UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS-GRADUAÇÃO
EM ENGENHARIA DE ENERGIA

Sistema *fuzzy* tipo-2 intervalar implementado em FPGA baseado no algoritmo de Nie-Tan

Regimar Maciel

Itajubá, dezembro de 2020

UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS-GRADUAÇÃO
EM ENGENHARIA DE ENERGIA

Regimar Maciel

Sistema *fuzzy* tipo-2 intervalar implementado em FPGA baseado no algoritmo de Nie-Tan

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

Área de Concentração: Microeletrônica

Orientador: Prof. Dr. Robson Luiz Moreno
Coorientadora: Profª. Dr. Paloma M. S. Rocha Rizol

Itajubá, dezembro de 2020

UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS-GRADUAÇÃO
EM ENGENHARIA DE ENERGIA

Regimar Maciel

Sistema *fuzzy* tipo-2 intervalar implementado em FPGA baseado no algoritmo de Nie-Tan

Dissertação apresentada à divisão de Pós-Graduação da Universidade Federal de Itajubá como parte dos requisitos para obtenção de título de Mestre em Engenharia de Elétrica.

Aprovado em ___/____/____

BANCA EXAMINADORA

_____

Prof. Dr. Robson Luiz Moreno (Orientador)

_____

Profª. Dr. Paloma M. S. Rocha Rizol (Coorientadora)

_____

Prof. Dr.Tales Cleber Pimenta

_____

Prof. Dr. Evaldo Renó Faria Cintra

Itajubá, dezembro de 2020

RESUMO

O presente trabalho tem como objetivo propor e validar uma arquitetura de um Sistema de Inferência *Fuzzy* (SIF) tipo-2 intervalar implementado em FPGA, que atenda aplicações de controle em tempo real, com processamento série no mecanismos de inferência. Neste trabalho são apresentados o projeto e a implementação dos circuitos que compõem o SIF tipo-2 intervalar, tais como: o circuito fuzificador tipo-2 intervalar (FOU); os circuitos de mínimo e máximo que são utilizados no mecanismo de inferência Mamdani (tipo-2); o circuito aplicado a máquina de estados que é utilizada pela base de regras e o circuito que processa as operações de tipo-redutor e defuzificação baseado no algoritmo de Nie-Tan. O hardware apresentado possui duas entradas de 8 bits com quatro funções de pertinência gaussianas para cada entrada, dezesseis regras e uma saída de 8 bits com sete funções de pertinência. E por fim, os resultados da implementação em FPGA são validados utilizando o mesmo SIF tipo-2 intervalar implementado no Matlab® com a *Toolbox* para type-2 fuzzy.

ABSTRACT

This work aims to propose and validate an architecture of an interval type-2 fuzzy logic systems implemented in FPGA, which meets real-time control applications, with serial processing in the inference mechanisms. This work presents the design and implementation of the circuits that make up the interval type-2 fuzzy logic systems, such as: the type-2 interval fuzzifier circuit (FOU); the minimum and maximum circuits that are used in the Mamdani inference mechanism (type-2); the circuit applied to the state machine that is used by the rules base and the circuit that processes the reducer-type and defuzzification operations based on the Nie-Tan algorithm. The hardware presented has two 8-bit inputs with four Gaussian pertinence functions for each input, sixteen rules, and an 8-bit output with seven pertinence functions. Finally, the results of the FPGA implementation are validated using the same interval type-2 fuzzy logic systems implemented in Matlab® with a Toolbox for type-2 fuzzy.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

FLIPS	*Fuzzy logic Inference Per Second*
BMM	Biglarbegian–Melek–Mendel
F_A	Frenagem alta
F_B	Frenagem baixa
F_BL	Frenagem bloqueio
F_E	Frenagem emergência
F_M	Frenagem média
F_MA	Frenagem muito alta
F_MB	Frenagem muito baixa
FOU	*Footprint of uncertainty*
FPGA	*Field programmable gate array*
KM	Karnik-Mendel
LMF	*lower membership function*
MF	*Membership function*
NT	Nie-Tan
PLL	*Phase Locked Loop*
SIF	Sistema de inferência fuzzy
SIF-T2	Sistema de inferência fuzzy tipo-2
TR	Tipo redutor
UMF	*upper membership function*
WM	Wu-Mendel

Sumário

1. CONSIDERAÇÕES INICIAIS

1.1. Contextualização

A teoria da lógica *fuzzy* emergiu em 1965 quando Zadeh introduz o conceito de conjuntos *fuzzy* [1], esse trabalho apresentou um novo conceito para a lógica computacional lidar com as incertezas contidas no significado das palavras. O artigo teve influência na representação da incerteza levantando questionamentos sobre os fundamentos que a teoria das probabilidades se baseava [2].

Em 1975, Mamdani apresenta a primeira aplicação bem sucedida de lógica *fuzzy* para controle de motores a vapor [3]. Desde então, as aplicações de controle *fuzzy* vem crescendo constantemente. Nas décadas de 1980 e 1990, houve grandes esforços de universidades e equipes de pesquisa da indústria para desenvolver soluções para controle utilizando *fuzzy* [4]. Atualmente, a lógica *fuzzy* é encontrada em quase todos os campos da engenharia e da ciência, incluindo sistema de controle, eletroeletrônicos, processamento de dados, sistemas especialistas, visão computacional e processamento de sinais [5] .

No mesmo ano em que Mamdani apresentou a primeira aplicação de um controlador *fuzzy*, Zadeh apresentou o conceito de lógica *fuzzy* tipo-2 [6]. Durante a década de 1960 e início 1970, o conceito de conjuntos *fuzzy* era questionado sobre sua capacida de modelar a incerteza. Esse questionamento tem como base o fato de que uma vez que as funções de pertinência são definidas, toda a incerteza relacionada com o significado das palavras desaparece, isso porque as funções de pertinência *fuzzy* são totalmente precisas [7]. A nova proposta de Zadeh teve como principal objetivo modelar as incertezas das informações [8]. O conceito de *fuzzy* tipo-2 propôs modelos matemáticos que melhoram a capacidade de lidar com a tomada de decisão baseada na subjetividade humana [6]. No entanto, a exigência de um elevado processamento computacional para modelar os conjuntos *fuzzy* tipo-2 dificultou as aplicações práticas. Somente a partir de 1998 a lógica *fuzzy* tipo-2 é melhor compreendida com o trabalho apresentado por Karnik e Mendel [9] onde toda teoria do Sistema de Inferência *Fuzzy* (SIF) tipo-2 é detalhada [7]. Em 2000 Liang e Mendel desenvolveram a teoria do SIF tipo-2 intervalar [10], a qual simplificou os cálculos das operações de entrada das funções de pertinência. Usando o conceito de duas funções de pertinência, superior e inferior, a incerteza pode ser modelada pelo SIF tipo-2 no intervalo estabelecido entre as funções [7]. Apesar das contribuições dos trabalhos [9], [11] e [10] o processamento para um SIF tipo-2 intervalar ainda apresentava um "gargalo" computacional na operação tipo-redutor, o que impossibilitava aplicações de controle

em tempo real. Em 2002, Wu e Mendel [12] apresentam o método iterativo para o cálculo do conjunto tipo-redutor de Karnik-Mendel. Essa proposta apresentou um aliviou na carga computacional, deixou de fornecer na saída apenas uma estimativas sobre a incerteza e passou a projetar um valor estimado dentro de um intervalo de conjuntos *fuzzy* tipo-2.

Motivados pelos conceitos apresentados nos trabalhos [9], [11], [10] e [12], Melgarejo et al. apresentaram em 2004 a primeira implementação em hardware para SIF tipo-2 intervalar com velocidade de processamento compatível para aplicações de controle em tempo real [13]. Essa arquitetura utilizou processamento em paralelo para implementar em FPGA os estágios de fuzificação, base de regras, mecanismo de inferência e tipo-redutor.

Após a publicação de Melgarejo et al. [13], diversas implementações em FPGA para SIF tipo-2 intervalar foram publicadas apresentando diferentes propostas de arquiteturas. Os métodos de tipo-redutor apresentados por Nie e Tan [14] em 2008 e M. Biglarbegian et al. [15] em 2010 contribuíram para que algumas trabalhos propusessem arquiteturas mais simples e com menor custo de *hardware*.

Em 2007, Melgarejo et al. [16] apresenta uma atualização da arquitetura já apresentada em 2004 [13], com maior número de regras e maior velocidade de processamento. Em 2012, Sepúlveda et al. [17] apresentam uma arquitetura que utiliza o método de Karnik-Mendel na operação tipo-redutor para o controle de velocidade de um motor DC em tempo real. Em 2014, Schrieber e Biglarbeian [18] apresentam a primeira aplicação de SIF tipo-2 intervalar que utiliza o algoritmo de tipo-redutor apresentado por M. Biglarbegian et al. [15]. Essa proposta obteve respostas mais rápida utilizando menos recursos de *hardware* em comparação as arquiteturas já apresentadas. Em 2015, Schrieber et al. [19] apresentam um estudo comparativo de arquitetura implementadas em FPGA utilizando os algoritmos tipo-redutor de M. Biglarbegian et al. [15], Nie e Tan [14], Wu e Mendel [12] e Karnik e Mendel [9]. Esse estudo detalhou as características de velocidade de processamento, tamanho de *hardware* e precisão de resposta que cada algorítmo apresenta. Em 2016, Robles et al. [20] apresentam um implementação SIF tipo-2 intervalar utilizando o algoritmo de Wu e Mendel [12] para processamento de imagens que se adapta a diferentes contrastes de iluminação.

Os principais trabalhos apresentados para implementação de SIF tipo-2 intervalar em FPGA utilizam processamento paralelo para obter hardware com alta velocidade de resposta. Em 2003, A. Gaona et al. [21] implementaram um sistema de inferência *fuzzy* com processamento serial no mecanismo de inferência. Entretanto, a velocidade de processamento alcançada ficou abaixo de 1Mega *Fuzzy logic Inference Per Second* (FLIPS), que é a taxa de processamento mínima para aplicações em tempo real [22]. Até o presente momento, não foi encontrado

implementação de SIF tipo-2 intervalar em FPGA com processamento sequencial em qualquer um de seus blocos dedicados.

Dentro deste panorama, o trabalho aqui apresentado propõe implementar um SIF tipo-2 intervalar em FPGA com capacidade de processamento superior 1Mega FLPIS, com processamento sequencial no mecanismo de inferência e utilização do algoritmo de Nie-Tan [14] para obter uma defuzificação direta.

1.2. Motivação

Quando Zadeh apresenta sua proposta para *fuzzy* tipo-2 em 1975, um profundo estudo ainda era necessário para que sua teoria pudesse ser aplicada. Diversos autores, além dos já mencionados, contribuíram no entendimento e aplicação do sistema de inferência *fuzzy* tipo-2 intervalar em diferentes ambientes. Atualmente, existe vários métodos e técnicas para implementar *fuzzy* tipo-2 intervalar que dão liberdade para projetar implementações em *hardware* com diferentes características [23].

A escolha de uma arquitetura em paralelo ou em série tem grande impacto na velocidade de processamento, tamanho e consumo de *hardware* [20]. Entretanto, a dificuldade em se alcançar velocidade de processamento para aplicações de tempo real com processamento série, fez com que os principais trabalhos já publicados optassem por um processamento totalmente em paralelo. Desta forma, existe uma lacuna na pesquisa sobre *hardware* com processamento série aplicados a sistema *fuzzy* tipo-2 intervalar. Neste contexto a escolha de um processamento serial também pode ser um boa opção para um *hardware* compacto com velocidade que atenda as aplicações de tempo real.

1.3. Objetivos

O objetivo deste trabalho é propor e validar a arquitetura de Sistema de Inferência *Fuzzy* (SIF) tipo-2 intervalar para implementação em hardware digital. Neste trabalho são apresentados o projeto e a implementação dos circuitos que compõem o SIF tipo-2 Intervalar, os quais são:

- Circuito de fuzificador com processamento paralelo para conjuntos *fuzzy* tipo-2 intervalar.
- Circuito de máximo e mínimo com processamento série para implementar a inferência Mamdani.

- Base de regras com máquina de estados.
- Circuito para aplicação do método de Nie-Tan com processamento paralelo.

Todos esses circuitos são implementados em uma FPGA EP4CE115F29C7 e validados utilizando a *Interval Type-2 Fuzzy Logic Toolbox* [24] do Matlab®.

1.4. Contribuições

As principais contribuições deste trabalho podem ser divididas em quatro tópicos, os quais são:

1. O código em Verilog que propõem uma solução para implementar o circuito aplicado nas funções de pertinência gaussianas;
2. O processamento em série para o mecanismo de inferência Mamdani;
3. A máquina de estados aplicada a base de regras para gerenciar as regras ativas;
4. A implementação do método de Nie-Tan para uma defuzificação direta.

O código em Verilog implementa as operações aritméticas em complemento de 2 e ajusta os valores polinomiais para manter a melhorar precisão utilizando o menor número de bits possível para implementar o circuito proposto Melgarejo et al. [25].

O mecanismo de inferência opera de maneira sequencial com um único circuito de identificação de mínimos e máximos independentemente da quantidade de regras existentes. Já as arquiteturas com processamento paralelo necessitam de um circuito dedicado a cada regra existente, ou seja, quanto maior o número de regras maior é o número de circuitos dedicados.

A máquina de estado opera a base de regras para organizar e processar somente as regras ativas, isso melhora a velocidade de processamento prejudicada pelo operação sequencial aplicada no mecanismos de inferências.

E finalmente, a implementação do algoritmo de Nie-tan em um SIF tipo-2 intervalar, com processamento sequencial no mecanismo de inferência, contribui para uma proposta de um *hardware* compacto e com velocidade de processamento compatível para aplicação de controle em tempo real.

1.5. Organização da dissertação

O capítulo 2 dessa dissertação apresenta o fundamento teórico dos conjuntos *fuzzy* tipo-2 e os blocos constituintes de um sistema de um SIF tipo-2 intervalar.

No capítulo 3, será apresentada a proposta para implementação em *hardware* de uma arquitetura de SIF tipo-2 intervalar. Serão mostradas e discutidos as especificações, o projeto e

o *layout* de todos os circuitos e códigos *Verilog* desenvolvidos para implementação do SIF tipo-2 intervalar.

O capitulo 4 apresenta os resultados da arquitetura implementada em FPGA os quais são comparados com os resultados do modelo implementado no Matlab® utilizando o Interval Type-2 *Fuzzy* Logic Toolbox [24].

O capitulo 5 traz as considerações finais sobre este trabalho e as sugestões de trabalhos futuros.

2. Lógica *Fuzzy*

2.1. Introdução

Este capítulo apresenta os fundamentos básicos da lógica *fuzzy* tipo-1 e tipo-2 que serão necessários para implementação em hardware digital FPGA. A primeira parte deste capítulo apresenta a conceituação de conjuntos *fuzzy* tipo-1 e tipo-2, suas principais definições e terminologias. Em seguida são apresentadas as operações e os principais módulos constituintes do controlador *fuzzy* tipo-2 intervalar *singleton*.

