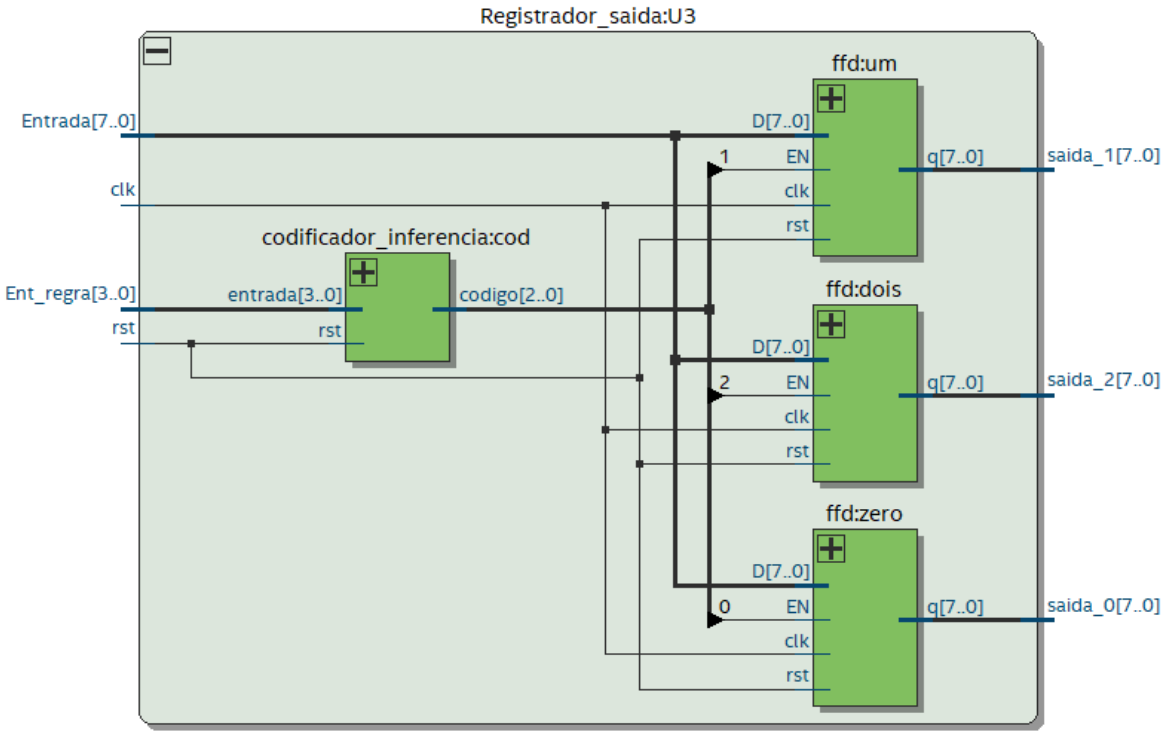

Figura 16 – Circuito Registrador que armazena o valor das Funções Consequentes.

3.1.3 Base de Regras

A base de regras implementa 9 regras, estabelecendo relações lógicas entre as funções de pertinência de entrada e as funções de saída para um SIF tipo-2. As regras podem ser especificadas ou extraídas de dados reais e cada regra é uma declaração *Se-Então*. A parte Se de uma regra é chamada antecedente e a parte Então é a consequente. Para o sistema implementado, cada entrada *Crisp* ativa apenas uma ou duas funções de pertinência. Portanto, apesar de existirem 9 regras, nem todas são ativas ao mesmo tempo. Caso apenas uma função de pertinência seja ativa em cada entrada, somente uma regra é selecionada. Caso uma entrada ative duas funções de pertinência e a outra ative apenas uma função de pertinência, duas regras são selecionadas. Se ambas as entradas ativarem duas funções de pertinência, são selecionadas quatro regras.

A base de regras implementada neste trabalho utiliza uma máquina de estado representada pelas Figuras 18 e 19 que tem como entrada um barramento "FOU_ativo" de 8 bits, uma entrada Enable "EN_REGRAS" e uma entrada Reset "rst". Cada bit do barramento "FOU_ativo" é dedicado a monitorar uma função de pertinência, em que "0" representa função inativa e "1" representa função ativa. As entradas Enable "EN_REGRAS" e Reset "rst" são chaves externas para habilitar e "resetar" o SIF tipo-2. As saídas "EN_Memórias", "Reset_Memória" e o barramento de 4 bits "Sequencia_regras" da máquina de estados (Figura 39) gerenciam o mecanismo de inferência e os registradores. A saída "EN_Memórias" é dedicada a habilitar os registradores, a saída "Reset_Memória" é dedicada a limpar os dados dos registradores e o barramento de saída de 4 bits "Sequencia_regras" é dedicado a sequenciar as regras ativas. A base de regras proposta gerencia 9 regras que estão relacionadas com suas respectivas funções antecedentes e consequentes apresentadas na Figura 17. Para identificar e gerenciar as regras ativas a máquina de estado é projetada com 11 estados como mostra a Figura 18. Um estado limpa as memórias, nove estados são dedicados a gerar o código das 9 regras e um estado habilita as memórias para atualização de valores.

A sequência lógica dos estados é apresentada a seguir:

- A primeira condição para iniciar o processo é analisar a entrada Reset. Caso o reset apresente nível lógico alto o estado atual é travado na condição de estado final (END), caso contrário o estado atual passa a ser o estado inicial (START).

- Estado inicial (START): é dedicado a limpar os valores calculados na inferência *fuzzy* anterior armazenados nas memórias.

- Estados (R0, R1, R2, R3, R4, R5, R6, R7 e R8) dedicados às regras: são responsáveis por gerar o código de cada regra ativa e analisar a possibilidade de existir uma próxima regra. Caso exista mais uma regra ativa, o estado posterior será o da regra identificada.

- Estado final (END): é o estado que finaliza a sequência de regras ativas, habilita as memórias para atualização e reinicia o processo com a verificação da entrada Reset. Caso a entrada Reset esteja em nível alto o estado final (END) é mantido, caso a entrada Reset esteja em nível baixo retorna para o estado inicial (START).

Entrada_01 \ Entrada_02	FOU_4	FOU_5	FOU_6
FOU_1	Regra 0	Regra 1	Regra 2
FOU_2	Regra 3	Regra 4	Regra 5
FOU_3	Regra 6	Regra 7	Regra 8

Figura 17 – Base de regras.

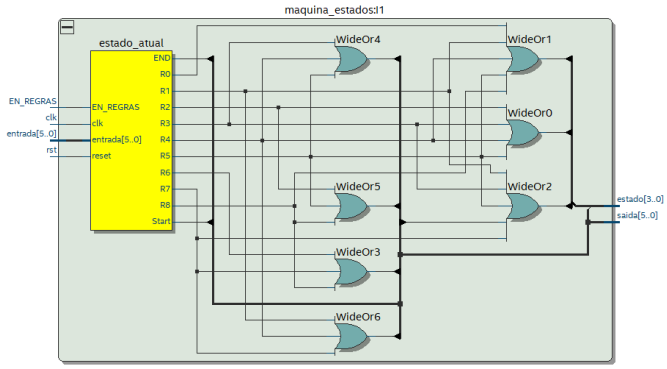

Figura 18 – Circuito da base de regras.

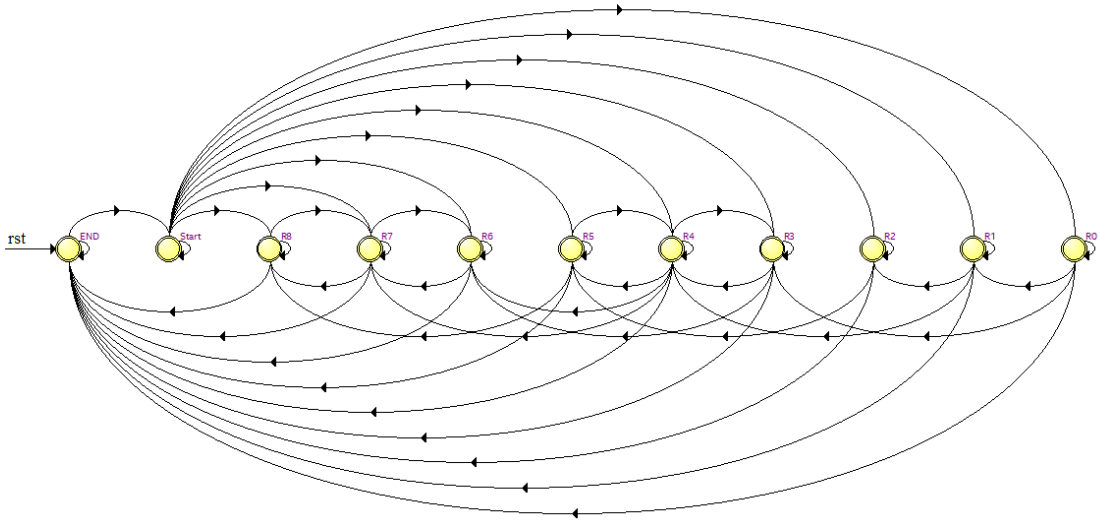

Figura 19 – Máquina de estado para base de regras.

3.1.4 Tipo-Redutor e Defuzzificador

O algoritmo Nie-Tan (NT), define a operação do tipo-redutor que é realizada através da média entre os graus de ativação superior e inferior das funções de pertinências. Onde $\underline{f}^n$ representa a função de pertinência superior e $\overline{f}^n$ representa a função de pertinência inferior. Esta operação é utilizada para calcular a saída *crisp* utilizando as funções consequentes de saída do bloco de inferência, seção 3.1.2. Desta forma a redução de tipo NT para n-ésima função de pertinência é definida na equação 3.1.

$$f_n = \frac{\underline{f}^n + \overline{f}^n}{2}, \tag{3.1}$$

As funções de pertinências tipo-1, obtidas pela operação de tipo-redução apresentada na equação 3.1, são utilizadas para determinar o valor de saída *crisp* utilizando o método de defuzzificação da Altura. Desta forma a equação (3.2) define a operação para determinação do valor *crisp* de saída pelo algoritmo de Nie-Tan. Onde Y_{NT} é o valor real *crisp* x_i é a posição da altura máxima da função de pertinência da i-ésima função de pertinência. Substituindo a equação (3.1) na (3.2) obtém a equação 2.12. A Figura 20 apresenta o tipo-redutor e o defuzzificador.

$$Y_{NT} = \frac{\sum_{i=1}^{N} x_i(\overline{f}^n + \underline{f}^n)}{\sum_{i=1}^{N}(\overline{f}^n + \underline{f}^n)}, \tag{3.2}$$

A equação (2.12) demonstra que o valor *crisp* pode ser obtido diretamente dos valores dos graus de ativação superiores e inferiores das funções de pertinência tipo-2. Essa característica dispensa a implementação de um circuito tipo-redutor dedicado, isso reduz o número de blocos do processamento de saída. O processamento de saída proposto neste trabalho opera com três funções de pertinência. A equação (3.3) apresenta a expansão da equação (3.2) onde a operação yk_n determina a posição do centro de cada função de pertinência "n" no universo de discurso.

$$Y_{NT} = y\frac{k_1(\underline{f}^1 + \overline{f}^1) + k_2(\underline{f}^2 + \overline{f}^2) + k_3(\underline{f}^3 + \overline{f}^3)}{(\underline{f}^1 + \overline{f}^1) + (\underline{f}^2 + \overline{f}^2) + (\underline{f}^3 + \overline{f}^3)}, \tag{3.3}$$

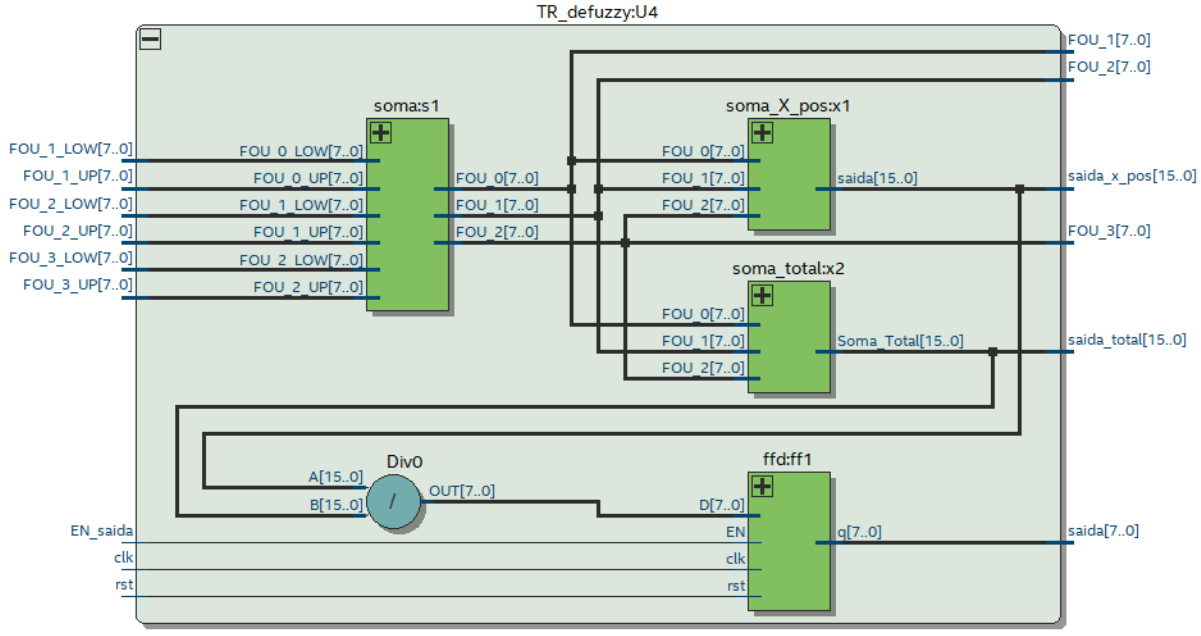

Figura 20 – Tipo redutor e defuzzificador.

4 RESULTADOS

4.1 Introdução

Este capítulo apresenta os resultados obtidos para a implementação do sistema de inferência *fuzzy*tipo-2 intervalar proposto. Para permitir a validação dos resultados do circuito foi utilizada uma ferramenta integrada ao Matlab® que permite modelar o sistema de inferência *fuzzy* implementado [39]. Com esta ferramenta obtêm-se arquivos e funções que empregam a teoria de conjuntos *fuzzy* para verificar os dados de um sistema. A área de interação do usuário permite que se realize o ajuste ao uso do SIF tipo-2 intervalar.

Para a implementação da descrição do sistema em Verilog foi utilizado o software Intel Quartus® Prime, que permite obter os valores de saída dos blocos de fuzzificação, base de regras, inferência, tipo-redutor e defuzzificação utilizando o simulador *ModelSim* a partir das descrições das simulações do circuito desenvolvido (testbenches). Todos os módulos empregam a linguagem de descrição de hardware Verilog e definem relações entre entradas e saídas em termos de fluxo de operações dentro do hardware através do nível de transferência de registradores RTL.

Para permitir a comparação com o modelo teórico, os resultados obtidos para diferentes combinações de valores de entrada (variando de 1 a 254) na saída do circuito (saída_defuzzy) foram compilados e transferidos para o próprio software Matlab®. As simulações validam o funcionamento do circuito implementado no Quartus® em relação ao modelo teórico elaborado no Matlab®.

4.1.1 *Fuzzy* Logic Toolbox para sistemas *fuzzy* tipo-2 intervalar

A Toolbox *fuzzy* tipo-2 para sistema de inferência *fuzzy* tipo-2 intervalar do Matlab® verifica os dados de entrada e saída utilizados do modelo computacional. A Figura 21 apresenta a janela principal do *Fuzzy* Logic Toolbox para SIF tipo-2. Na entrada do painel de visualização é gerada a configuração do sistema SIF por meio dos parâmetros a, b, c e d. Estes parâmetros formam as funções de pertinência trapezoidais de 2.3 das entradas x_1 e x_2, elaboradas através dos comandos Edit, Add Variable, respectivamente. É inserido no campo Range um delimitador das funções de pertinência de entrada x_1 e x_2 compreendidas entre 0 e 255.