2.2. Conjunto *fuzzy* tipo-1

Na teoria dos conjuntos clássicos *crisp* ou em conjuntos *fuzzy*, o universo de discurso define todas as informações disponíveis sobre um determinado problema. Uma vez definido esse universo, é possível identificar certos eventos nesse espaço de informações. A Figura 1 (a) mostra uma abstração de um universo de discurso, X, em um conjunto A (clássico), ou seja, não há incerteza na prescrição ou localização dos limites para esse conjunto. A Figura 1 (b) apresenta uma abstração de um conjunto *fuzzy*, que por outro lado, é prescrito por propriedades vagas ou ambíguas, como mostra o limite do conjunto Ã na Figura 1 (b). O ponto "a" na Figura 1 (a) é claramente um membro do conjunto *crisp* de A; O ponto b não é um membro do conjunto A. A Figura 1 (b) mostra a fronteira vaga e ambígua de um conjunto *fuzzy* Ã em um mesmo universo X. Na região central do conjunto *fuzzy*, o ponto "a" é claramente um membro completo do conjunto. Fora da região limite do conjunto *fuzzy*, o ponto "b" claramente não é um membro do conjunto *fuzzy*. No entanto, a participação do ponto "c" é ambígua. Se a associação completa de um conjunto for representada pelo número 1, e a não associação de um conjunto for representada por 0, o ponto "c" na Figura 1 (b) deve tem valor intermediário de associação no conjunto *fuzzy* Ã dentro um intervalo entre [0,1]. Portanto, quando o ponto "c", no conjunto Ã, se aproxima da região central, seu valor tende a 1. A medida que se move para deixar a região de limite central, seu valor tende a 0, [2].

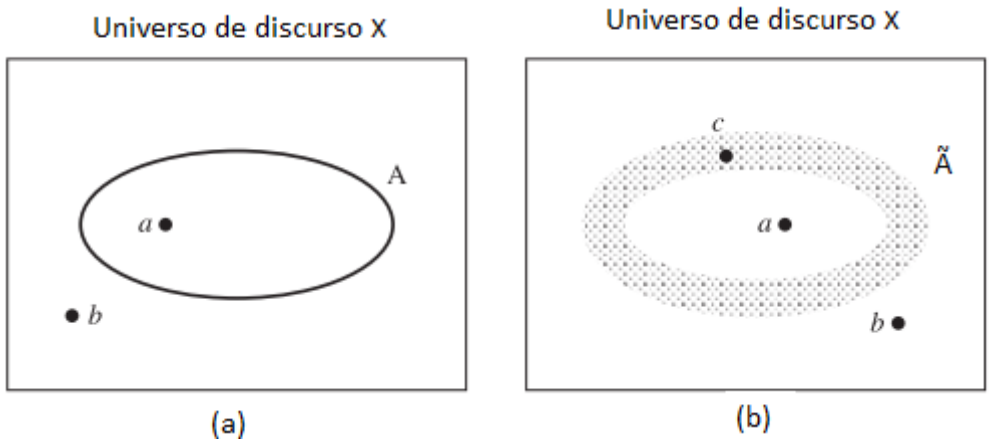


Figura 1: (a) Limite de conjuntos crisp (b) Limite de conjuntos fuzzy. Fonte:[2]

Na logica clássica, a transição de um elemento no universo de discurso entre associação e não associação em um determinado conjunto é abrupta. Para um elemento em um universo que contém conjuntos *fuzzy*, essa transição pode ser gradual e em conformidade com o fato de que os limites dos conjuntos *fuzzy* são vagos e ambíguos. Portanto, a participação de um elemento do universo de discurso neste conjunto é medida por uma função que descreve a imprecisão e a ambiguidade [2].

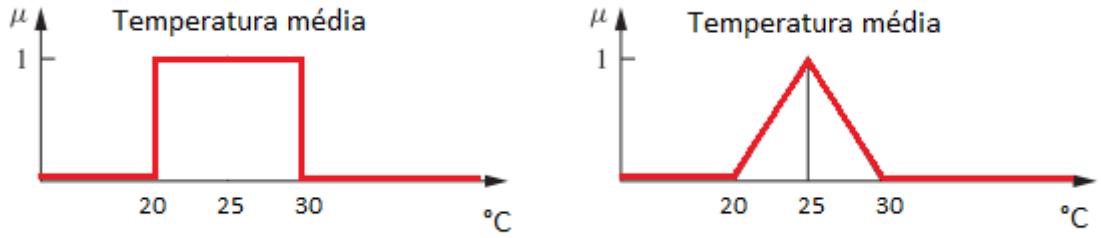


Figura 2:Conjunto representando temperatura média (a) Clássico (b) Fuzzy. Fonte:[26]

Seja um conjunto para determinar a temperatura média em °C em um determinado processo, Figura 2. Pode-se representar este conjunto tanto por conjuntos clássicos quanto por conjuntos *fuzzy*. A função de pertinência $\mu_A(x)$ do conjunto clássico, é representada na equação (2.1).

$$\mu_A(x) = \begin{cases} 1 & x \in A \\ 0 & x \notin A \end{cases} \qquad (2.1)$$

Já a função de pertinência $\mu_N(x)$ que representa o conjunto *fuzzy* [26], é representada na equação (2.2).

$$\mu_A(x) = \begin{cases} \dfrac{x-20}{25-20}, & 20 < x < 25 \\ \dfrac{30-x}{30-25}, & 25 < x < 30 \\ 0, & 20 > x < 30 \end{cases} \qquad (2.2)$$

Se X é uma coleção de objetos denotados genericamente por x, em um conjunto *fuzzy* A, X é definido como um conjunto de pares ordenados onde $\mu_A(x)$ é a função de pertinência que mapeia cada elemento de X para um grau de associação entre 0 e 1 de um conjunto *fuzzy* [27]. Matematicamente, o conjunto *fuzzy* A, definido em um universo de discurso X pode ser representado pela Equação (2.3).

$$A = \{(x, \mu_A(x)) | x \in X\} \tag{2.3}$$

Para um universo de discurso discreto o conjunto A é representado pela Equação (2.4).

$$A = \sum_{i=1}^{n}, \mu_A(x_i)/x_i \tag{2.4}$$

As principais operações de conjuntos *fuzzy* tipo-1 são apresentadas a seguir com base nos trabalhos [2] e [27].

Support: o suporte de um conjunto *fuzzy* A é o conjunto *fuzzy* de todos os pontos $x \in X$, de modo que $\mu_A(x) > 0$.

$$suporte\ (A) = \{\ x | \mu_A(x) > 0\ \} \tag{2.5}$$

Core**:** o *core* ou centro de um conjunto *fuzzy* é o conjunto de todos os pontos $x \in X$,d e modo que $\mu_A(x) = 1$.

$$core\ (A) = \{\ x | \mu_A(x) = 1\ \} \tag{2.6}$$

Boundaries: é o limite que compreende os elementos do universo de discurso que se encontram no intervalo de [0,1]. Esses elementos são definidos por funções matemáticas que representam a incerteza do conjunto. Na Figura 3 é exemplificado as definições de support, core e boundary.

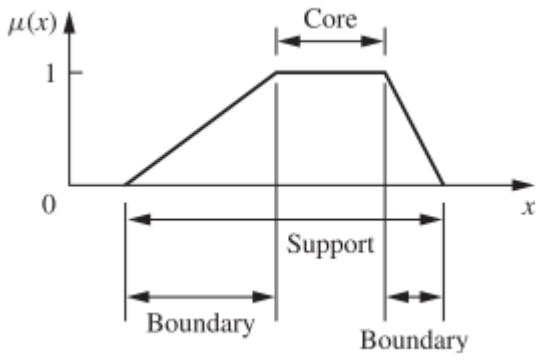


Figura 3: Core, support, e boundaries de um conjunto fuzzy. Fonte:[2]

Normal: Um conjunto *fuzzy* A é "normal" se seu núcleo não for vazio. Em outras palavras, sempre é possível encontrar um ponto x ∈ X tal $\mu_A(x) = 1$. A Figura 4 (a) representa um conjunto normal.

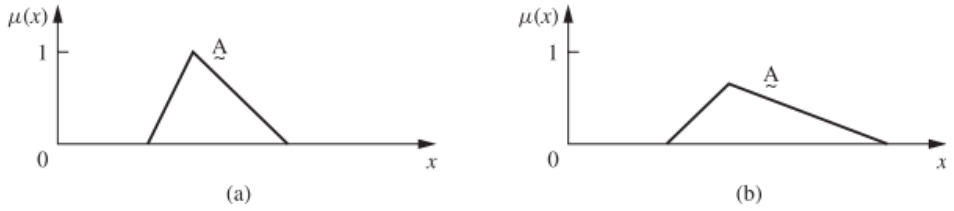


Figura 4:(a) conjunto fuzzy normal (b) subnormal. Fonte:[2].

***Crossover point*:** ou ponto de cruzamento de um conjunto *fuzzy* A em um ponto x ∈ X no qual $\mu_A(x) = 0{,}5$.

$$\text{crossover}\ (A) = \{\, x | \mu_A(x) = 0{,}5 \,\} \tag{2.7}$$

α-cut: é o conjunto *fuzzy* A definido por :

$$A_\alpha = \{\, x | \mu_A(x) \geq \alpha \,\} \tag{2.8}$$

***Containment*:** quando um conjunto *fuzzy* A está contido no conjunto *fuzzy* B (ou, equivalentemente, A é um "subconjunto" de B) se e somente se $\mu_A(x) \leq \mu_B(x)$ para todos os x.

$$A \subseteq B \Leftrightarrow \mu_A(x) \leq \mu_B(x) \tag{2.9}$$

Union: A "união" de dois conjuntos *fuzzy* A e B é um conjunto *fuzzy* C, escrito como C = A∪B ou C = A ou B, cujo MF está relacionado aos de A e B por:

$$\mu_C(x) = \max(\mu_A(x), \mu_B(x) = \mu_A(x) \vee \mu_B(x). \tag{2.10}$$

Intersection: A "interseção" de dois conjuntos *fuzzy* A e B é um conjunto *fuzzy* C, escrito como C = A∩B ou C = A e B, cujo MF está relacionado aos elementos de A e B por:

$$\mu_C(x) = \min(\mu_A(x), \mu_B(x) = \mu_A(x) \wedge \mu_B(x). \tag{2.11}$$

Complement or Negation: O "complemento" de um conjunto *fuzzy* A, denotado por A (¬ A, not A), é definido como:

$$\mu_A(x) = 1 - \mu_A(x). \tag{2.12}$$

Variáveis linguísticas: Para caracterizar um conjunto *fuzzy* as funções de pertinência *fuzzy* são representadas por variáveis linguísticas, as quais são definidas por três elementos principais: (x, T(x), I), onde x é o nome da variável, T(x) é um conjunto de valores linguísticos para os valores de x e I é o universo de discurso onde a variável está contida [7]. Os valores linguísticos T(x) são definidos por MF elaboradas por fórmulas matemáticas. Onde as funções

de pertinência podem ser definidas triangular, trapezoidal, gaussiana, sino, S e Z. As equações que estabelecem cada tipo de função de pertinência é abordada na seção (2.3).

2.3. Conjunto *fuzzy* tipo-2

O conceito de conjunto *fuzzy* do tipo-2 é introduzido por Zadeh em 1975 [6] como uma extensão do conceito do conjunto *fuzzy* tipo-1 [1]. Os conjuntos *fuzzy* tipo-2 podem ser usados em situações em que há incerteza na forma da função de pertinência ou em alguns de seus parâmetros. Quando não é determinada a pertinência de um elemento em um conjunto como 0 ou 1, utiliza-se conjuntos *fuzzy* tipo-1. Da mesma forma, quando a situação é tão imprecisa que se torna difícil determinar o grau de associação de uma função de pertinência com valores especificados entre 0 e 1, utiliza-se conjuntos *fuzzy* tipo-2 [27].

Na função de pertinência tipo-1, Figura 5 (a), para x = x'= 6 o valor de pertinência representado na função é de 0,6, esse valor é exato e não possui incerteza relacionada ao grau de pertinência da função.

Na função de pertinência tipo-2, para x = x'=6 são obtidos valores de pertinência no intervalo onde a linha vertical intercepta a mancha de incerteza *Footprint Of Uncertainty* (FOU), que permite que a incerteza seja trabalhada pelo SIF tipo-2, como pode ser observado na Figura 5 (b). No conjunto *fuzzy* tipo-2 cada valor da função de pertinência primária, dentro do intervalo 0,3 e 0,9, Figura 5 (b), tem um peso associado na terceira dimensão, chamada de função de pertinência secundária. A função de pertinência secundária do conjunto *fuzzy* tipo-2 pode ser uniforme ou não uniforme, como representado na Figura 5 (b). A função de pertinência secundária uniforme é chamada de função de pertinência *fuzzy* tipo-2 intervalar, e é obtida quando todos os valores são unitários [7].

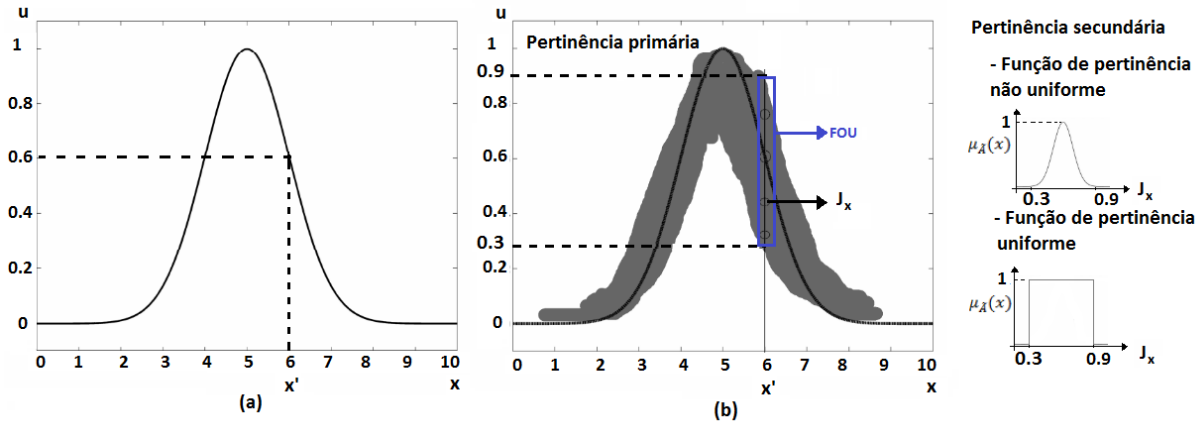


Figura 5: (a) Função de pertinência tipo-1 (b) Função de pertinência tipo-1 com mancha de incerteza.

As funções de pertinência fuzzy tipo-2 também são totalmente precisas, porém, a mancha de incerteza FOU, permite que a incerteza seja modelada pelo Sistema de Inferência *Fuzzy* (SIF) tipo-2. A Figura 6 (b) apresenta a visão tridimensional para as funções de pertinência primária e secundária no formato gaussiana.