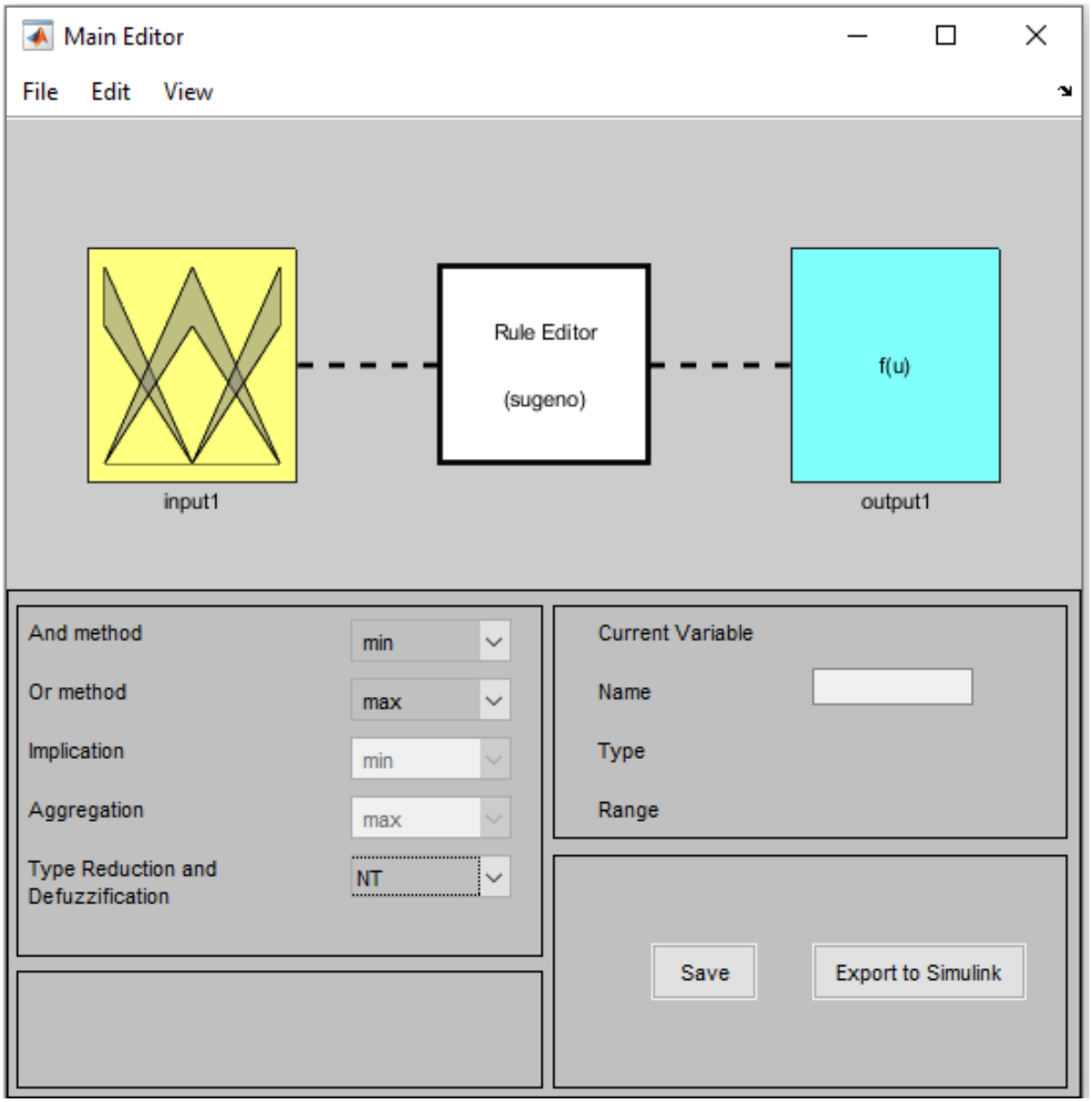

Figura 21 – Toolbox Matlab® *fuzzy* tipo-2.

As funções de pertinência do tipo-2 antecedentes superior e inferior que formam N1, Z1, P1 da entrada x_1 e N2, Z2, e P2 da x_2, foram adotadas de trabalhos anteriores [40]. São representadas no Toolbox através do Edit e, em seguida, utilizado o comando Membership Function Editor para sua criação, de acordo com a Figuras 22, 23, 24, 25, 26 e 27. De maneira semelhante à inserção das variáveis de entrada, ocorre a introdução de três variáveis de saída N, Z e P. Deste modo, é elaborada uma base de regras para cada dado de saída associado aos conjuntos de entrada N1, Z1, P1, e N2, Z2 e P2.

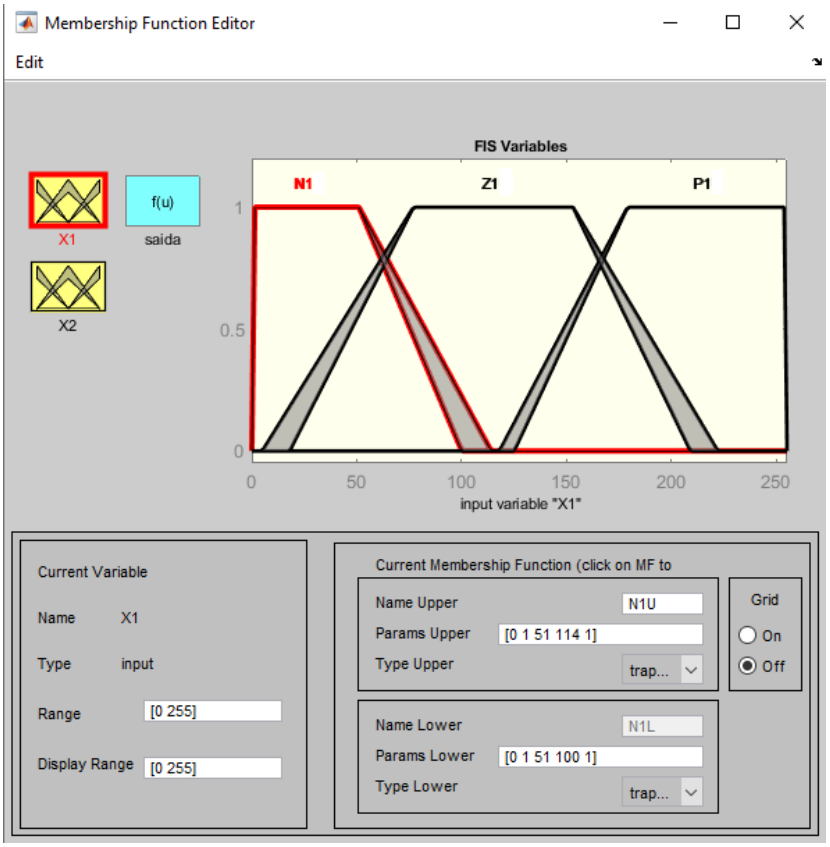

Figura 22 – Visão da função de pertinência tipo-2 N1 da entrada X1.

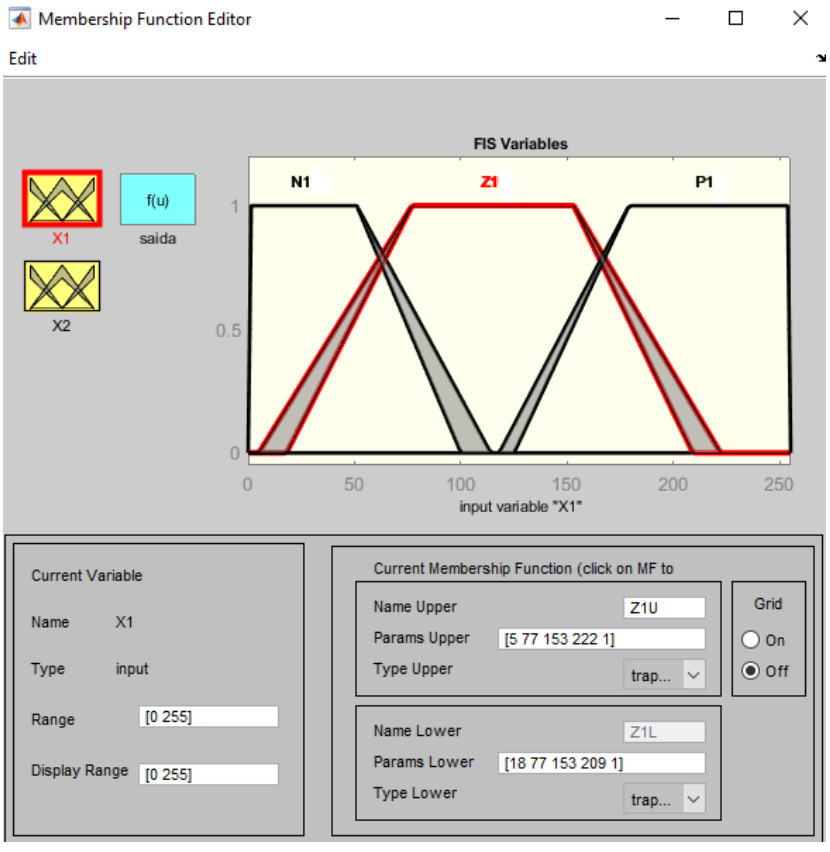

Figura 23 – Visão da função de pertinência tipo-2 Z1 da entrada X1.