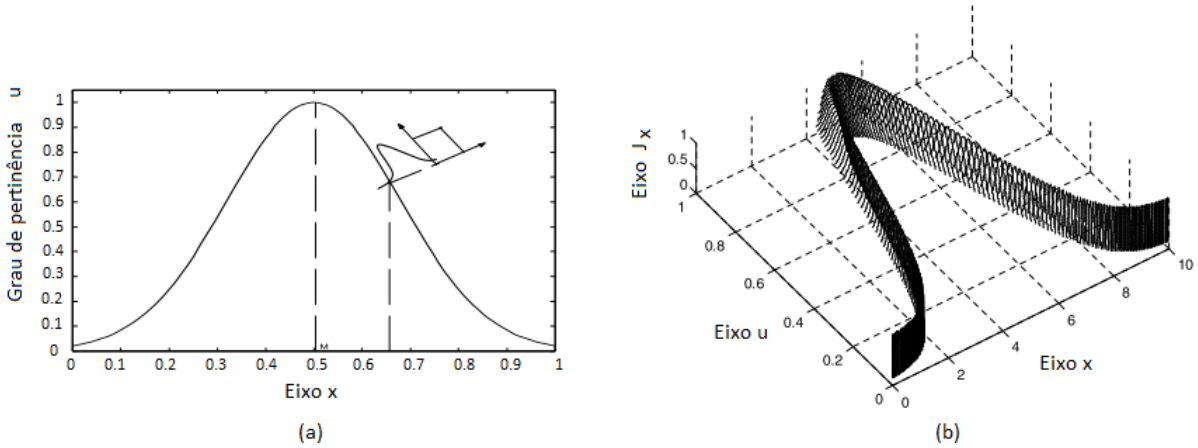


Figura 6: (a) Conjunto fuzzy tipo-2 em que o grau de pertinência de cada ponto de domínio é um conjunto Gaussiano tipo-1 (b) Visão tridimensional de uma função de pertinência Gaussiana tipo-2. Fonte [28]

A função de pertinência *fuzzy* tipo-2 intervalar é delimitada por uma função de pertinência inferior, *Lower Membership Function* (LMF), e uma função de pertinência superior, *Upper Membership Function* (UMF), Figura 7. As funções de pertinência inferior e superior são conjuntos *fuzzy* tipo-1. A área entre a função de pertinência inferior e superior é o intervalo que determina o *Footprint Of Uncertainty* (FOU). Também é importante notar que um conjunto *fuzzy* tipo-2 incorpora inúmeros conjuntos *fuzzy* tipo-1 representados pelo FOU.

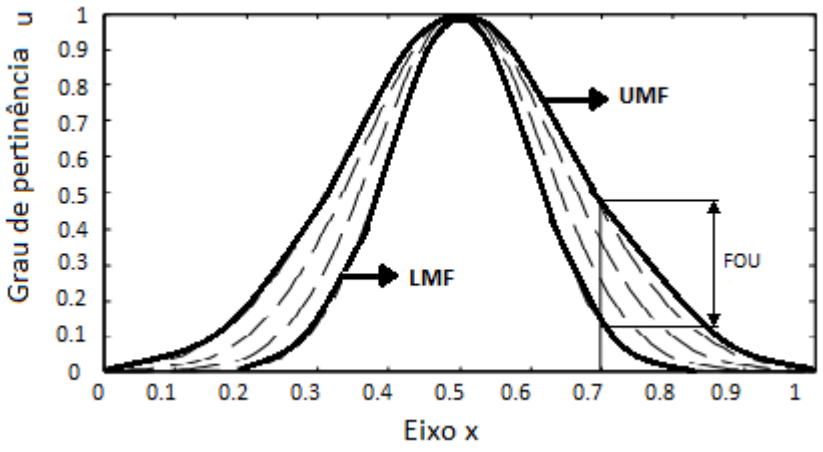


Figura 7: FOU formada pelas funções *UMF LMF*.

Utilizando o intervalo estabelecido entre as funções de pertinência superior (UMF) e inferior (LMF) para modelar a incerteza FOU, Figura 7, pode-se criar diversos tipos de função, como por exemplo: triangular, trapezoidal, gaussiana, sino, S e Z, utilizando funções *fuzzy* tipo-1. A Figura 8 mostra diferentes tipos de funções fuzzy tipo-2 [27].

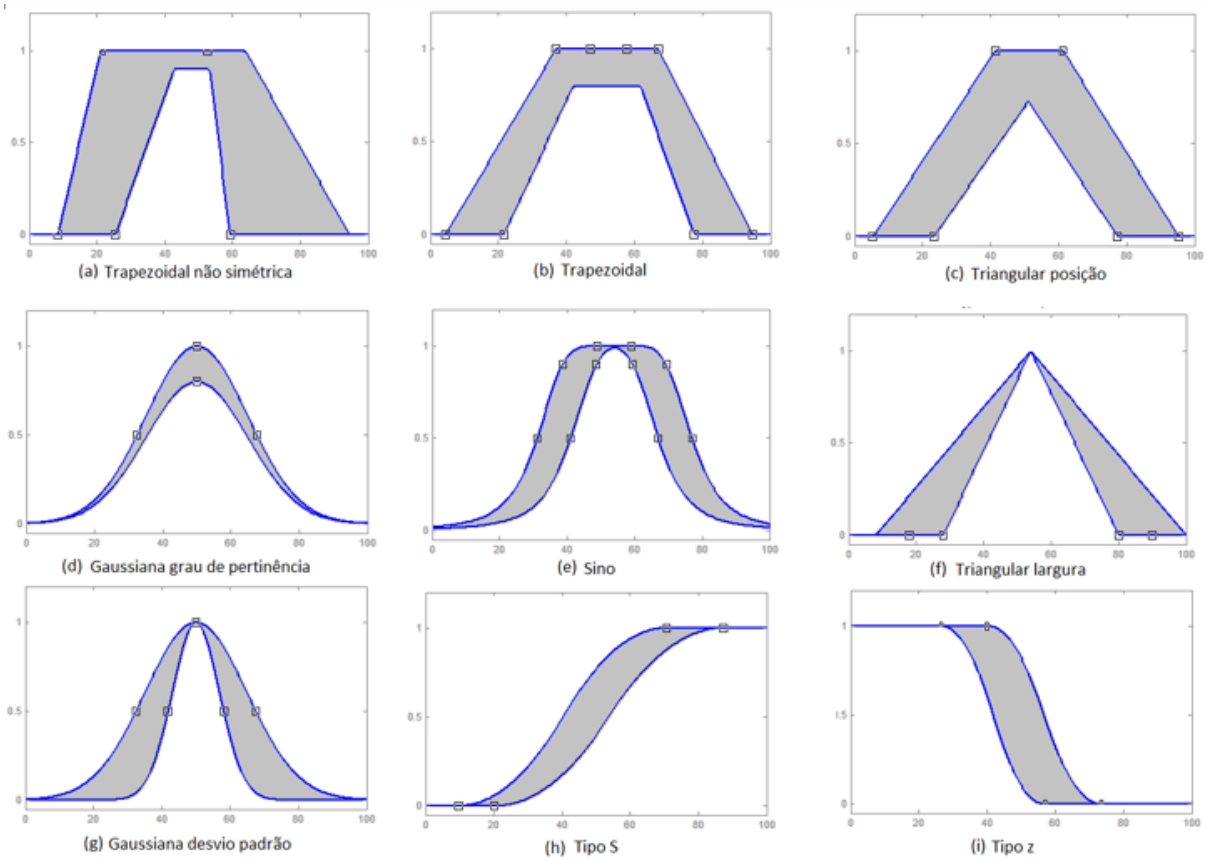


Figura 8: Tipos de funções de pertinência SIF T2

A mancha de incerteza em um conjunto *fuzzy* tipo-2 possui duas categorias principais, gaussiano ou linear por partes. Os conjuntos *fuzzy* tipo-2 em formato gaussiano podem ser modelados pela posição de subida e descida das funções, pelo desvio padrão ou utilizando diferentes valores para o máximo grau de ativação.

A equação (2.13) define o formato da função UMF da forma sino apresentada na Figura 8 (e), e a equação (2.14) define o formato da função LMF. Os parâmetros ajustáveis são [m1, m2, σ]. Onde "m1" define o centro da função de subida, "m2" o centro da função de descida e "σ" é o desvio padrão. Esse formato de configuração permite representar a função gaussiana com um platô máximo que se estende entre o valor de "m1" e "m2" criando uma forma de sino, Figura 8 (e).

$$\mu_{\bar{x}}(x) = \begin{cases} e^{-\frac{(x-m_1)^2}{2\sigma^2}}, & x < m_1 \\ 1, & m_1 < x < m_2 \\ e^{-\frac{(x-m_2)^2}{2\sigma^2}}, & x > m_2 \end{cases} \qquad (2.13)$$

$$\mu_{\underline{x}}(x) = mín\left\{ e^{-\frac{(x-m_1)^2}{2\sigma^2}}, e^{-\frac{(x-m_2)^2}{2\sigma^2}} \right. \qquad (2.14)$$

As funções gaussianas tipo S, Figura 8 (h), e tipo Z, Figura 8 (i), são determinadas utilizando os mesmo princípios da função sino, Figura 8 (e). Para a função gaussianas tipo S, Figura 8 (h), os valores de entradas maiores que "m1" são definidos com o valor máximo do grau de pertinência da função. Da mesma forma os valores menores que "m2", para a função gaussianas tipo Z, Figura 8 (i), são definidos com o valor máximo do grau de pertinência da função.

Para modelar a forma gaussiana por desvio padrão apresentada na Figura 8 (g), os paramentos [m, σ1, σ2] devem ser ajustados. A equação (2.15) apresenta a função aplicada a UMF e a equação (2.16) apresenta a função aplicada a LMF do FOU. A configuração é feita com a definição do centro das funções "m" e com um desvio padrão "σ1, σ2", onde σ2 deve ter um valor menor ao estabelecido e em σ1.

$$\mu_{\overline{x}}(x) = e^{-\frac{(x-m)^2}{2\sigma_1{}^2}} \tag{2.15}$$

$$\mu_{\underline{x}}(x) = e^{-\frac{(x-m)^2}{2\sigma_2{}^2}} \tag{2.16}$$

Já a função gaussiana por grau de pertinência apresentada na Figura 8 (d), utiliza o mesmo modelo matemático já apresentado nas equações (2.15) e (2.16), porém com um desvio padrão "σ" idêntico para as funções UMF e LMF. O máximo grau de ativação da função LMF é ajustado com um valor menor ao estabelecido para UMF.

As funções lineares por partes, são aplicadas nas construções das formas trapezoidais e triangulares, equações (2.17) e (2.18). As funções trapezoidais, Figura 8 (a) e Figura 8 (b), são definidas por seus quatro pontos de intersecção, (a, b, c, d), sendo que "a" e "d" definem a base e "c" e "b" o topo do trapézio, equação (2.17).

$$\mu(x) = \begin{cases} \dfrac{x-a}{b-a}, & a < x < b \\ 1, & b < x < c \\ \dfrac{d-x}{d-c}, & c < x < d \\ 0, & a > x < d \end{cases} \tag{2.17}$$

As funções triangulares são determinadas pelos mesmos princípios das funções trapezoidais, ou seja, utilizando equações lineares por partes. As formas triangular posição, Figura 8 (c), e triangular largura, Figura 8 (f), utilizam a equação (2.18). Essas funções são definidas por três pontos de intersecção, (a, b, c), sendo que "a" e "c" definem a base do triangulo e o ponto "b" o topo do triângulo. $2^2 = 4$

$$\mu(x) = \begin{cases} \dfrac{x-a}{b-a}, & a < x < b \\ \dfrac{c-x}{c-b}, & b < x < c \\ 0, & a > x < c \end{cases} \qquad (2.18)$$

As principais definições de conjuntos *fuzzy* tipo-2 são apresentadas a seguir com base nos trabalhos [7], [27] e [29].

Definição 1: Conjuntos *fuzzy* tipo-2, denominado Ã, é caracterizado por uma função de pertinência tipo-2 $\mu_{\tilde{A}}(x,u)$, onde x ∈ X e u ∈ $J_x \subseteq [0,1]$, Figura 6 (b), ou seja:

$$\tilde{A} = \left\{ \left((x,u), \mu_{\tilde{A}}(x,u) \right) | \forall x \in X, \forall u \in J_x \subseteq [0,1] \right\} \qquad (2.19)$$

Para $0 \leq \mu_{\tilde{A}}(x,u) \leq 1$. Ã pode ser expresso como:

$$\tilde{A} = \int_{x \in X} \int_{u \in [0,1]} \mu_{\tilde{A}}(x,u) / (x,u), \; com \; J_x \subseteq [0,1] \qquad (2.20)$$

Onde ∬ representa a união sobre todos os valores x e u. Para universos discretos, ∫ é substituindo pelo somatório ∑.

Uma restrição fundamental para que um sistema *fuzzy* seja considerado tipo-2 é a presença da incerteza, ou seja, quando as incertezas desaparecem das funções de pertinência tipo-2, o sistema de inferência *fuzzy* tipo-2 é reduzido para um sistema tipo-1.

Outra restrição importante é considerar o fato de que as amplitudes das funções de pertinência devem situar-se entre $0 \leq \mu_{\tilde{A}}(x,u) \leq 1$, ou seja, assumir valores dentro do intervalo de 0 e 1.

Definição 2: As funções de pertinência secundárias do SIF tipo-2 apresenta informações que podem ser tratadas de maneira unitária, portanto, para simplificar os cálculos os valores referentes ao numerador da Equação (2.20**),** assumem valores unitários, $\mu_{\tilde{A}}(x,u) = 1$. Esse procedimento da origem ao sistema de inferência *fuzzy* intervalar. Os gráficos em três dimensões são apresentados na Figura 9(a) para o sistema *fuzzy* tipo-2 e a Figura 9(b) apresenta o sistema *fuzzy* tipo-2 intervalar. A Equação (2.21) apresenta os cálculos para o sistema *fuzzy* tipo-2 intervalar.

$$\tilde{A} = \int_{x \in X} \int_{u \in [0,1]} 1 / (x,u), \; com \; J_x \subseteq [0,1] \qquad (2.21)$$

Devido ao fato de o grau secundário dos conjuntos *fuzzy* do tipo-2 intervalares ser sempre igual a 1, a terceira dimensão acaba não mostrando nenhuma informação adicional; desta forma, o conjunto tipo-2 intervalar pode ser representado apenas por sua FOU. A maioria dos trabalhos utiliza conjuntos *fuzzy* tipo-2 intervalares e SIF tipo-2 intervalares, pois apresentam menor complexidade computacional [7].