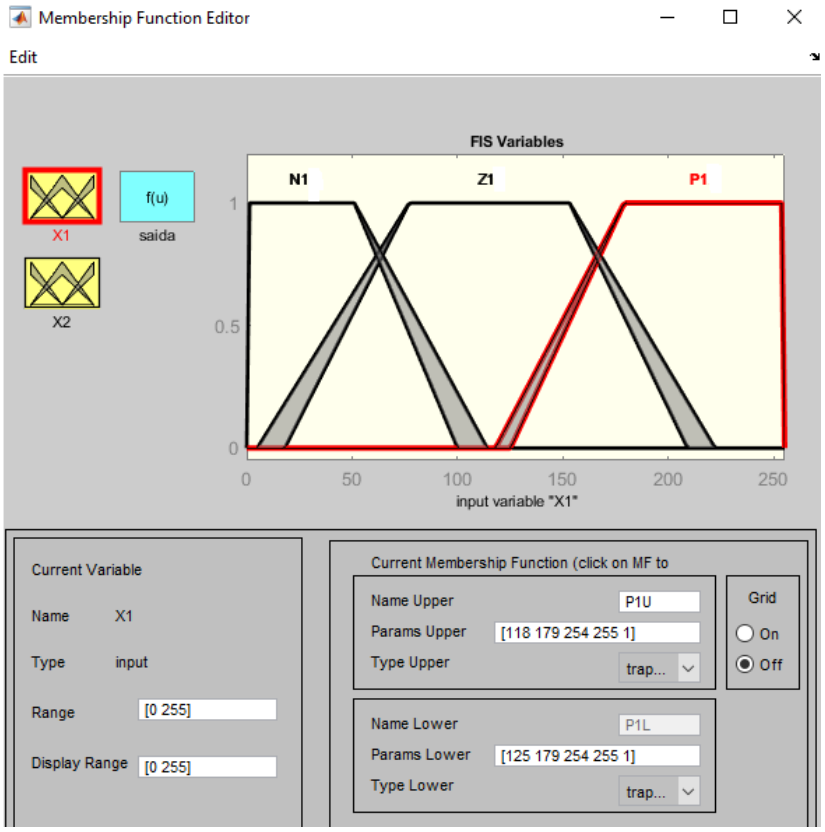

Figura 24 – Visão da função de pertinência tipo-2 P1 da entrada X1.

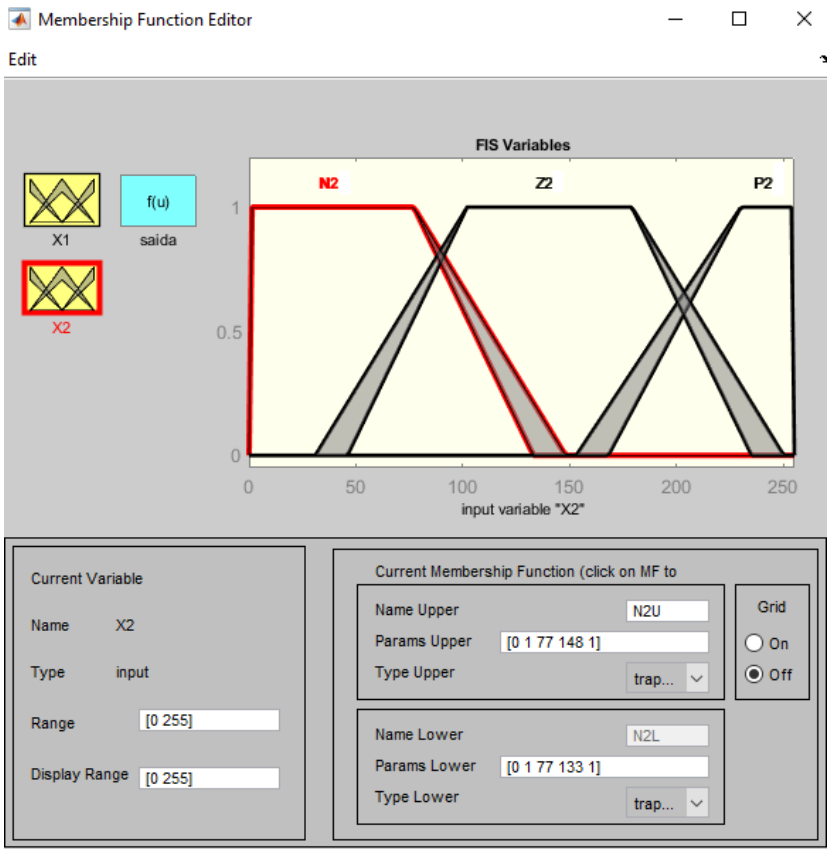

Figura 25 – Visão da função de pertinência tipo-2 N2 da entrada X2.

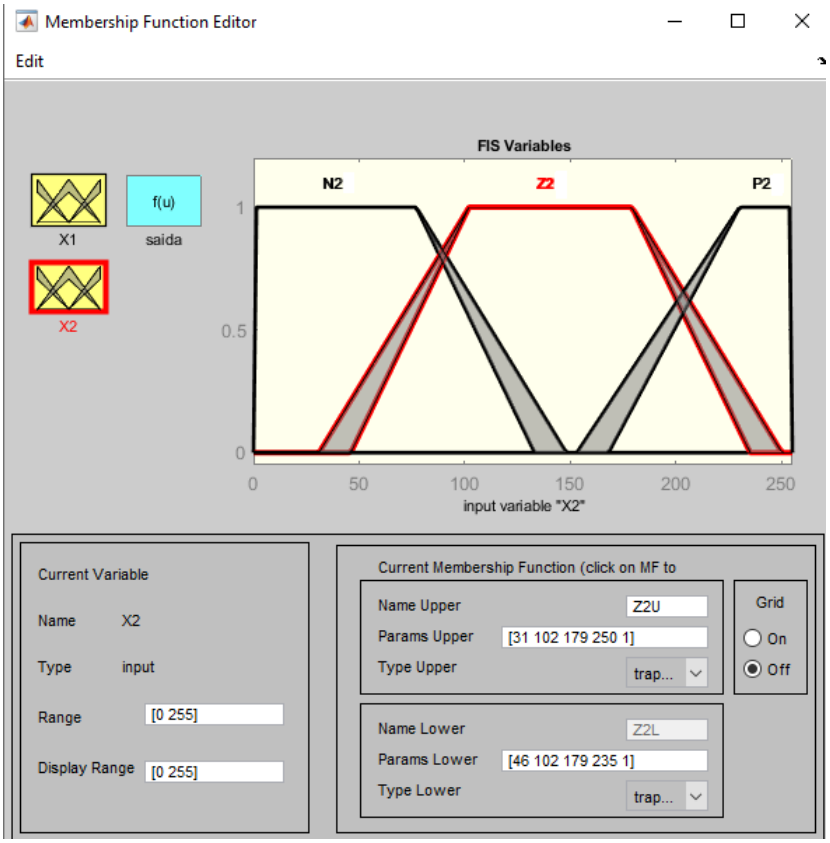

Figura 26 – Visão da função de pertinência tipo-2 Z2 da entrada X2.

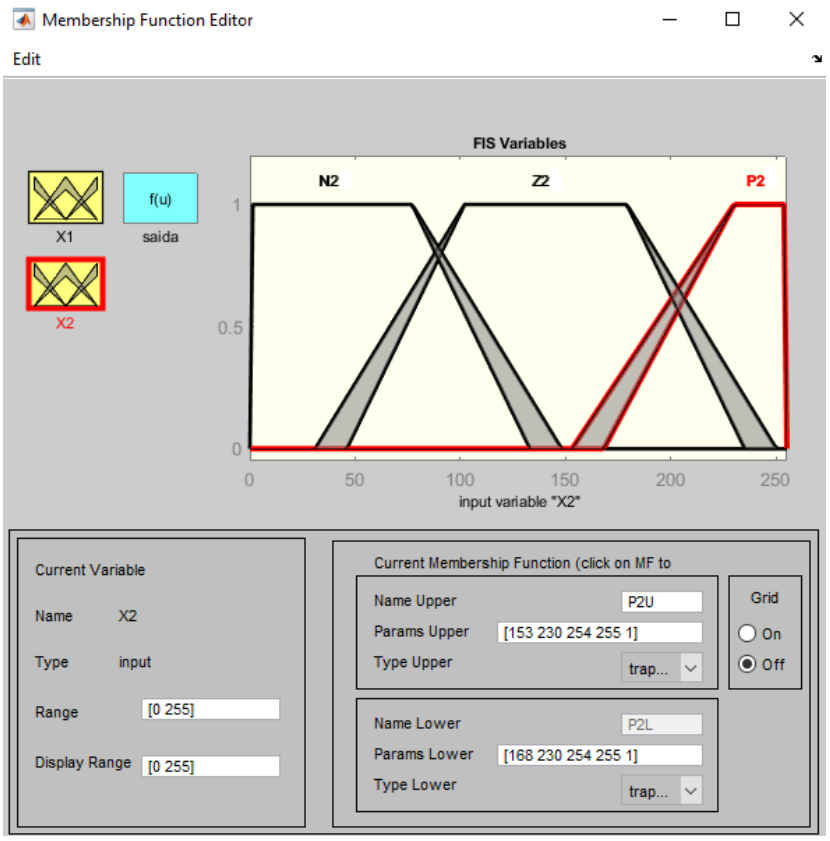

Figura 27 – Visão da função de pertinência tipo-2 P2 da entrada X2.