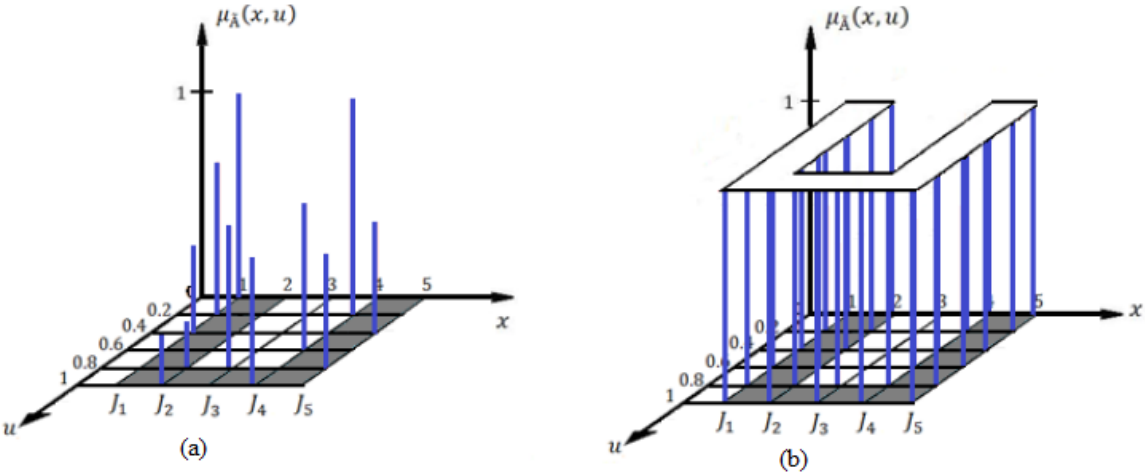


Figura 9: a) Funções de pertinência tipo-2 b) Funções de pertinência tipo-2 intervalar. Fonte: [7]

Definição 3: A representação de um corte vertical determina a função de pertinência secundária $\mu_{\tilde{A}}(x,u)$ para um determinado valor de $x = x'$, essa representação pode ser observada na Figura 10(a) para um SIF tipo-2 não intervalar e na Figura 10(b) SIF tipo-2 intervalar. A partir do conceito de função de pertinência secundária, o corte vertical do conjunto *fuzzy* tipo-2 intervalar pode ser reescrito na equação (2.22).

$$\tilde{A} = \{(x, \mu_{\tilde{A}}(x)) | \forall x \in X\} \qquad (2.22)$$

Se x e j_x forem discretizados em N e $M_i (i = 1,2,3 \ldots\ldots, N)$ a equação (2.22), que representa uma função de pertinência tipo-2 intervalar, pode ser representada da seguinte forma:

$$\tilde{A} = \sum_{x \in X} \frac{\left[\sum_{u \in J_x} \left(\frac{1}{u}\right)\right]}{x} = \sum_{i=1}^{N} \frac{\left[\sum_{u \in J_{x_i}} \left(\frac{1}{u}\right)\right]}{x_i} = \frac{\left[\sum_{k=1}^{M_i} \left(\frac{1}{u_{1k}}\right)\right]}{x_1} + \cdots + \frac{\left[\sum_{k=1}^{M_k} \left(\frac{1}{u_{Nk}}\right)\right]}{x_N} \qquad (2.23)$$

Onde "+" na equação(2.23) significa união. Cada valor de x é discretizado em N valores, e cada um destes valores é discretizado em M_N valores.

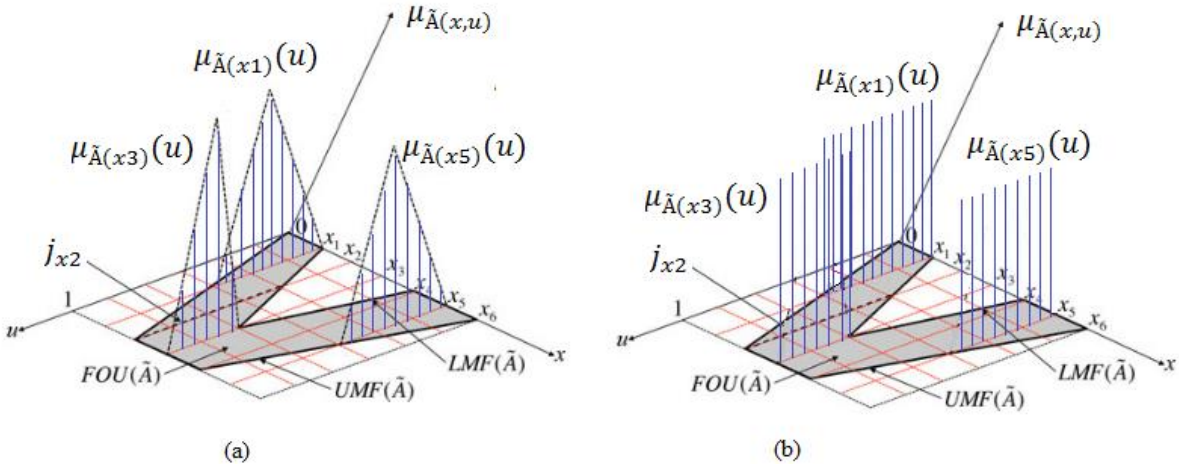


Figura 10: Definição de corte vertical. (a) SIF tipo-2, (b) SIF tipo-2 intervalar. Fonte: [30] *e* [7]

Definição 4: A pertinência primária x (J_x) é definida como o domínio da função de pertinência secundária para um valor de x, com $J_x \subseteq [0,1]$ para $\forall x \in X$.

Definição 5: A amplitude de uma função de pertinência secundária é definida como grau secundário. O grau secundário $\mu_{\tilde{A}}(x,u)$ de uma função de pertinência tipo-2 intervalar é igual à $\mu_{\tilde{A}}(x,u) = 1$.

Definição 6: A incerteza das funções de pertinências primárias de um conjunto *fuzzy* tipo-2 é determinada pela "mancha" de incerteza, *Footprint Of Uncertainty* (FOU). Quando a incerteza desaparece, o conjunto *fuzzy* tipo-2 se torna um conjunto *fuzzy* tipo-1. A "mancha" de incerteza (FOU) é definida como a união de todas as pertinências primárias.

$$FOU(\tilde{A}) = U_{x \in X} J_x \qquad (2.24)$$

Definição 7: A FOU de um conjunto *fuzzy* do tipo-2 é delimitada por uma função de pertinência do tipo-1 superior e uma inferior. A função de pertinência superior é representada na forma $\overline{\mu}_{\tilde{A}}(x)$, para $\forall x \in X$, e a função de pertinência inferior é a função mais interna que limita a FOU(Ã), e é representada na forma de $\underline{\mu}_{\tilde{A}}(x)$, para $\forall x \in X$. A Figura 7 apresenta um modelo de um FOU com a função de pertinência superior nomeada com UMF e a inferior como LMF, na Figura 8 é apresenta as duas categorias de funções de pertinência, gaussiano ou linear por partes.

$$\overline{\mu}_{\tilde{A}}(x) = \overline{FOU(\tilde{A})} = U_{x \in X} \overline{J}_x \qquad \forall x \in X \qquad (2.25)$$

$$\underline{\mu}_{\tilde{A}}(x) = \underline{FOU(\tilde{A})} = U_{x \in X} \underline{J}_x \qquad \forall x \in X \qquad (2.26)$$

O conjunto *fuzzy* tipo-2 intervalar pode ser representado pelas suas funções de pertinência superior e inferior, da seguinte forma:

$$\tilde{A} = \int_{x \in X} [\int_{u \in [\, \overline{\mu}_{\tilde{A}}(x), \underline{\mu}_{\tilde{A}}(x)\,]} 1/u] \,/x \qquad (2.27)$$

As operações união, interseção e complemento utilizadas em conjuntos *fuzzy* tipo-2 serão apresentadas a seguir. Todos os resultados necessários para implementar um SIF tipo-2 são obtidos usando as operações matemática já apresentadas para o SIF tipo-1 na seção (2.2), entretanto, as operações para SIF tipo-2 são duplicadas em diversos momentos [31].

A união de dois conjuntos é a definição de um outro conjunto que contém os elementos em Ã ou $\tilde{B}$. Cada elemento de um SIF tipo-2 é definido em um subconjunto, onde os termos desse conjunto são distinto [31].

A união dos conjuntos *fuzzy* Ã e $\tilde{B}$ é denotada por Ã + $\tilde{B}$ ou Ã U $\tilde{B}$. A Equação (2.28) define a operação de união nos conjuntos *fuzzy* tipo-2.

$$\tilde{A} \cup \tilde{B}(x) = 1/[\underline{\mu}_{\tilde{A}}(x) \vee \underline{\mu}_{\tilde{B}}(x), \overline{\mu}_{\tilde{A}}(x) \vee \overline{\mu}_{\tilde{B}}(x)] \quad \forall x \in X \qquad (2.28)$$

Onde ∨ é a representação do operador máximo.

A intersecção entre os conjuntos *fuzzy* tipo-2 $\tilde{A}$ e $\tilde{B}$ é feita da mesma forma que a união. A equação (2.29) apresenta a intersecção de dois conjuntos *fuzzy* $\tilde{A}$ e $\tilde{B}$.

$$\tilde{A} \cap \tilde{B}(x) = 1/[\underline{\mu}_{\tilde{A}}(x) \wedge \underline{\mu}_{\tilde{B}}(x), \quad \overline{\mu}_{\tilde{A}}(x) \wedge \overline{\mu}_{\tilde{B}}(x)] \quad \forall x \in X \qquad (2.29)$$

O complemento do conjunto *fuzzy* tipo-2 intervalar $\tilde{A}$, é dado pela equação (2.30).

$$\overline{\tilde{A}} = 1/[1 - \overline{\mu}_{\tilde{A}}(x), 1 - \underline{\mu}_{\tilde{A}}(x)] \quad \forall x \in X \qquad (2.30)$$

Um exemplo de complemento de um conjunto *fuzzy* tipo-2 intervalar é mostrado na Figura 11. O complemento de um conjunto é dado pelo valor do máximo grau de pertinência da função menos o valor da função.

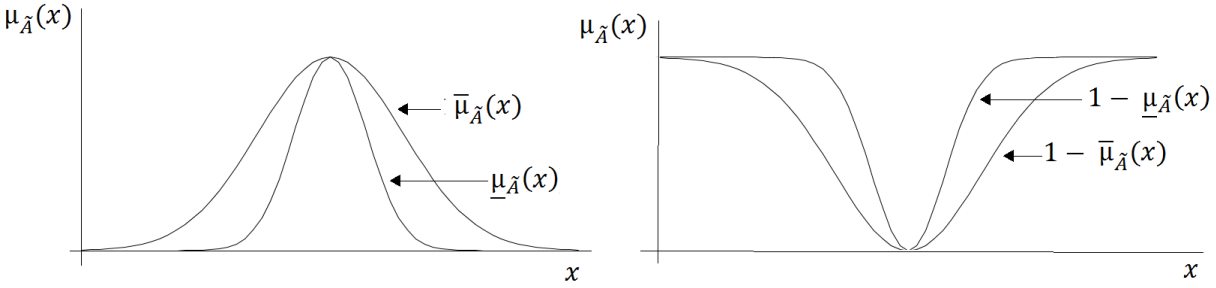


Figura 11: Complemento de conjuntos fuzzy tipo-2 intervalar

2.4. Controladores *Fuzzy*

2.4.1. Controlador *fuzzy* tipo-1

O sistema de inferência *fuzzy* imita o processo seguido pela mente humana ao executar ações de controle ou de tomada de decisão baseado em palavras, por exemplo: pressione o pedal de aceleração levemente. A pessoa então executa uma ação de pressionar o pedal de aceleração levemente. Essa ação pode representar 10% de acionamento do pedal. O sistema de inferência *fuzzy* converte as informações linguísticas em informações de controle. A Figura 12 apresenta um SIF tipo-1que é composto por quatro componentes principais: fuzificador, base de regras, mecanismo de inferência e defuzificador [29].

A fuzificação mapeia um vetor de entrada *crisp* em um grupo de conjuntos *fuzzy* definidos para cada entrada. O fuzificador pode utilizar funções *singleton*.

A Figura 13 apresenta uma fuzificação *singleton* em uma função de pertinência gaussiana. O valor de entrada *crisp* fornece a posição do *singleton* e a função de pertinência estabelece a amplitude *singleton* [32].

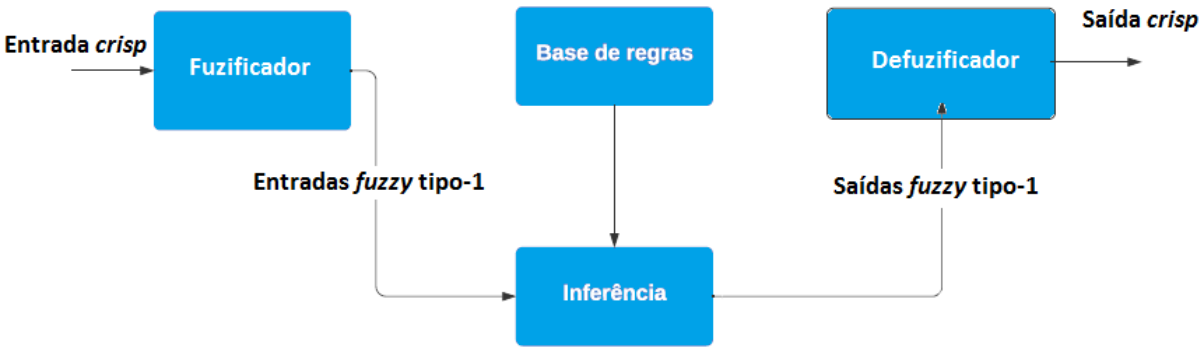


Figura 12: Sistema de inferência fuzzy tipo1.

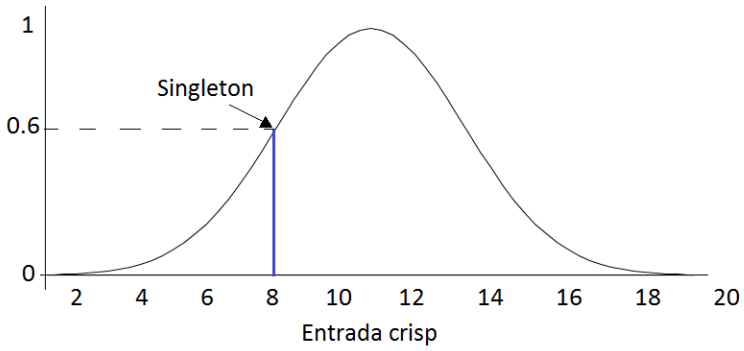


Figura 13: Fuzificação singleton sobre uma função de pertinência gaussiana

A base de regras é o cérebro do sistema *fuzzy*. A definição das regras pode ser feita por um especialistas ou extraídos de dados reais. Cada regra é uma declaração IF-THEN. Cada parte IF de uma regra é chamada de antecedente e a parte THEN é chamada de consequente. As regras são relacionam aos conjuntos *fuzzy* de entrada e aos conjuntos *fuzzy* de saída. Todas as regras são coletadas em uma base de regras [29]. A Equação (2.31) mostra que SE uma premissa, relacionada as funções antecedentes, for atendida; ENTÃO uma ação será executada nas funções consequentes. A Tabela 1 apresenta o formato de um base de regras.

$$IF\ premissa\ (antecedente), THEN\ conclusão\ (consequente) \tag{2.31}$$

Tabela 1: Base de regras do sistema fuzzy. Adaptada de:[2].

Regra 1:	IF premissa P^1, THEN conclusão C^1
Regra 2:	IF premissa P^2, THEN conclusão C^2
⋮	
Regra r:	IF premissa P^r, THEN conclusão C^r

O circuito de inferência combina e calcula as regras, fornecendo um mapeamento dos conjuntos antecedentes para os conjuntos consequentes. Existem várias opções para executar as operações antecedentes e consequentes exigidas pelo mecanismo de inferência [13],

entretanto, será abordado neste trabalho somente os métodos de Mamdani dos mínimos e máximo, Figura 14, e o modelo de Larsen, Figura 15.