A base de regras é desenvolvida ao acessar a janela Rule Editor, como mostra a Figura 28, através dos comandos Edit, e Rule e é descrita pela relação das funções antecedentes da entrada x_1 e x_2. Por exemplo, considerando a entrada x_1 com a variável N1 e a entrada x_2 com a variável N2 o resultado obtido será o valor N, explicado no capítulo 3. Desta maneira, foram realizadas todas as combinações das variáveis de entrada, associando-se aos conjuntos de saída das variáveis, de acordo com a base de regras desenvolvida em [40].

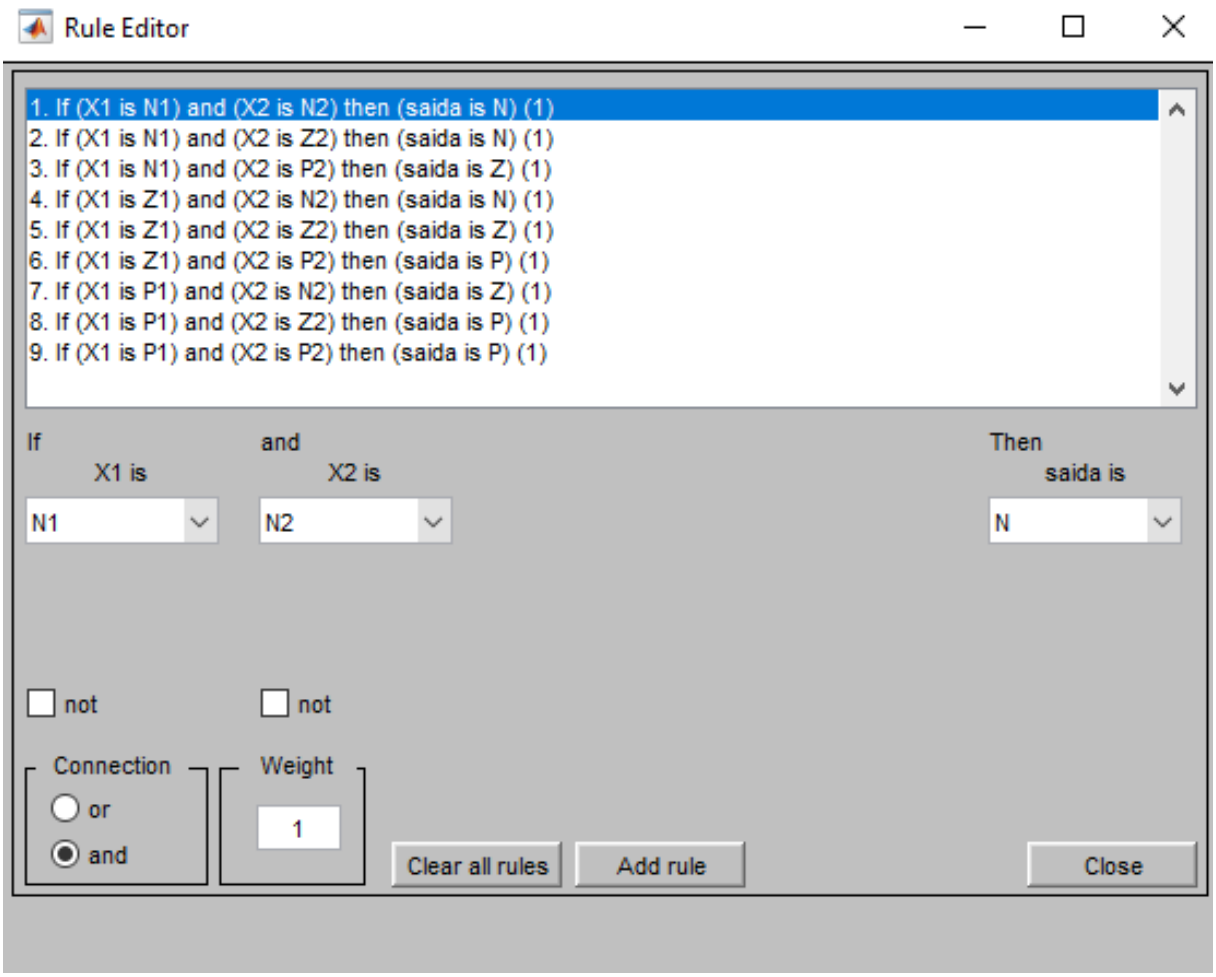

Figura 28 – Criação da base de regras

Uma vez definidos todos os parâmetros que caracterizam o sistema de inferência *fuzzy* tipo-2, é possível obter a superfície de controle em função das variáveis de entrada x_1, e x_2, conforme o gráfico mostrado na Figura 29.

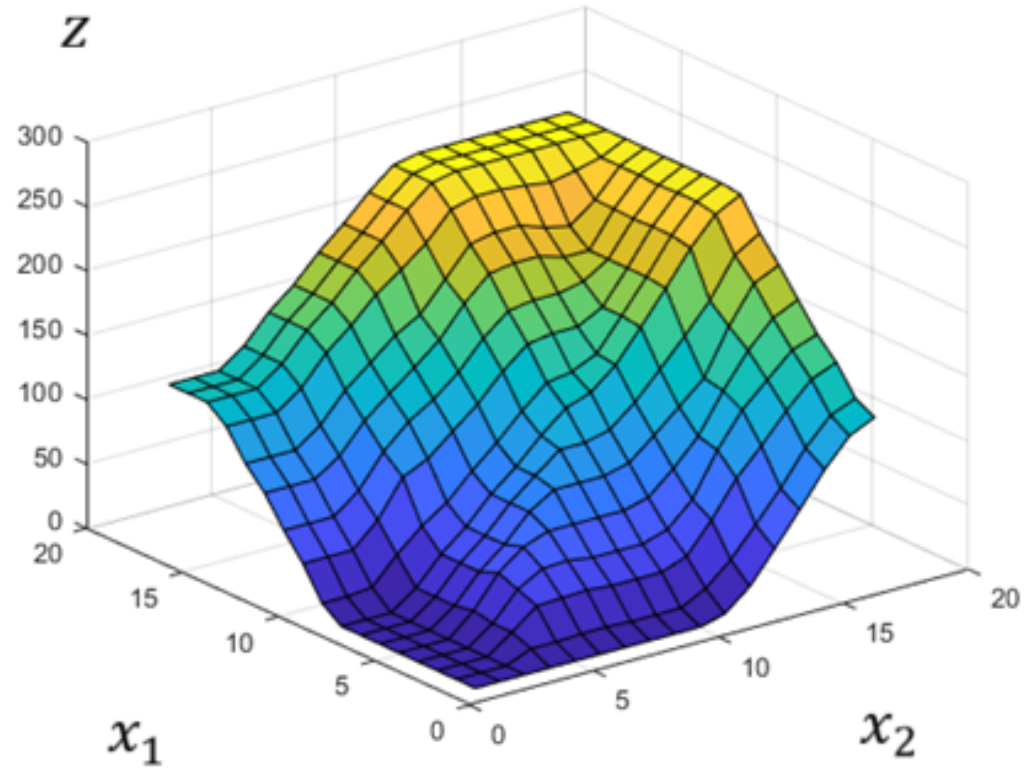

Figura 29 – Superfície de saída do sistema *fuzzy* tipo-2 em função das duas entradas do sistema.

4.1.2 Software Intel Quartus® Prime

O *ModelSim* é uma ferramenta para simular circuitos lógicos instalada em conjunto com o software Intel Quartus® Prime. Essa ferramenta proporciona simulações através de um analisador de forma de ondas, no qual é possível verificar os sinais em determinados instante de tempo. O código em Verilog desenvolvido no Quartus® denominado Fuzzy_1_tb é apresentado no apêndice A e descreve o Testebench responsável por gerar os estímulos e caracterizar as saídas do circuito por meio de formas de ondas. Por meio deste procedimento é possível analisar o comportamento do circuito e verificar o resultado do sistema *fuzzy* em relação ao valor teórico encontrado no Matlab®.

O fluxo de operação do sistema é ilustrado empregando-se um exemplo com dois valores de entrada, definidos como 176 para a Entrada_01 e 208 para Entrada_02. A Figura 30 exibe o resultado da unidade denominada FOU, que corresponde ao circuito do Fuzzificador. É possível observar que quando ocorre o pulso de CLK no instante 85ns, os dados 176 e o 208 seguem para as entradas de dois Flip Flops tipo D, que têm a função de armazenar os dados de entrada.