Tanto para o método de Mamdani, Figura 14, como para o de Larsen, Figura 15, as formas A11 e A12 referem se ao primeiro e ao segundo conjunto antecedentes *fuzzy* da primeira regra. A forma B1 refere-se ao conjunto consequente *fuzzy* da primeira regra. Já as formas A21 e A22 se referem ao primeiro e ao segundo conjuntos antecedentes da segunda regra. A forma B2 refere-se ao conjunto consequente da segunda regra. O modelo de Mamdani apresentado na Figura 14 e exposto na Equação(2.32) determina o menor grau de ativação "min" entre as funções antecedente utilizando a operação de intersecção, $A_{11} \cap A_{12}$ e $A_{21} \cap A_{22}$. Para determinar o maior grau de ativação "max" entre as funções consequentes é utilizado a operação de união, $B_1 \cap B_2$.

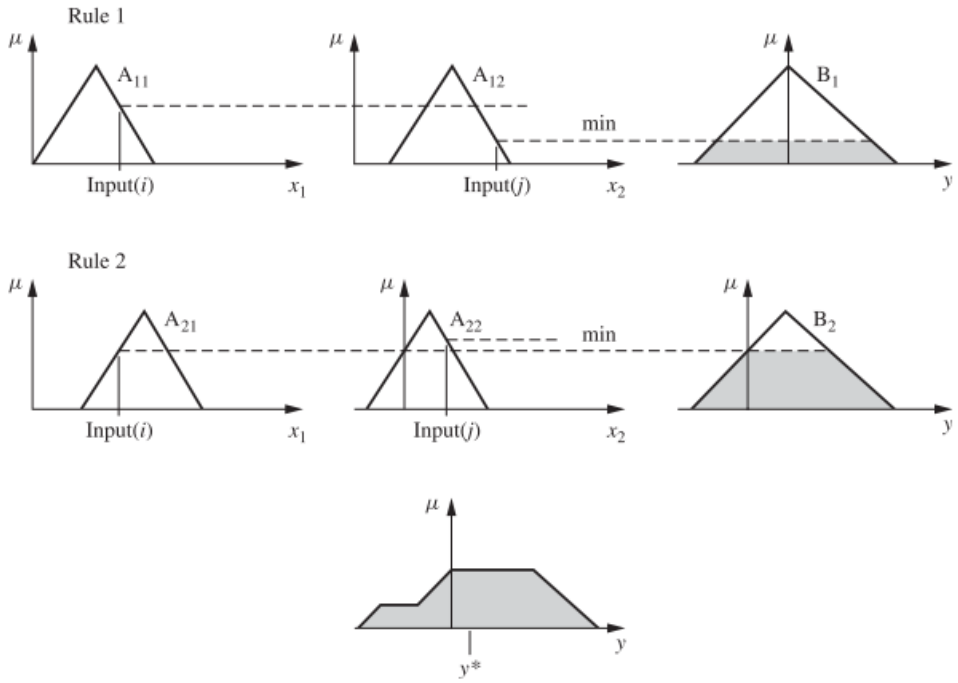


Figura 14: Modelo de Mamdani. Fonte: [2].

$$\mu_{B^K}(y) = \max[\min\left[\mu_{A_1^K}\left(\text{input}(i)\right), \mu_{A_2^K}\left(\text{input}(j)\right)\right], \ k = 1, 2, \ldots \text{R} \qquad (2.32)$$

Já o método de Larsen apresentado na Equação (2.33) e exposto na Figura 15, determina as funções consequentes utilizando a operação de multiplicação "produto" entre as funções antecedentes, $A_{11} * A_{12}$ e $A_{21} * A_{22}$. Para determinar o maior grau de ativação "max" entre as funções consequentes é utilizado a operação de união, $B_1 \cap B_2$, assim como o método de Mamdani.

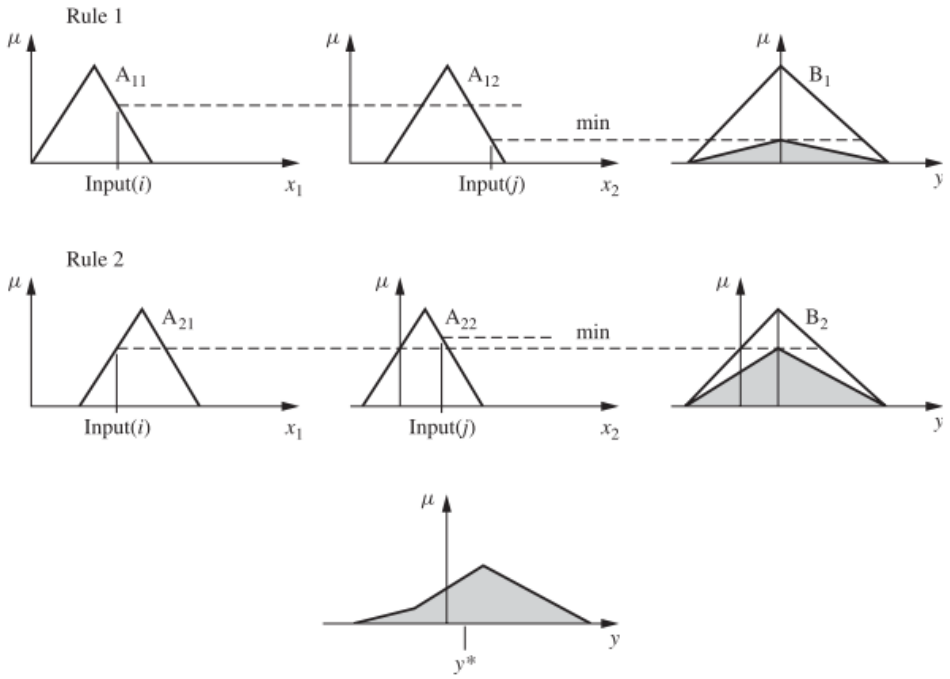


Figura 15: Modelo de Larsen. Fonte:[2]

$$\mu_{B^K}(y) = \max[\,\mu_{A_1^K}\left(\text{input}(i)\right) * \mu_{A_2^K}\left(\text{input}(j)\right)],\ \ k = 1, 2, \ldots \text{R} \qquad (2.33)$$

A defuzificação reduz os conjuntos *fuzzy*, da saída do mecanismo de inferência, em uma única saída *crisp*. Existem vários métodos de defuzificação, dentre os quais são citados os métodos mais utilizados no (Apêndice-B), os quais são:

- Método da Altura.
- Método centro de área.
- Primeiro do Máximo.
- Último do Máximo.
- Média do Máximo.
- Centro de soma.

2.4.2. Controlador *fuzzy* tipo-2

O diagrama em blocos do Sistema de Inferência *Fuzzy* (SIF) tipo-2, também denominado de controlador *fuzzy*, é apresentado na Figura 16. Este sistema é composto por cinco blocos: o fuzificador, a inferência, a base de regras, redutor de tipo e o defuzificador. A estrutura do controlador *fuzzy* tipo-2 é similar à do controlador *fuzzy* tipo-1, sendo a principal diferença o processamento de saída que além da etapa de defuzificação, encontrada no controlador *fuzzy* tipo-1, apresenta ainda uma etapa de redução de tipo, conforme pode ser observado na Figura 16.

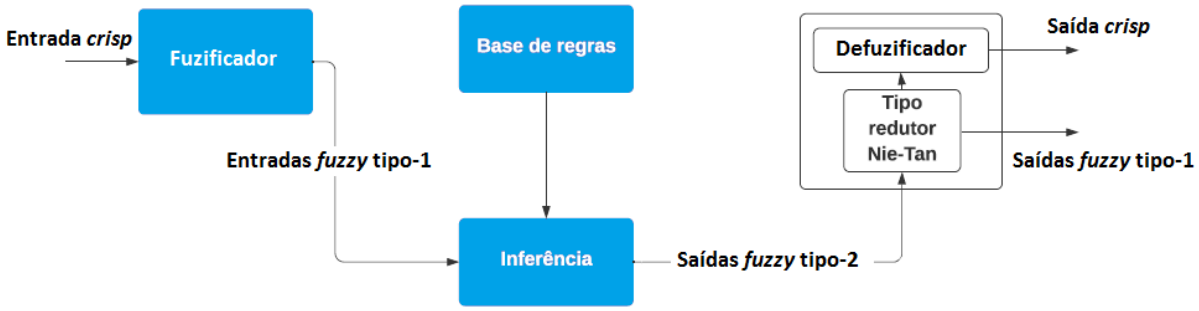


Figura 16: Sistema de Inferência Fuzzy Tipo-2 Intervalar

Conforme já visto no controlador *fuzzy* tipo-1, o **bloco fuzificador** tem o papel de associar os valores das entradas aos graus de pertinências modelados pelas funções de pertinência de entrada. Para os controladores *fuzzy* tipo-2 intervalares, o valor de pertinência atribuído a cada entrada pode ser representado por duas funções de pertinência, superior e inferior, que delimitam a mancha de incerteza (FOU), conforme descrito na seção (2.3).

A base de regras para SIF tipo-2 é a mesma utilizada no SIF tipo-1, a qual é abordada na seção (2.4.1).

O mecanismo de inferência opera de acordo com a base de regras e não depende da natureza dos conjuntos *fuzzy*. Tanto para *fuzzy* tipo-1 como para o tipo-2, a inferência funciona da mesma maneira. O mecanismo de inferência combina e calcula as regras, fornecendo um mapeamento dos conjuntos antecedentes para os conjuntos consequentes. Existem várias opções para executar as operações antecedentes e consequentes exigidas pelo mecanismo de inferência [13]. O modelo do mecanismo utilizado neste trabalho é o operador *t-norma* mínimo de Mamdani, que é representado na Figura 17. O funcionamento dos operadores *t-norma* mínimo de Mamdani e de Larsen estão detalhados na seção (2.4.1) nas figuras 14 e 15 para SIF tipo-1. Para um SIF tipo-2 intervalar é necessário acrescentar o operador *t-norma* tanto para o limite inferior da mancha de incerteza quanto para o limite superior. As Figuras 17 e 18 apresentam um exemplo de operação do SIF tipo-2 intervalar para os operadores de *t-norma* de Mamdani e Larsen, respectivamente.

A redução de tipo transforma os conjuntos de saída *fuzzy* tipo-2, determinados no mecanismo de inferência, para um conjuntos *fuzzy* tipo-1 [27]. Os quatros algoritmos mais utilizados na redução de tipo são: Karnik-Mendel (KM) [9], Wu-Mendel (WM) [12], Biglarbegian–Melek–Mendel (BMM) [15] e Nie-Tan (NT) [14]. Nos métodos de KM e WM, o redutor de tipo gera uma saída de conjuntos *fuzzy* do tipo-1, que então são convertida em uma saída *crips* por um defuzificador [27]. O processo de saída dos algoritmos de BMM e NT apresentam uma estrutura mais simples que dispensa o circuito dedicado a defuzificação [19].

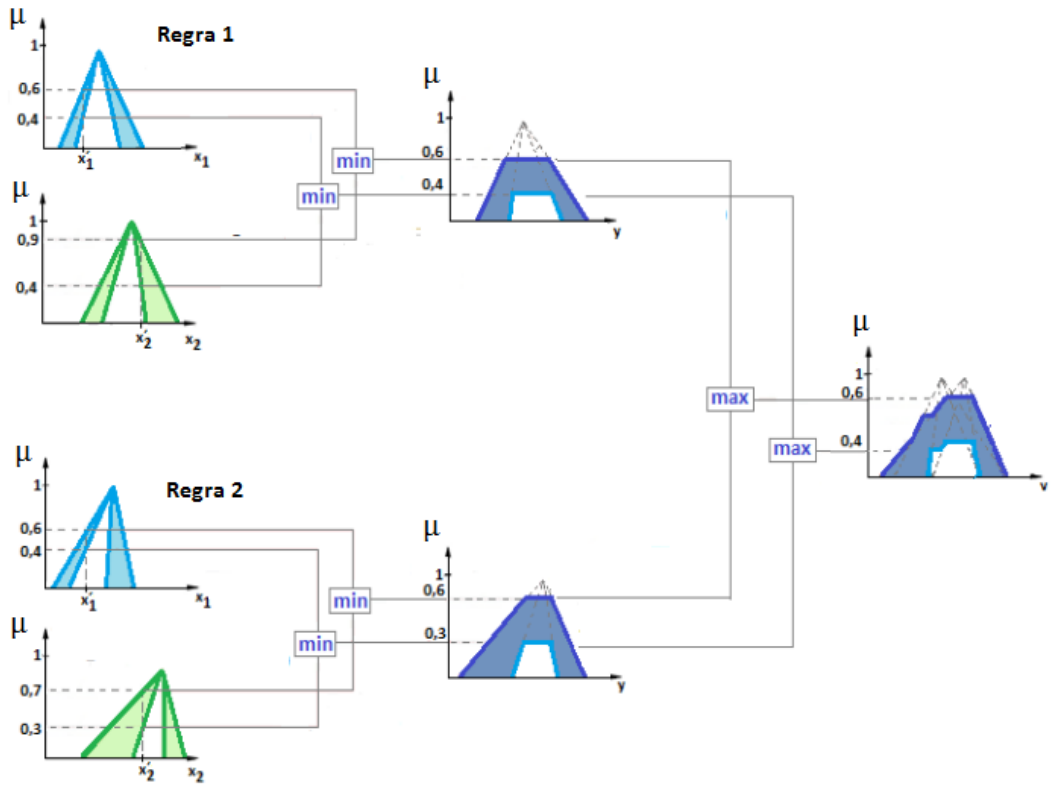


Figura 17: Fuzzy tipo-2 intervalar utilizando o operador t-norma Mamdani. Fonte [7].

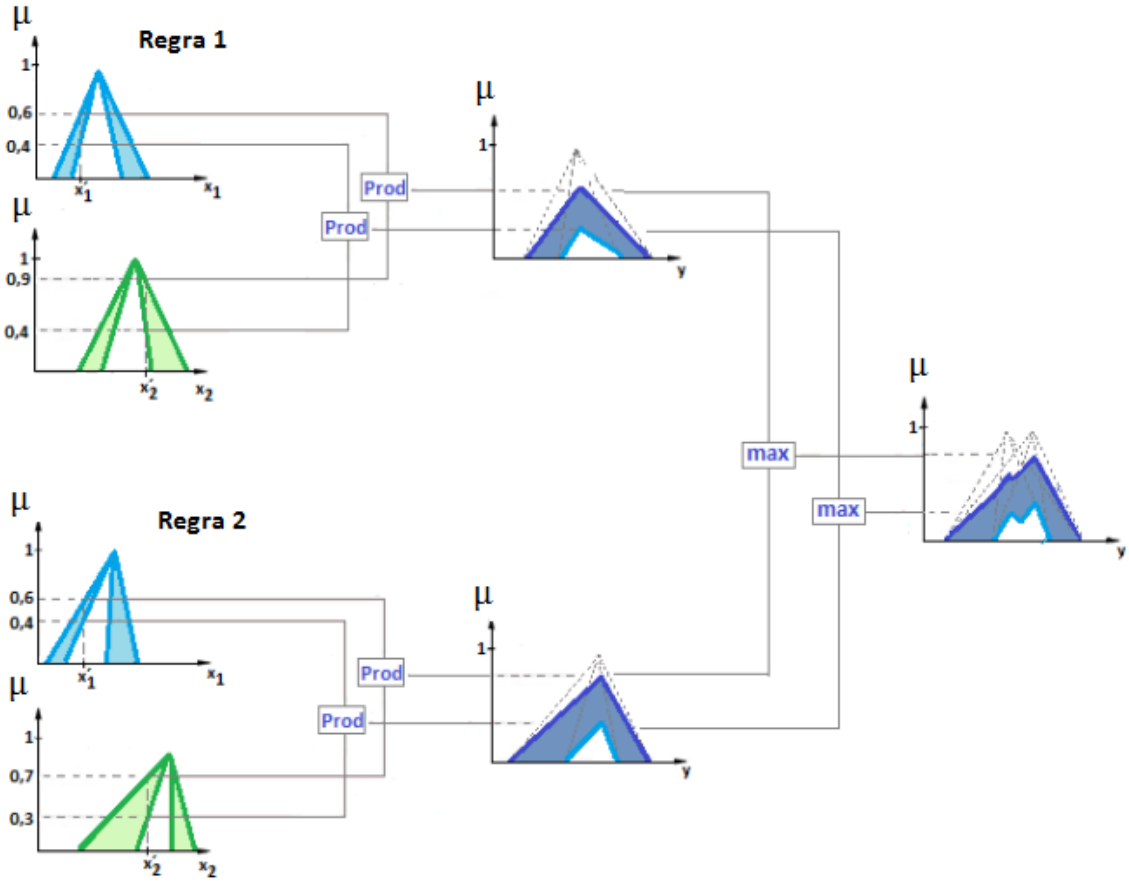


Figura 18: Fuzzy tipo-2 intervalar utilizando o operador t-norma Larsen. Fonte [7].

O algoritmo KM utiliza iterações para calcular os valores exatos dos pontos finais esquerdo e direito,y_l *e* y_r utilizando as seguintes equações e procedimentos:

1. Suponha que y_r^i sejam organizados em ordem crescente $y_r^1 \leq y_r^2 \leq \cdots \leq y_r^M$.
2. Equação (2.34) apresenta os calculo para y_r.

$$y_r = \frac{\sum_{i=1}^{M} f_r^i \, y_r^i}{\sum_{i=1}^{M} f_r^i} \qquad (2.34)$$

Onde $f_r^i = \frac{\bar{f}^i + \overline{f^i}}{2}$ para $i = 1, \ldots., M$, e $y'_r \equiv y_r$.

3. Determine $R(1 \leq R \leq M-1)$ de tal maneira que $y_r^R \leq y'_r \leq y_r^{M=1}$.
4. Calcule y_r da equação (2.34) com $f_r^i = \overline{f^i}$ para $i < R$ e $f_r^i = \overline{f}^i$ para $i > R$, e $y''_r \equiv y_r$.
5. Se $y''_r \neq y'_r$, então vá para passo 6. Se $y''_r = y'_r$ para no conjunto $y_r \equiv y''_r$.
6. Se $y'_r = y''_r$ então retorne para o passo 3.

O algoritmo para calcular y_l é semelhante ao do algoritmo y_r, Finalmente, a defuzificação é determinada com a média dos valores estabelecidos em y_l e y_r como segue:

$$Y_{KM} = \frac{y_r + y_l}{2} \qquad (2.35)$$

O algoritmo de WM é usado para aproximar o conjunto redutor de tipos por limites de incerteza min-máx., onde $\bar{y}_l \leq y_l \leq \underline{y}_l$ e $\bar{y}_r \leq y_r \leq \underline{y}_r$ [19]. Os limites de incerteza são definidos nas equações (2.36), (2.37), (2.38), (2.39) e (2.40).

$$\bar{y}_l = min\left\{\frac{\sum_{i=1}^{M} \underline{f}^i y_l^i}{\sum_{i=1}^{M} \underline{f}^i}, \quad \frac{\sum_{i=1}^{M} \overline{f}^{\,i} y_l^i}{\sum_{i=1}^{M} \overline{f}^{\,i}}\right\} \qquad (2.36)$$

$$\underline{y}_r = máx\left\{\frac{\sum_{i=1}^{M} \overline{f}^{\,i} y_r^i}{\sum_{i=1}^{M} \overline{f}^{\,i}}, \quad \frac{\sum_{i=1}^{M} \underline{f}^i y_r^i}{\sum_{i=1}^{M} \underline{f}^i}\right\} \qquad (2.37)$$

$$\underline{y}_l = \bar{y}_l - \left[\frac{\sum_{i=1}^{M} \left(\overline{f}^{\,i} - \underline{f}^i\right)}{\sum_{i=1}^{M} \overline{f}^{\,i} \sum_{i=1}^{M} \underline{f}^i} \text{x} \frac{\sum_{i=1}^{M} \underline{f}^i (y_l^i - y_l^1) \sum_{i=1}^{M} \overline{f}^{\,i} (y_l^M - y_l^i)}{\sum_{i=1}^{M} \underline{f}^i (y_l^i - y_l^1) + \sum_{i=1}^{M} \overline{f}^{\,i} (y_l^M - y_l^i)}\right] \qquad (2.38)$$

$$\bar{y}_r = \underline{y}_r - \left[\frac{\sum_{i=1}^{M} \left(\overline{f}^{\,i} - \underline{f}^i\right)}{\sum_{i=1}^{M} \overline{f}^{\,i} \sum_{i=1}^{M} \underline{f}^i} \text{x} \frac{\sum_{i=1}^{M} \underline{f}^i (y_r^i - y_r^1) \sum_{i=1}^{M} \overline{f}^{\,i} (y_r^M - y_r^i)}{\sum_{i=1}^{M} \underline{f}^i (y_r^i - y_r^1) + \sum_{i=1}^{M} \overline{f}^{\,i} (y_r^M - y_r^i)}\right] \qquad (2.39)$$

$$Y_{WM} = \frac{1}{2}\left[\frac{\bar{y}_l + \underline{y}_l}{2} + \frac{\bar{y}_r + \underline{y}_r}{2}\right] \quad (2.40)$$

Com os conjuntos reduzidos ao modelo tipo-1 é utilizado o método da altura para realizar a defuzificação

O algoritmo de NT apresenta um operação matemática mais simples e que apresenta uma defuzificação direta. A equação (2.41) estabelece as operações matemáticas para o algoritmo de NT.

$$Y_{NT} = \frac{\sum_{i=1}^{M} y^i(\bar{f}^i + \underline{f}^i)}{\sum_{i=1}^{M} (\bar{f}^i + \underline{f}^i)} \quad (2.41)$$

O algoritmo de BMM é apresentado na equação (2.42), onde os parâmetros "m" e "n" são usados para ajustar o sistema. Esse algoritmo também apresenta uma defuzificação direta, assim como o algoritmo de NT.

$$Y_{BMM} = m\frac{\sum_{i=1}^{M} \underline{f}^i y^i}{\sum_{i=1}^{M} \underline{f}^i} + n\frac{\sum_{i=1}^{M} \bar{f}^i y^i}{\sum_{i=1}^{M} \bar{f}^i} \quad (2.42)$$

Os resultados apresentados em [19], mostram que os métodos de NT e BMM possuem maior velocidade de processamento e menor custo de hardware do que os algoritmos KM e WM. Porém, o controlador NT e BMM produz um *overshoot* com um erro de ajuste na casa de 2%.

O bloco defuzificador para um SIF tipo-2 gera a saída *crisp* a partir dos conjuntos *fuzzy* tipo-1 da saída do bloco de redução de tipo. Os principais métodos de defuzificação são abordados no (Apêndice-B).

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

Os sistemas de controle baseados na lógica fuzzy tipo-2 são capazes de trabalhar com incerteza e quando comparados aos sistemas de controle fuzzy tipo-1, apresentam as seguintes vantagens [33].

- A função de pertinência fuzzy tipo-2 contém a "mancha" de incerteza (FOU), que é capaz de trabalhar com incertezas nas funções de pertinência de entrada e saída do controlador fuzzy;

- A utilização de funções de pertinência fuzzy tipo-2 na entrada de controladores fuzzy pode resultar na diminuição do número de regras da base de regras quando comparado com sistema fuzzy tipo-1;
- O controlador fuzzy tipo-2 é capaz de responder a sistemas que não podiam ser controlados pelo sistema fuzzy tipo-1 com o mesmo número de funções de pertinência [34].

Em suma, o controlador fuzzy tipo-2 é utilizado nos seguintes casos [35]:

- Em aplicações onde existe incerteza na determinação exata do grau de pertinência, como por exemplo, em casos de dados ruidosos.
- Em aplicações onde não existe alta confiança no modelo ou quando é difícil determinar o modelo adequado em função da não linearidade, não estacionariedade ou variância no tempo.

2.6. Considerações finais

Neste capítulo foram apresentadas as principais definições e terminologias sobre os conjuntos *fuzzy* tipo-1 e tipo-2, os módulos constituintes do sistema de inferência *fuzzy* tipo-1 e tipo-2 intervalar e as estruturas dos controlados *fuzzy* tipo-1 e tipo-2 intervalar. 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 apresentada a arquitetura de um Sistema de Inferência *Fuzzy* (SIF) tipo-2 intervalar implementado na FPGA EP4CE115F29C7 utilizando linguagem de programação *Verilog*. O diagrama da arquitetura em questão é apresentado na Figura 19 e tem como característica duas entradas de 8 *bits* e uma saída de 8 *bits*, sendo composto pelos seguintes módulos: um fuzificador tipo-2 baseado em funções gaussianas *singleton*, um mecanismo de inferência baseado no método de Mamdani, uma base de regras para 16 regras e um redutor de tipo baseado no algoritmo de Nie-Tan. As especificações, o projeto e a implementação de todos os circuitos que compõem o SIF tipo-2 intervalar também serão apresentados e discutidos neste capítulo.

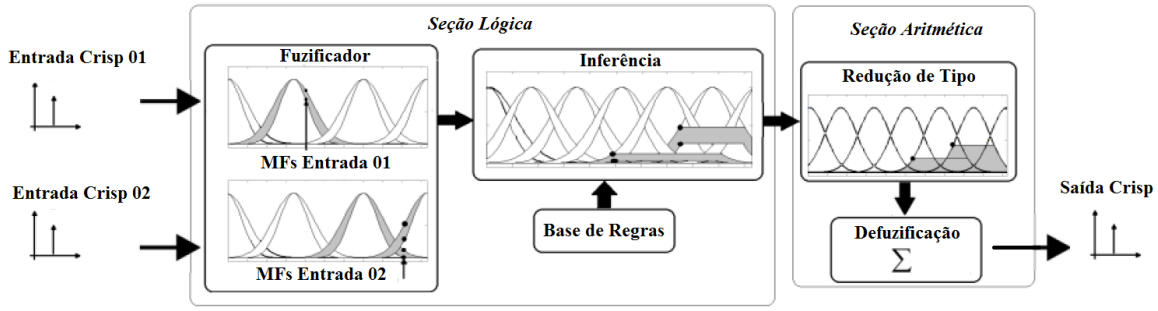


Figura 19: Sistema de inferência fuzzy tipo-2

3.2. Especificação

O *hardware* apresentado consiste em duas entradas de 8 bits com quatro funções de pertinência gaussianas para cada entrada, dezesseis regras e uma saída de 8 bits com sete funções de pertinência. O *hardware* se mostrou compacto e com possibilidade de configuração do sistema *fuzzy* tipo-2 ou *fuzzy* tipo-1. O número de *Fuzzy logic Inference Per Second* (FLIPS) se mostrou superior ao valor mínimo estabelecido para aplicações em tempo real [22].

A Figura 20 apresenta a arquitetura do *hardware* programado em FPGA para implementar o SIF tipo-2 intervalar. O bloco fuzificador utiliza em cada entrada quatro funções de pertinência tipo-2. O bloco de inferência tem como entrada as informações do grau de ativação das funções antecedentes, fornecidas pelo bloco de fuzificação, as quais estão divididas da seguinte maneira: quatro funções da parte superior do *Footprint Of Uncertainty* (FOU) para entrada 1, quatro funções da parte inferior do FOU para entrada 1, quatro funções da parte

superior do FOU para entrada 2 e quatro funções da parte inferior do FOU para entrada 2. Cada circuito FOU do bloco fuzificador possui um bit para identificar a ativação da função, essa informação é direcionada a base de regras que identifica as regras ativas e gerencia o bloco de inferência. A arquitetura apresentada pode operarar com até 16 regras, dependendo da configuração estabelecida no bloco fuzificador, tendo até sete funções consequentes na saída do bloco de inferência. As funções consequentes são direcionadas a seção aritmética que aplica o algoritmo de Nie-Tan para calcular o valor de saída *crisp*. A configuração dos FOUs no bloco de fuzificação estabelece a quantidade de funções de pertinência em cada entrada, a posição da função no universo de discurso e largura do FOU.

As próximas subseções abordam as características construtivas e dinâmicas de cada bloco que compõem a Figura 20

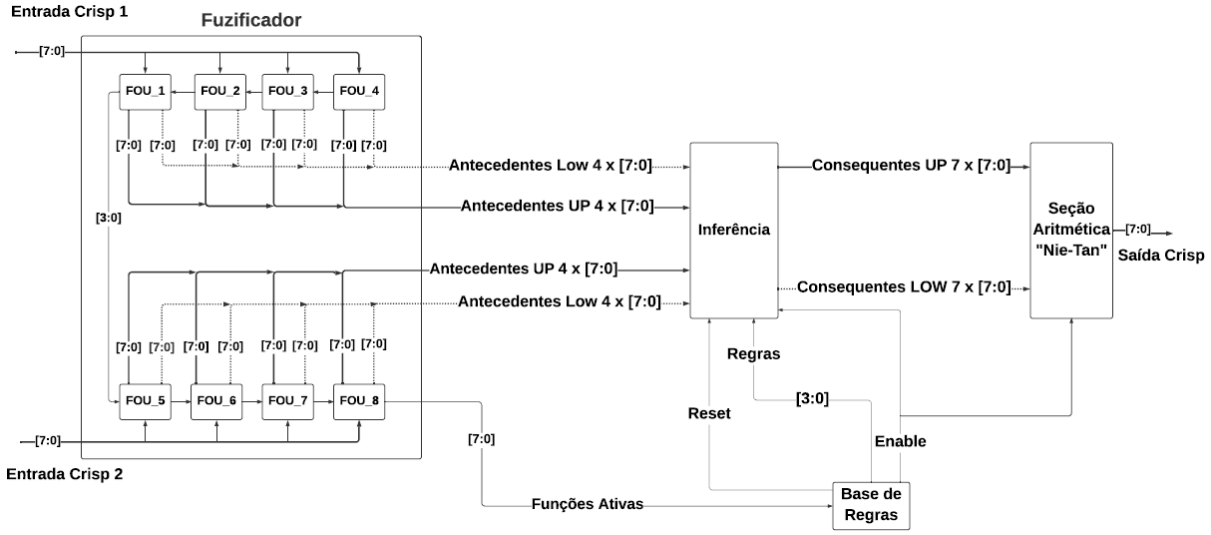


Figura 20: Arquitetura SIF-tipo-2 intervalar

3.2.1. Fuzificador

O fuzificador para um SIF tipo-2 intervalar utiliza funções *fuzzy* tipo-1 para definir os limites inferior e superior do *Footprint Of Uncertainty* (FOU). Os valores numéricos para os parâmetros do FOU são especificados por um especialista e podem ser otimizados durante o ajuste do controlador. Na proposta apresentada para o SIF tipo-2 intervalar, o modelo das funções antecedentes, estabelecidas no fuzificador, é replicado para as funções consequentes definidas no bloco de inferência, o restante do *hardware* não sofre influência pelo tipo de função utilizada.