Os sinais Srst e EN_ENTRADA_FOU são comuns aos blocos denominados FOU, Unidade_controle_regras e TR_defuzzy. Após o sistema ser resetado pelo Srst em 65ns, o sinal enable denominado EN_ENTRADA_FOU é habilitado. Desta forma, começa o processamento do circuito. No momento que o dado for atualizado na saída dos Flip Flops, iniciam-se as funções de pertinência trapezoidais do tipo-2 intervalar.

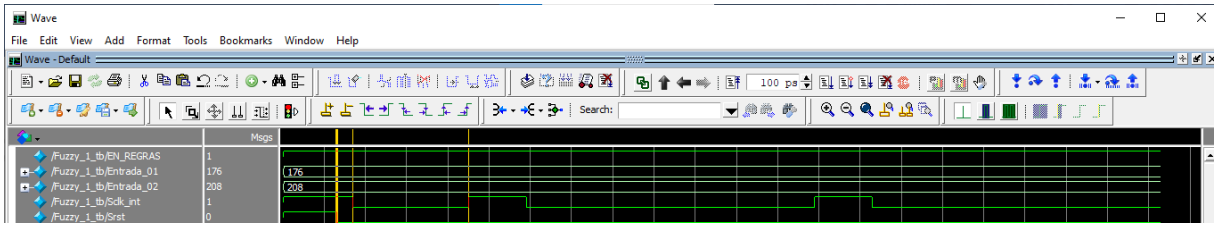

Figura 30 – Entrada dos valores de Entrada_01 e Entrada_02 no circuito.

Os dados da saída dos Flip Flops entram nas funções identificadas como Trapezio. Os dados de Input_01 entram no Trapezio UP_UM_01, Trapezio LOW_UM_01 até Trapezio UP_TRES_01 e Trapezio LOW_TRES_01. Os dados de Input_02 entram em Trapezio UP_UM_02, Trapezio LOW_UM_02 até Trapezio UP_TRES_02 e Trapezio LOW_TRES_02. Esta etapa gera os valores até o instante 225ns. É calculado o grau de pertinência de cada função e este valor é colocado nas saídas sFOU_01_UP, sFOU_01_LOW até sFOU_06_UP, e sFOU_06_LOW do bloco FOU por cada função de pertinência superior e inferior, como mostra a Figura 31.

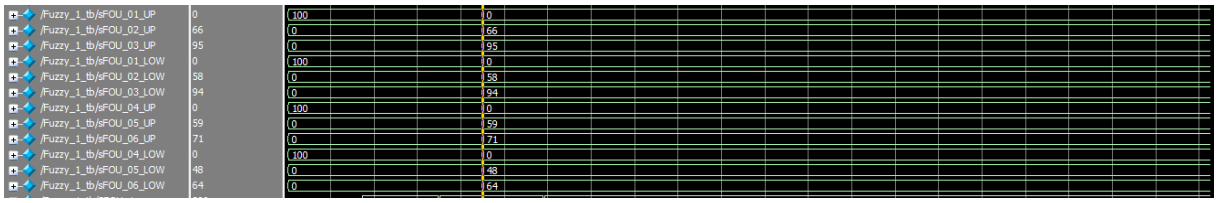

Figura 31 – Geração dos graus de pertinência correspondentes às funções de entrada.

Os blocos “geraSativo detecta_1” até “geraSativo detecta_6”, pertencentes à unidade FOU, são comparadores que identificam as funções ativas. São necessários seis bits para cada uma das seis funções de pertinência trapezoidais do tipo-2 para serem reconhecidas. Os comparadores geraSativo são conectados somente nos blocos das funções de pertinência superior, pelo fato destas possuírem um valor maior em sua base. Por este motivo, a verificação da função ativa é apresentada através do FOU_ativo. Se as funções de pertinência superior estiverem ativas, é suficiente para identificar que toda a função de pertinência *fuzzy* do tipo-2 intervalar está ativada.

A base de regras implementada é formada pela Unidade_controle_regras, que utiliza a máquina de estado que tem como entrada os barramentos FOU_ativo de 8 bits, EN_REGRAS, rst e clk. O fuzzificador informa à Unidade_controle_regras quais as funções estão ativas e esta unidade opera as regras a serem executadas no bloco de

inferência. As entradas EN_REGRAS e rst são chaves externas para habilitar e resetar o SIF tipo-2. As saídas Reset_Memoria e a Sequencia_regras da máquina de estado gerenciam o mecanismo de inferência e os registradores. A saída EN_Memorias tem a função de habilitar os registradores. O Reset_Memoria é dedicado a limpar os dados dos registradores e seu resultado é apresentado a partir do SReset_Memoria no instante 85ns. O barramento de 4 bits é dedicado a definir a sequência de regras ativas.

A base de regras proposta gerencia nove regras relacionadas com suas respectivas funções antecedentes e consequentes. Para identificar e projetar as regras ativas, a máquina de estado é projetada com onze estados. Um estado limpa as memórias, nove estados são dedicados a gerar o código das nove regras e um estado habilita as memórias para atualização dos valores. A primeira condição para iniciar o processo é analisar a entrada rst. Caso apresente nível lógico alto (nível 1), o estado atual é travado na condição de estado final END. Caso contrário, o estado atual passa a ser o estado inicial START. Os estados R0, até R8 dedicados às regras são responsáveis por gerar o código de cada regra ativa e analisar a possibilidade de existir uma próxima regra. Caso exista mais uma regra, o estado posterior será o da regra identificada. O estado final END finaliza a sequência de regras ativas, habilita as memórias para atualização e reinicia o processo com a verificação da entrada rst. Caso esta entrada esteja em nível alto, o estado final END é mantido. Caso contrário, a entrada rst está em nível baixo e retorna para o estado inicial START. Por exemplo, a entrada FOU_ativo igual a 100100 ativa a regra 0 de Sequencia_regras e as funções de pertinência do tipo-2 intervalar ativas são FOU_1 e FOU_4. Em 225ns, a Sequencia_regras ativa ainda está em 0 e a regra 4 é ativa com suas funções de pertinência FOU_2, FOU_3, FOU_5 e FOU_6, com um FOU_ativo equivalente a 011011. Isto resulta em saídas do bloco FOU definidas como sFOU_02_UP, sFOU_02_LOW, sFOU_03_UP, sFOU_03_LOW da Entrada_01, e sFOU_05_UP, sFOU_05_LOW, sFOU_06_UP e sFOU_06_LOW da Entrada_02. Estas constituem os graus secundários das funções de pertinência do tipo-2 ativas, como mostra a Figura 31.

O bloco inferencia relaciona as funções antecedentes com as funções consequentes. Nesta proposta, há três funções antecedentes N1, Z1, P1 para entrada x_1 e três N2, Z2, e P2 para x_2, conforme as Figuras 12, 13, 14 e 15. As três funções antecedentes da entrada x_1 relacionadas com as funções da entrada x_2 formam as nove regras apresentadas na Tabela 1, definida na base de regras Unidade_controle_regras.

Para implementar o mecanismo de inferência é usada a relação de mínimo e máximo de Mamdani. Por exemplo, para os graus sFOU_02_UP e sFOU_02_LOW, sFOU_03_UP e sFOU_03_LOW, sFOU_05_UP e sFOU_05_LOW, sFOU_06_UP e sFOU_06_LOW, com seus valores 66, 58, 95, 94, 59, 48, 71 e 64, para as funções ativas FOU_2, FOU_3, FOU_5 e FOU_6, com um FOU_ativo equivalente a 011011, a operação das funções de pertinência do tipo-1 superior ocorre de forma separada da inferior.

A Sequencia_regras executa as regras ativas, que relacionam os valores dos graus de pertinência das funções antecedentes da entrada x_1 com a x_2. Para este exemplo de valores de entrada estão ativas as regras 4, 5, 7 e 8. Os antecedentes da regra 4 são FOU_02 e FOU_05 Neste caso, isso corresponde aos graus de pertinência superior e inferior iguais a 66 e 58 para FOU_02_UP e FOU_02_LOW, respectivamente. Da mesma forma isso corresponde a 59 e 48 para FOU_05_UP e FOU_05_LOW. Como ambos são antecendentes da regra 4, aplicando a função de mínimo isso corresponde aos valores 59 e 48 para os níveis de disparo superior e inferior da regra 4. Os antecedentes da regra 5 são FOU_02 e FOU_06. Neste caso, isso corresponde aos graus de pertinência iguais a 66 e 58 para FOU_02_UP e FOU_02_LOW, respectivamente. Da mesma forma isso corresponde a 71 e 64 para FOU_06_UP e FOU_06_LOW. Aplicando a função de mínimo isso corresponde aos valores 66 e 58 para os níveis de disparo superior e inferior da regra 5.