3.2.1.1. Circuito Função Gaussiana

O bloco de fuzificação utiliza funções gaussianas *fuzzy* tipo-1 *singleton* para determinar os limites superior e inferior do FOU. O circuito utilizado para estabelecer as funções *fuzzy* tipo-1 é apresentado na *Figura 21*, o qual utiliza dois polinômios de segunda ordem para implementar cada função gaussiana [25]. Os polinômios são conectados no ponto de flexão "σ" e "m" da função Gaussiana, equação (3.1), onde "σ" é o desvio padrão e "m" é o centro da função (*Figura 22*). Os coeficientes polinomiais listados na *Tabela 2* dependem da variação do desvio padrão "σ" [25]. Na *Figura 21* os coeficientes polinomiais são representados por a1, a2, b1, b2, c1 e c2, o valor do desvio padrão "σ" é estabelecido nas entradas dos comparadores Comp1 e Comp2, o centro "m" da função é determinado na entrada nomeada como "Centro" e o valor *crisp* da função é inserido na entrada nomeada como "*Crisp*".

$$\mu(x) = e^{-\frac{(x-m)^2}{2\sigma^2}} \qquad (3.1)$$

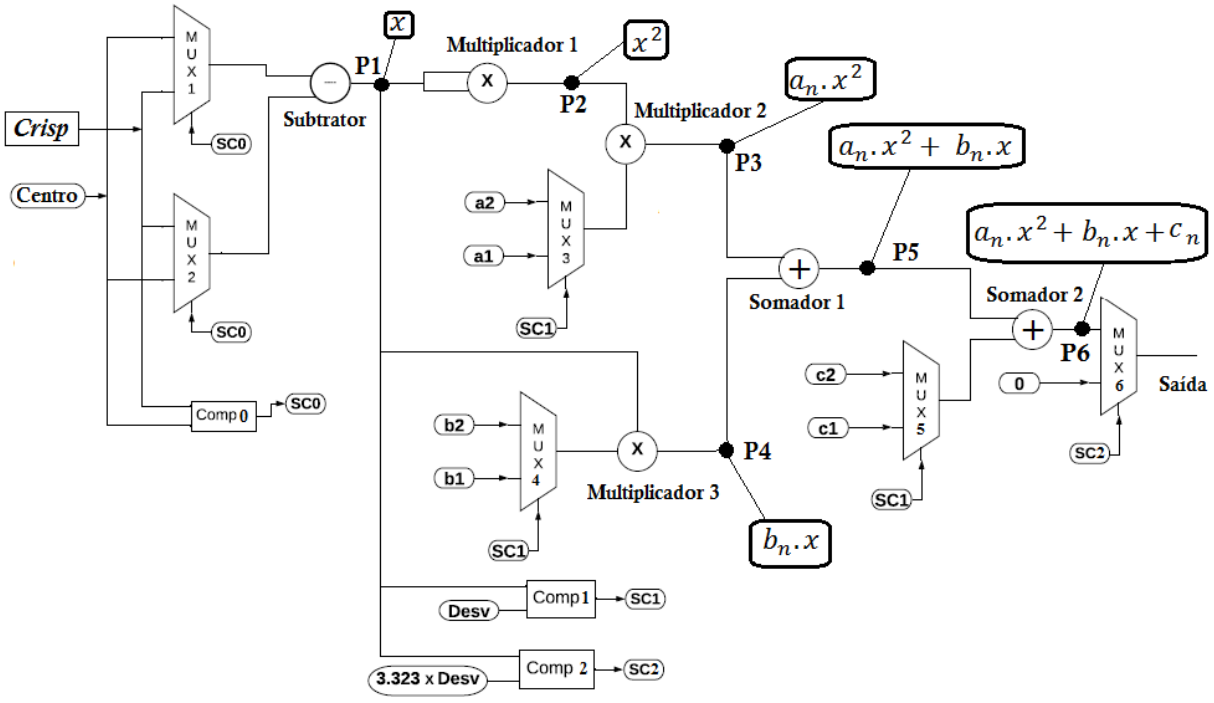


Figura 21: Circuito para implementação da função gaussiana a partir de funções polinomiais. Fonte [25].

A primeira parte do circuito da Figura 21, composto pelos multiplexadores MUX1 e MUX2, o comparador Comp0 e o Subtrator, tem como objetivo calcular a diferença entre o valor da entrada "*Crisp*" e o centro da função "m". O circuito de subtração tem como entrada as saída dos multiplexadores MUX1 e MUX2. Cada multiplexador tem sua entrada de seleção ativa pela saída "SC0" do circuito comparador "Comp0". Quando a entrada "C*risp*" é maior que a entrada "Centro", o circuito comparador "Comp0" mantém sua saída "SC0 = 1", quando

o valor *"Crisp*" é menor que o valor "Centro" "SC0 = 0". As entradas dos multiplexadores, MUX1 e MUX2, são posicionadas para que a entrada de seleção "SC0" mantenha a saída do MUX1 com o maior valor e a do MUX2 com o menor valor, sendo assim, o resultado da subtração sempre vai apresentar o valor em módulo da diferença entre a entrada *"Crisp*" e o valor de "Centro" da função. A diferença entre os dois valores é a variável "x", identificada no ponto "P1" da *Figura 21*, esse valor é utilizado para determinar "x^2" na saída do Multiplicador1, identificado pelo ponto P2 na *Figura 21*. Os valores de "x" e "x^2" são utilizados para determinar os valores das funções $f_1(x)$ e $f_2(x)$ da Tabela 2. A variável "x" também é utilizada nos circuitos comparadores Comp1 e Comp2.

Tabela 2: Coeficiente polinomial. Fonte [25].

Coeficiente polinomial	**Polinômio $f_1(x)$**	**Polinômio $f_2(x)$**
$a.x^2$	$-\frac{39.453}{\sigma^2}$	$\frac{11.328}{\sigma^2}$
$b.x$	0	$-\frac{74.6875}{\sigma}$
c	100	124.1171

O circuito comparador "Comp1" estabelece sua saída "SC1=1" quando o valor de "x" é maior que o valor de desvio padrão "σ" e mantém "SC1=0" quando "x" é menor que "σ". A informação de "SC1" é utilizada como entrada de seleção dos multiplexadores MUX3, MUX4 e MUX5, os quais possuem em suas entradas de dados os polinômios a1 e a2 para o MUX3, b1 e b2 para o MUX4 e c1 e c2 para o MUX5. Portanto, quando "SC1=1" o polinômio a1 é direcionado para saída do MUX3, b1 para saída do MUX4 e c1 para a saída do MUX5, esses polinômios são referentes a $f_1(x)$. Quando "SC1=0" o polinômio a2 é direcionado para saída do MUX3, b2 para saída do MUX4 e c2 para a saída do MUX5, esses polinômios são referentes a $f_2(x)$.

Já o circuito comparador "Comp2" estabelece sua saída "SC2=1" quando o valor de "x" é maior que "3,3 × σ" e mantém "SC2=0" quando "x" é menor que "3,3 × σ". A informação "SC2" é utilizada como entrada de seleção do multiplexador MUX6, o qual tem como finalidade definir a saída com valor zero para os valores de "x" que ultrapassar "3,3 × σ" e manter o resultado do polinômio "$a_n x^2 + b_n x + c_n$" para valores de "x" menores que "3,3 × σ".

Finalizando as operações apresentadas na *Figura 21*, os operadores de multiplicação e soma tem como finalidade construir a equação polinomial resultante na saída para cada entrada *Crisp*. O "Multiplicado 1" estabelece o valor de "x^2" no ponto P2, o "Multiplicado 2" estabelece

o valor de "$a_n x^2$" no ponto P3, o "Multiplicado 3" estabelece o valor de "$b_n x$" no ponto P4, o "Somador 1" estabelece o valor de "$a_n x^2 + b_n x$" no ponto P5 e por fim o "Somador 2" estabelece o valor de "$a_n x^2 + b_n x + c_n$" no ponto P6.

A Figura 22 apresenta um gráfico do Grau de pertinência de um função gaussiana relacionado com a Entrada *Crisp*(x). O exemplo em questão utiliza as operações matemáticas estabelecidas no circuito da Figura 21 configuradas com um desvio padrão $\sigma = 10$ e centro $m = 33$. As equações (3.2) e (3.3) estabelecem os polinômios para as funções $f_1(x)$ e $f_2(x)$.

$$f_1(x) = -0{,}39453.x^2 + 0.x + 100 \quad (3.2)$$

$$f_2(x) = 0{,}11328.x^2 - 7{,}46875.x + 124{,}1171 \quad (3.3)$$

No gráfico da *Figura 22* as funções identificadas como $f_1(x)$ *e* $f_2(x)$ utilizam o valor de "x" para *Crisp(x)* < *m*, equação(3.4), e as funções $f'_2(x)$ *e* $f'_1(x)$ utilizam o valor de "x" para *Crisp(x)* > *m*, equação (3.5).

$$Para\ Crisp(x) < m\ \ x = m - Crisp \quad (3.4)$$

$$Para\ Crisp(x) > m\ \ x = Crisp - m \quad (3.5)$$

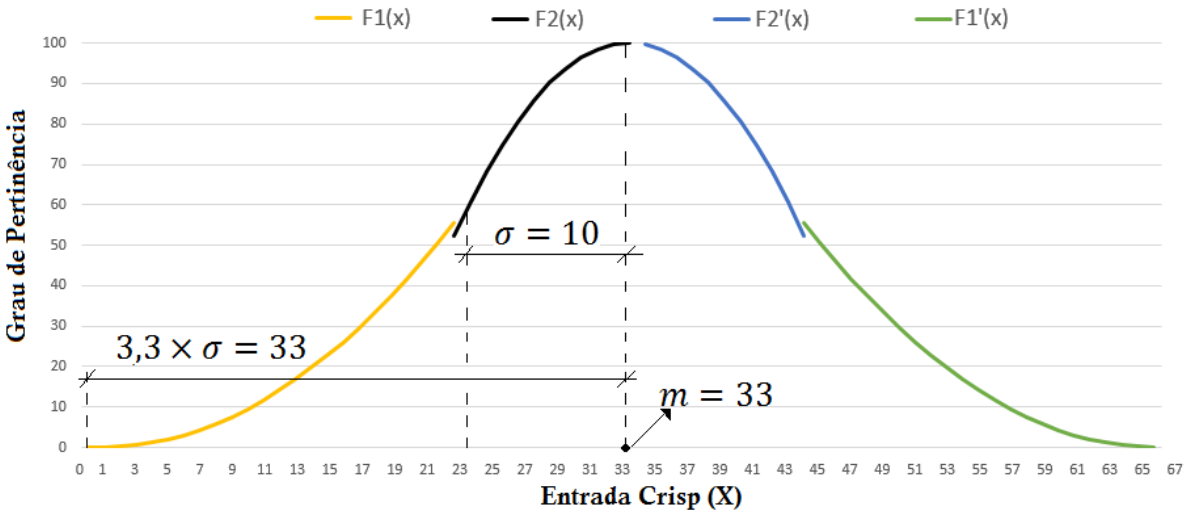


Figura 22: Função gaussiana a partir de polinómios de 2° grau

3.2.1.2. Representação numérica

O circuito apresentado na Figura 21 é implementado em código Verilog utilizando como representação numérica complemento de 2. A entrada *Crisp*, o parâmetro centro da função e a

saída utilizam 8 bits para representar seus dados. Os parâmetros que definem os polinômios a1, a2, b1, b2, c1 e c2, equações (3.4) e (3.5), são definidos em 24 bits. Onde os 16 bits mais significativos representam a parte inteira do polinômio e os 8 bits menos significativos representam a parte fracionária. Todas as operações matemáticas de soma e multiplicação são formatadas para trabalhar com 24 bits em complemento de 2, sendo os 16 bits mais significativos representando a parte inteira e os 8 bits menos significativos representa a parte fracionária. Apenas os multiplexadores MUX1 e MUX2, os quais recebem os dados de entrada, operam com canais de 8 bits, os demais multiplexadores MUX3, MUX4, MUX5 e MUX6, operam seus canais em 24 bits para recebem em suas entradas os valores dos polinômios com 24 bits em complemento de 2. O código em *Verilog* para implementar o circuito da Figura 21 é apresentado no APÊNDICE-B.

3.2.1.3. Resultados do Circuito

O código *Verilog* proposto é testado com a configuração do exemplo de função gaussiana apresentado na seção 3.2.1.1. O gráfico da Figura 23 apresenta o resultado da saída gaussiana *singleton* do circuito (*Figura 21*) com relação a entrada *crisp* variando de 0 a 68. Os dados são aquisitados utilizando a ferramenta *Signal Tap* do Quartus®, a qual possibilita um monitoramento dos sinais durante o funcionamento do *hardware*.

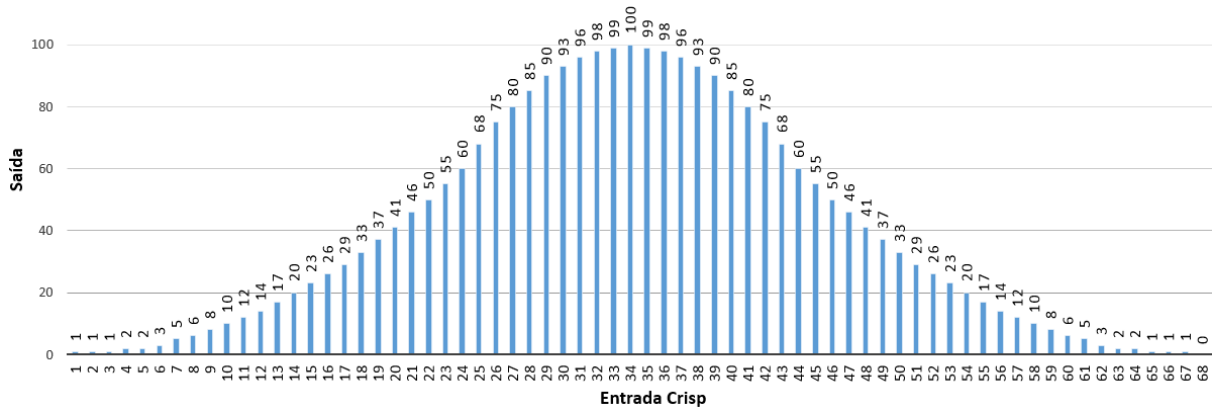


Figura 23: Resultado do circuito implementado

Para implementar a mancha de incerteza FOU do Sistema de Inferência *Fuzzy* (SIF) tipo-2 intervalar é utilizado duas funções de pertinência *fuzzy* tipo-1, uma determina a parte superior do FOU e a outra a parte inferior. O circuito apresentado na *Figura 21* é duplicado para atender as duas funções, a Figura 24 apresenta o diagrama em RTL *Viewer* dos circuitos duplicados. A saída nomeada como “Ativo_UP” é destinada a identificar a ativação do FOU. Essa identificação é feita por um circuito comparador que utiliza a saída da função de pertinência superior “Output_UP” (Figura 24) com relação ao valor zero, qualquer valor maior

que zero para a saída “Output_UP” a saída “Ativo_UP = 1”. Essa informação é utilizada pela base de regras para gerenciar as regras ativas.