Os antecedentes da regra 7 são FOU_03 e FOU_05. Neste caso, isso corresponde aos graus de pertinência iguais a 95 e 94 para FOU_03_UP e FOU_03_LOW, respectivamente. Da mesma forma isso corresponde a 59 e 48 para FOU_05_UP e FOU_05_LOW. Aplicando a função de mínimo isso corresponde aos valores 59 e 48 para os níveis de disparo superior e inferior da regra 7.

Por fim, os antecedentes da regra 8 são FOU_03 e FOU_06. Neste caso, isso corresponde aos graus de pertinência iguais a 95 e 94 para FOU_03_UP e FOU_03_LOW, respectivamente. Da mesma forma isso corresponde a 71 e 64 para FOU_06_UP e FOU_06_LOW. Aplicando a função de mínimo isso corresponde aos valores 71 e 64 para os níveis de disparo superior e inferior da regra 7.

Na sequência é obtido o máximo entres os valores associados aos três consequentes, que correspondem a zero para o consequente N (uma vez que estão associados às regras 0, 1 e 3), a 59 e 48 para o consequente Z (que está associado às regras 2, 4 e 6) e 71 e 64 para o consequente P (que está associado às regras 5, 7 e 8).

Os processos de redução de tipo e de defuzzificação são realizados em conjunto. É realizada a soma dos valores das funções consequentes superior e inferior do FOU. Cada um é multiplicado separadamente pela posição do centro de cada função de pertinência de saída no universo de discurso (que nesse caso são iguais a 0, 127 e 255) e dividido pela soma das funções consequentes da parte superior somada à inferior, conforme a equação 12. Todo este procedimento ocorre no instante 645ns, gerando o valor de saída do sistema de inferência igual a 198.

4.1.3 Validação de resultados

Os resultados teóricos do sistema de inferência *fuzzy* tipo-2, foram implementados no Matlab® utilizando o Interval Type-2 *Fuzzy* Logic Toolbox [39]. A fim de comparar e validar o resultado do circuito digital proposto, os resultados teóricos foram utilizados como referência de resposta em relação à saída obtida na simulação do *ModelSim*.

O *ModelSim* é configurado para monitorar as duas entradas e a saída *Crisp* da arquitetura implementada. Os sinais de entrada são sequências de oito bits, variando de 1 a 254 em intervalos regulares com espaçamento igual a 16. Dessa forma, variando as duas entradas para essas combinações de valores, foi possível obter a superfície correspondente à resposta do circuito, conforme mostra o gráfico na Figura 32, gerado utilizando o próprio software Matlab.

Para fins de comparação, foi calculada a superfície correspondente ao erro do resultado gerado pelo circuito em relação ao valor teórico modelado anteriormente, como mostra a 33. É possível notar que mesmo no pior caso o erro nunca é maior que 2 em termos absolutos. Considerando que a saída varia entre 0 e 255 isso corresponde a menos de 0,78% de erro, o que demonstra a capacidade do circuito de gerar efetivamente o comportamento correspondente ao sistema de inferência *fuzzy* tipo-2.

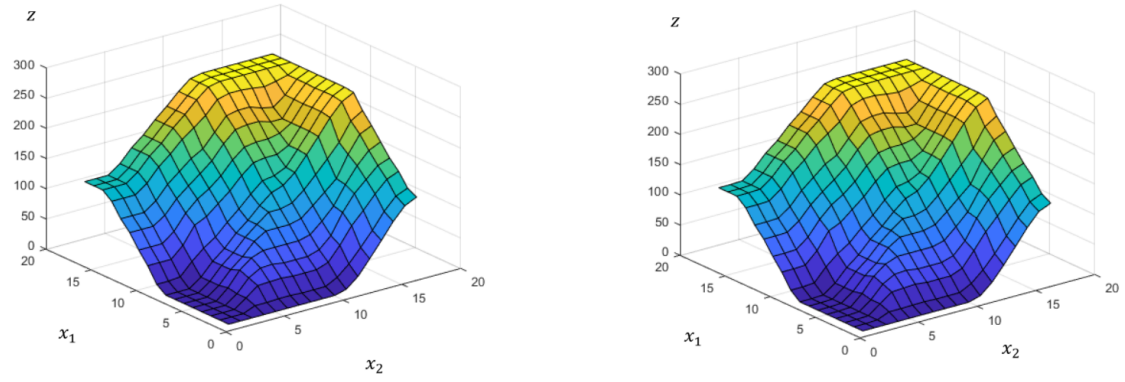

Figura 32 – Comparativo entre a superfície de controle gerada pelo circuito digital (esquerda) e o modelo teórico criado utilizando o Toolbox Matlab® (direita).

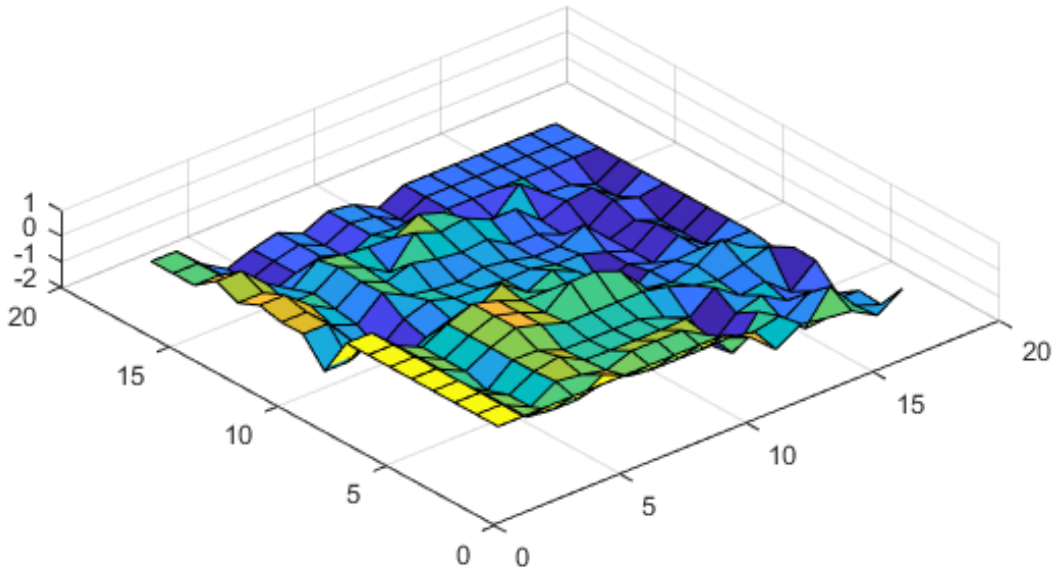

Figura 33 – Superfície de erro.

5 Conclusão

Este trabalho apresenta a implementação de uma arquitetura para um sistema de inferência *fuzzy* tipo-2 intervalar em Verilog. O hardware abordado consiste em cinco blocos: fuzzificador, inferência baseado no método de Mamdani, base de regras implementada em máquina de estados, tipo-redutor baseado no algoritmo de Nie-Tan e defuzzificador. O valor aplicado à entrada do circuito, também conhecido como crisp é convertido em seu valor *fuzzy* correspondente. Nesta etapa, são atribuídos valores linguísticos às variáveis de entrada para que essas possam ser trabalhadas pelo sistema de inferência *fuzzy*.

A arquitetura proposta tem como diferencial a utilização de funções de pertinência com formato trapezoidal. Além disso, há a possibilidade de programar a largura da mancha incerteza e a posição de cada função de pertinência no universo de discurso através da alteração dos paramêtros. A largura da FOU também pode ser ajustada para operar como um sistema de inferência *fuzzy* tipo-1. Desta maneira a arquitetura proposta pode operar como um SIF tipo-2 ou como um SIF tipo-1.

Por fim, a arquitetura com os módulos de fuzzificação, inferência, base de regras e redução de tipo apresentou valores dentro do esperado quando comparado com simulações feitas no Matlab® utilizando o Interval Type-2 Fuzzy Logic Toolbox. A comparação do resultado do circuito digital demonstrou um erro pequeno em relação ao modelo teórico, de 0,78% no pior caso, validando a sua funcionalidade.

A Código Verilog

O código Verilog desenvolvido como parte deste trabalho está disponível no link: <https://drive.google.com/file/d/1rh9suVOAo6B2ggrDb_KDPrgZvhatEm6b/>

Referências

1 ZADEH, L. A. Fuzzy sets. In: *Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers by Lotfi A Zadeh*. [S.l.]: World Scientific, 1996. p. 394–432.

2 ZADEH, L. A. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3, n. 1, p. 28–44, 1973.

3 AMADOR-ANGULO, L.; CASTILLO, O. A new fuzzy bee colony optimization with dynamic adaptation of parameters using interval type-2 fuzzy logic for tuning fuzzy controllers. *Soft Computing*, v. 22, n. 2, p. 571–594, Jan 2018. ISSN 1433-7479. Disponível em: <https://doi.org/10.1007/s00500-016-2354-0>.

4 HILLETOFTH, P.; SEQUEIRA, M.; ADLEMO, A. Three novel fuzzy logic concepts applied to reshoring decision-making. *Expert Systems with Applications*, v. 126, p. 133–143, 2019. ISSN 0957-4174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S095741741930123X>.

5 MAMDANI, E.; ASSILIAN, S. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, v. 7, n. 1, p. 1–13, 1975. ISSN 0020-7373. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0020737375800022>.

6 CHEN, G.; PHAM, T. T. *Introduction to fuzzy sets, fuzzy logic, and fuzzy control systems*. [S.l.]: CRC press, 2000.

7 CASTILLO, O. Introduction to type-2 fuzzy logic control. In: *Type-2 fuzzy logic in intelligent control applications*. [S.l.]: Springer, 2012. p. 3–5.

8 MITTAL, K. et al. A comprehensive review on type 2 fuzzy logic applications: Past, present and future. *Engineering Applications of Artificial Intelligence*, v. 95, p. 103916, 2020. ISSN 0952-1976. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0952197620302487>.

9 ZADEH, L. The concept of a linguistic variable and its application to approximate reasoning-iii. *Information Sciences*, v. 9, n. 1, p. 43–80, 1975. ISSN 0020-0255. Disponível em: <https://www.sciencedirect.com/science/article/pii/0020025575900171>.

10 ZHAO, J.; BOSE, B. Evaluation of membership functions for fuzzy logic controlled induction motor drive. In: *IEEE 2002 28th Annual Conference of the Industrial Electronics Society. IECON 02*. [S.l.: s.n.], 2002. v. 1, p. 229–234 vol.1.

11 LIANG, Q.; MENDEL, J. Interval type-2 fuzzy logic systems: theory and design. *IEEE Transactions on Fuzzy Systems*, v. 8, n. 5, p. 535–550, 2000.

12 CASTILLO, O. et al. Type-2 fuzzy logic: Theory and applications. In: *2007 IEEE International Conference on Granular Computing (GRC 2007)*. [S.l.: s.n.], 2007. p. 145–145.

13 KARNIK, N.; MENDEL, J.; LIANG, Q. Type-2 fuzzy logic systems. *IEEE Transactions on Fuzzy Systems*, v. 7, n. 6, p. 643–658, 1999.

14 MELGAREJO, M. A.; REYES, C. A. Peña. Hardware architecture and fpga implementation of a type-2 fuzzy system. In: . New York, NY, USA: Association for Computing Machinery, 2004. (GLSVLSI '04), p. 458–461. ISBN 1581138539. Disponível em: <https://doi.org/10.1145/988952.989063>.

15 MACIEL, R. Sistema fuzzy tipo-2 intervalar implementado em fpga baseado no algoritmo de nie-tan sistema fuzzy tipo-2 intervalar implementado em fpga baseado no algoritmo de nie-tan. In: . [S.l.: s.n.], 2020.

16 MELGAREJO, M.; PENA-REYES, C. A. Implementing interval type-2 fuzzy processors [developmental tools]. *IEEE Computational Intelligence Magazine*, v. 2, n. 1, p. 63–71, 2007.

17 SCHRIEBER, M. D.; BIGLARBEGIAN, M. Hardware implementation of a novel inference engine for interval type-2 fuzzy control on fpga. In: *2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. [S.l.: s.n.], 2014. p. 640–646.

18 SCHRIEBER, M. D.; BIGLARBEGIAN, M. Hardware implementation and performance comparison of interval type-2 fuzzy logic controllers for real-time applications. *Applied Soft Computing*, v. 32, p. 175–188, 2015. ISSN 1568-4946. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1568494615001787>.

19 ONTIVEROS-ROBLES, E. et al. A hardware architecture for real-time edge detection based on interval type-2 fuzzy logic. In: *2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. [S.l.: s.n.], 2016. p. 804–810.

20 ZANOTELLI, R. M. Consolidação do estudo e análise da robustez de operadores fuzzy considerando a abordagem intuicionista year= 2015, pages=83. In: . [S.l.: s.n.].

21 HEALD, G. The similarities between fuzzy logic and probability. *Research Gate, 3rd April*, 2017.

22 EBERHART, R. C.; SHI, Y. *Computational intelligence: concepts to implementations*. [S.l.]: Elsevier, 2011.

23 MAMDANI, E. H.; ASSILIAN, S. An experiment in linguistic synthesis with a fuzzy logic controller. *International journal of man-machine studies*, Elsevier, v. 7, n. 1, p. 1–13, 1975.

24 YASUNOBU, S.; MIYAMOTO, S. Automatic train operation by fuzzy predictive control. *Industrial Applications of Fuzzy Control. North Holland*, 1985.

25 MENDEL, J. M. Fuzzy sets for words: a new beginning. In: IEEE. *The 12th IEEE International Conference on Fuzzy Systems, 2003. FUZZ'03.* [S.l.], 2003. v. 1, p. 37–42.

26 KARNIK, N. N.; MENDEL, J. M. Introduction to type-2 fuzzy logic systems. In: IEEE. *1998 IEEE international conference on fuzzy systems proceedings. IEEE world congress on computational intelligence (Cat. No. 98CH36228)*. [S.l.], 1998. v. 2, p. 915–920.

27 KARNIK, N. N.; MENDEL, J. M.; LIANG, Q. Type-2 fuzzy logic systems, ieee transactions on fuzzy systems. *Volume*, v. 8, p. 808–821, 1999.

28 CASTILLO, O. et al. Type-2 fuzzy logic: theory and applications. In: IEEE. *2007 IEEE international conference on granular computing (GRC 2007)*. [S.l.], 2007. p. 145–145.

29 CASTILLO, O. et al. A comparative study of type-1 fuzzy logic systems, interval type-2 fuzzy logic systems and generalized type-2 fuzzy logic systems in control problems. *Information Sciences*, Elsevier, v. 354, p. 257–274, 2016.

30 SEPÚLVEDA, R. et al. Embedding a high speed interval type-2 fuzzy controller for a real plant into an fpga. *Applied Soft Computing*, Elsevier, v. 12, n. 3, p. 988–998, 2012.

31 SCHRIEBER, M. D.; BIGLARBEGIAN, M. Hardware implementation of a novel inference engine for interval type-2 fuzzy control on fpga. In: IEEE. *2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. [S.l.], 2014. p. 640–646.

32 MENDEL, J. M. Computing derivatives in interval type-2 fuzzy logic systems. *IEEE Transactions on Fuzzy Systems*, IEEE, v. 12, n. 1, p. 84–98, 2004.

33 LIMA, M. P. *A Lógica Fuzzy tipo 2 e um estudo de caso aplicado no Controle de Tráfego Aéreo*. Dissertação (Mestrado) — Universidade Federal da Bahia, 2008.

34 ROSS, T. J. *Fuzzy logic with engineering applications*. [S.l.]: John Wiley & Sons, 2005.

35 FRIAS-MARTINEZ, E. Real-time fuzzy processor on a dsp. In: IEEE. *ETFA 2001. 8th International Conference on Emerging Technologies and Factory Automation. Proceedings (Cat. No. 01TH8597)*. [S.l.], 2001. p. 403–408.

36 ANTÃO, R. *Type-2 fuzzy logic: uncertain systems' modeling and control*. [S.l.]: Springer, 2017.

37 HELLENDOORN, H.; THOMAS, C. Defuzzification in fuzzy controllers. *Journal of Intelligent & Fuzzy Systems*, IOS Press, v. 1, n. 2, p. 109–123, 1993.

38 ZANOTELLI, R. M. *Consolidação do estudo e análise da robustez de operadores fuzzy considerando a abordagem intuicionista*. Dissertação (Mestrado) — Universidade Federal de Pelotas, 2015.

39 TASKIN, A.; KUMBASAR, T. An open source matlab/simulink toolbox for interval type-2 fuzzy logic systems. In: IEEE. *2015 IEEE Symposium Series on Computational Intelligence*. [S.l.], 2015. p. 1561–1568.

40 SOUZA, G. A. et al. A novel fully-programmable analog fuzzifier architecture for interval type-2 fuzzy controllers using current steering mirrors. *Journal of Intelligent & Fuzzy Systems*, IOS Press, v. 34, p. 203–212, 2018. ISSN 1875-8967. 1. Disponível em: <https://doi.org/10.3233/JIFS-171118>.