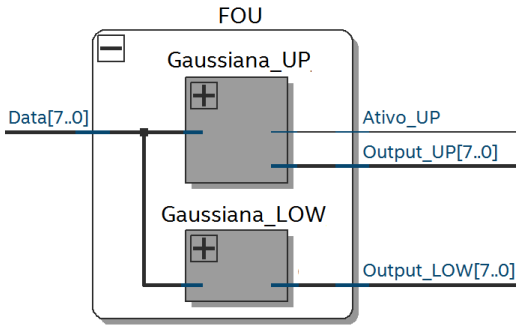


Figura 24: FOU RTL Viewer

O circuito dedicado a função de pertinência superior é configurado com um desvio padrão de $\sigma = 10$, já o circuito para a função de pertinência inferior é configurado com um desvio padrão de $\sigma = 7$. Ambos circuito são configurados com o centro da função em 33. Os dois circuitos operam de maneira paralela e apresentam os gráficos da Figura 25 para as funções superior e inferior do FOU com relação a entrada *crisp* variando de 0 a 68.

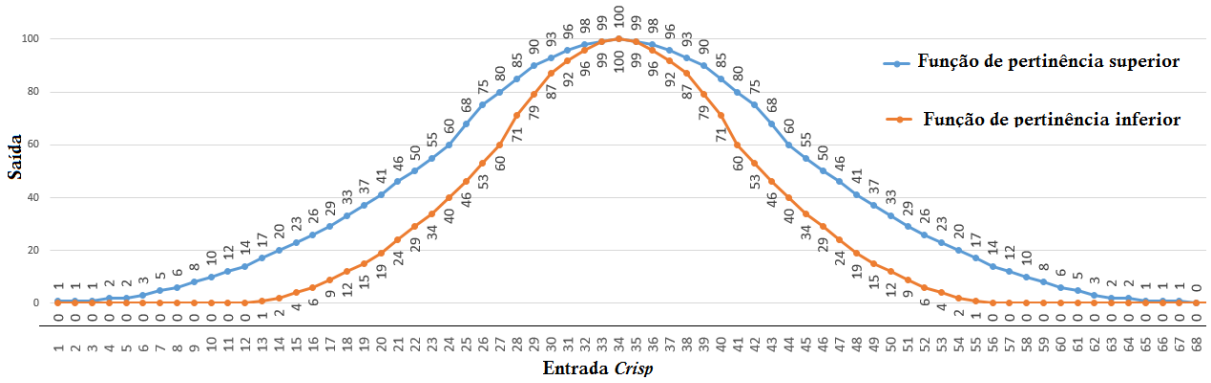


Figura 25: Resultado do circuito dedicado ao FOU

Como a proposta do trabalho é implementar um SIF tipo-2 intervalar de duas entradas com quatro funções de pertinência por entrada, o bloco de fuzificação utiliza quatro FOUs operando em paralelo para cada entrada (Figura 28). Os gráficos das Figuras 26 e 27 apresentam as respostas para as funções de pertinências implementadas nas entradas 01 e 02 da Figura 28. As funções são posicionadas no universo de discurso em 0, 33, 66 e 100 respectivamente para os FOU_1, FOU_2, FOU_3 e FOU_4 na entrada 01 (Figura 26) e em 0, 33, 66 e 100 respectivamente para o FOU_5, FOU_6, FOU_7 e FOU_8 na entrada 02 (Figura 27).

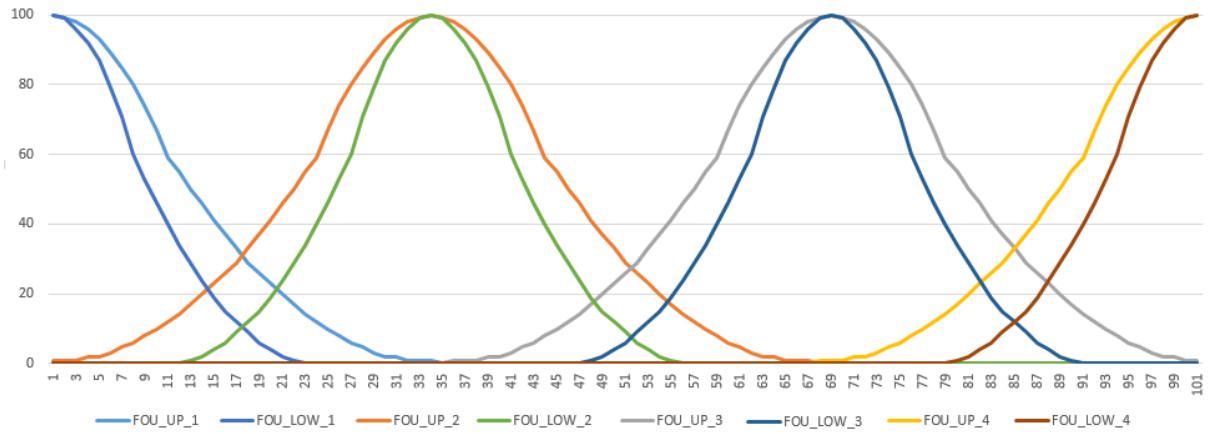


Figura 26: Resposta do circuito com quatro FOUs para entrada 01

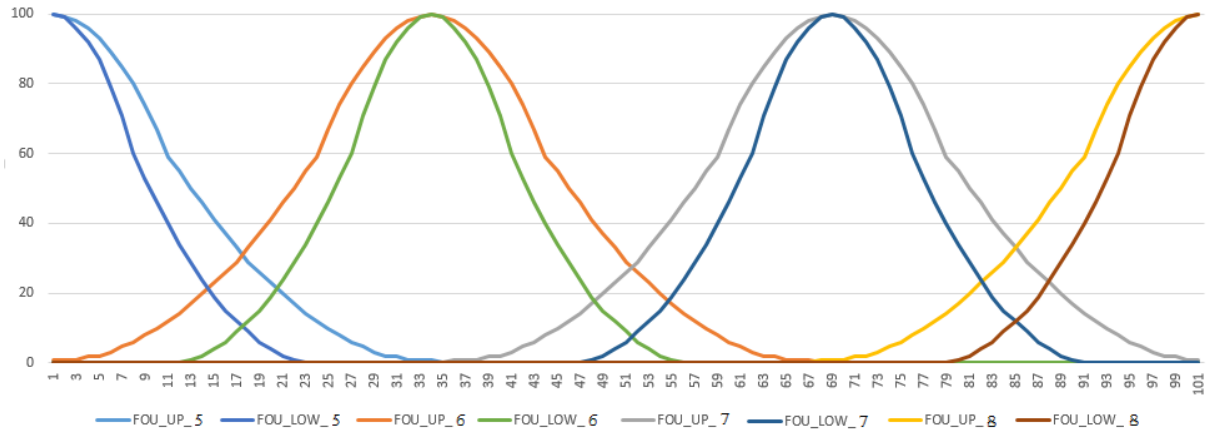


Figura 27: Resposta do circuito com quatro FOUs para entrada 02

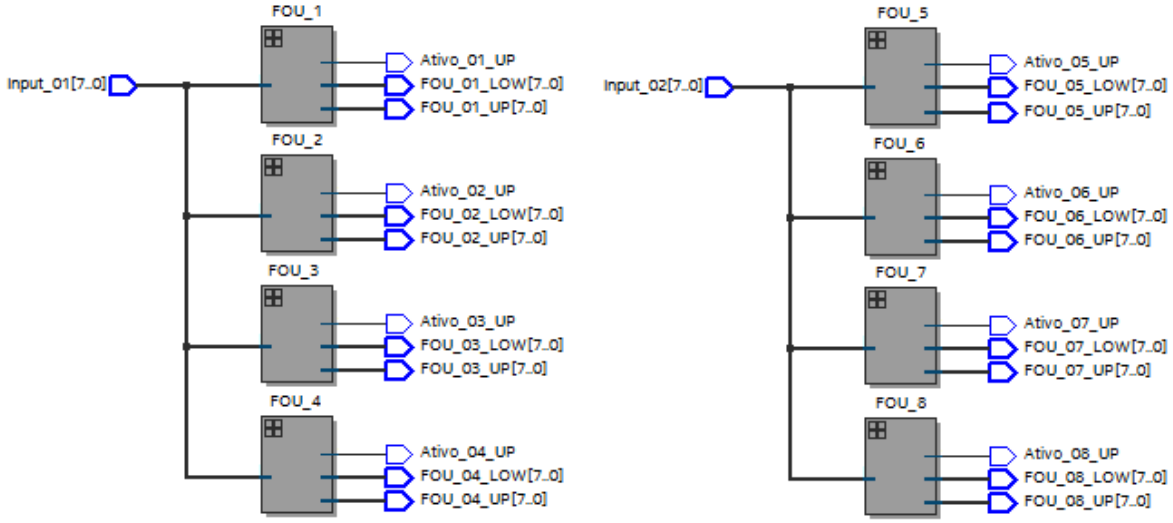


Figura 28: Fuzificador

3.2.2. Circuito de Inferência

A implementação de circuitos em paralelo ou em série tem grande impacto na velocidade de processamento, tamanho do hardware e consumo [20]. Os trabalhos apresentados

em [16], [17], [18], [19], [20] e [36] operam com todos os circuitos em paralelo para implementar um SIF tipo-2 intervalar em FPGA. As arquiteturas com processamento paralelo no mecanismo de inferência utilizam um circuito para cada regra existente, ou seja, quanto maior o número de regras maior é o número de circuitos dedicados no mecanismo de inferência.

Este trabalho opera 16 regras utilizando apenas um circuito de inferência com processamento sequencial para funções de pertinência *fuzzy* tipo-1 (Figura 29) . Para processar funções *fuzzy* tipo-2 intervalar dois circuitos são implementados, uma para as funções de pertinência tipo-1 da parte superior do FOU e o outro para as funções de pertinência tipo-1 da parte inferior do FOU.

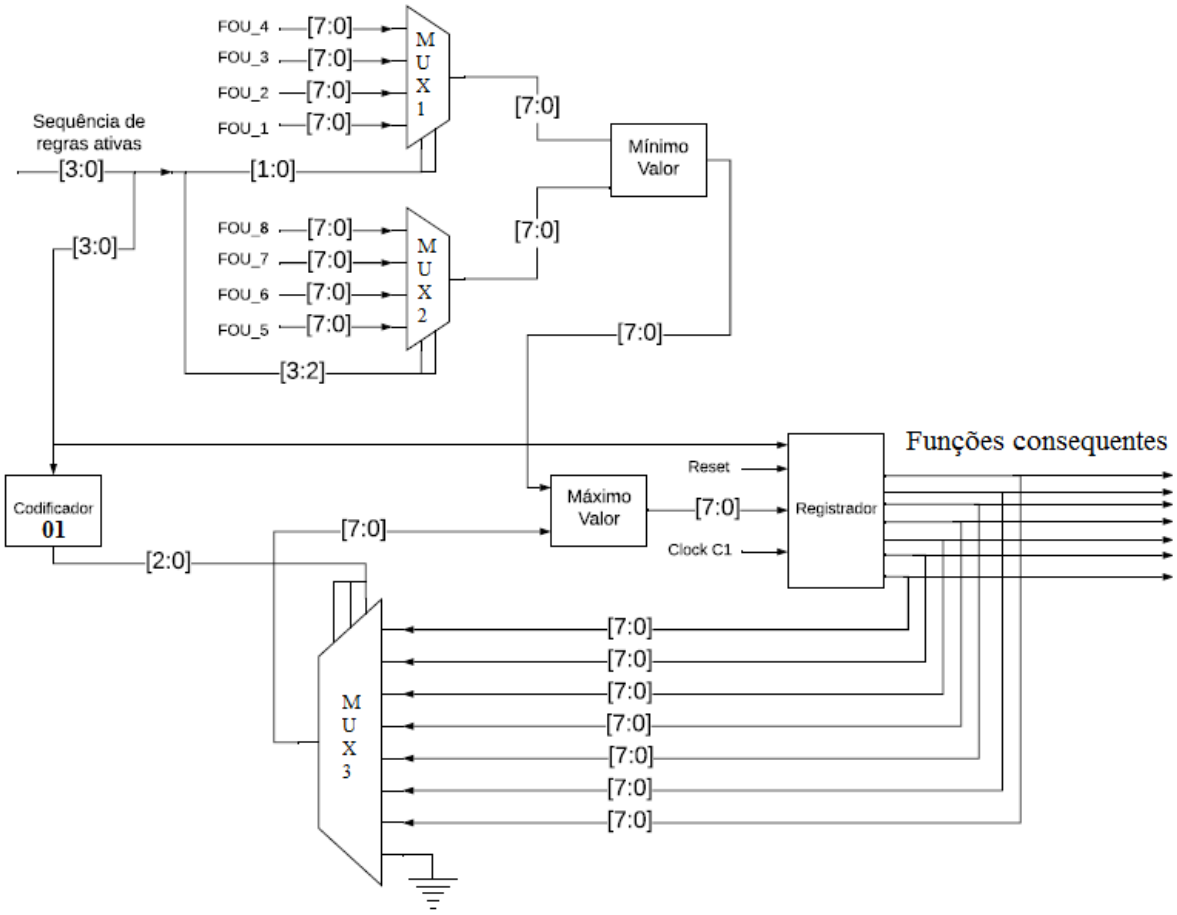


Figura 29: Circuito de inferência mínimo e máximo

O circuito de inferência (Figura 29) utiliza como entrada dois multiplexadores de 4 canais, MUX1 e MUX2. O multiplexado MUX1 recebe em suas entradas as funções de pertinência tipo-1 referente aos FOU_1, FOU_2, FOU_3 e FOU_4, já o multiplexado MUX2 recebe em suas entradas as funções de pertinência tipo-1 referente aos FOU_5, FOU_6, FOU_7 e FOU_8. As funções de pertinência estão detalhadas no fuzificador apresentado na seção (3.2.1.3). Para realizar as operações de inferência as entradas de seleção dos multiplexadores MUX1 e MUX2 são endereçadas com as regras ativas. Já as saídas do circuito de inferência

(Figura 29) utilizam sete registradores de 8 bits para armazenar as funções consequentes resultantes da operação de inferência.

3.2.2.1. Circuito de Mínimo

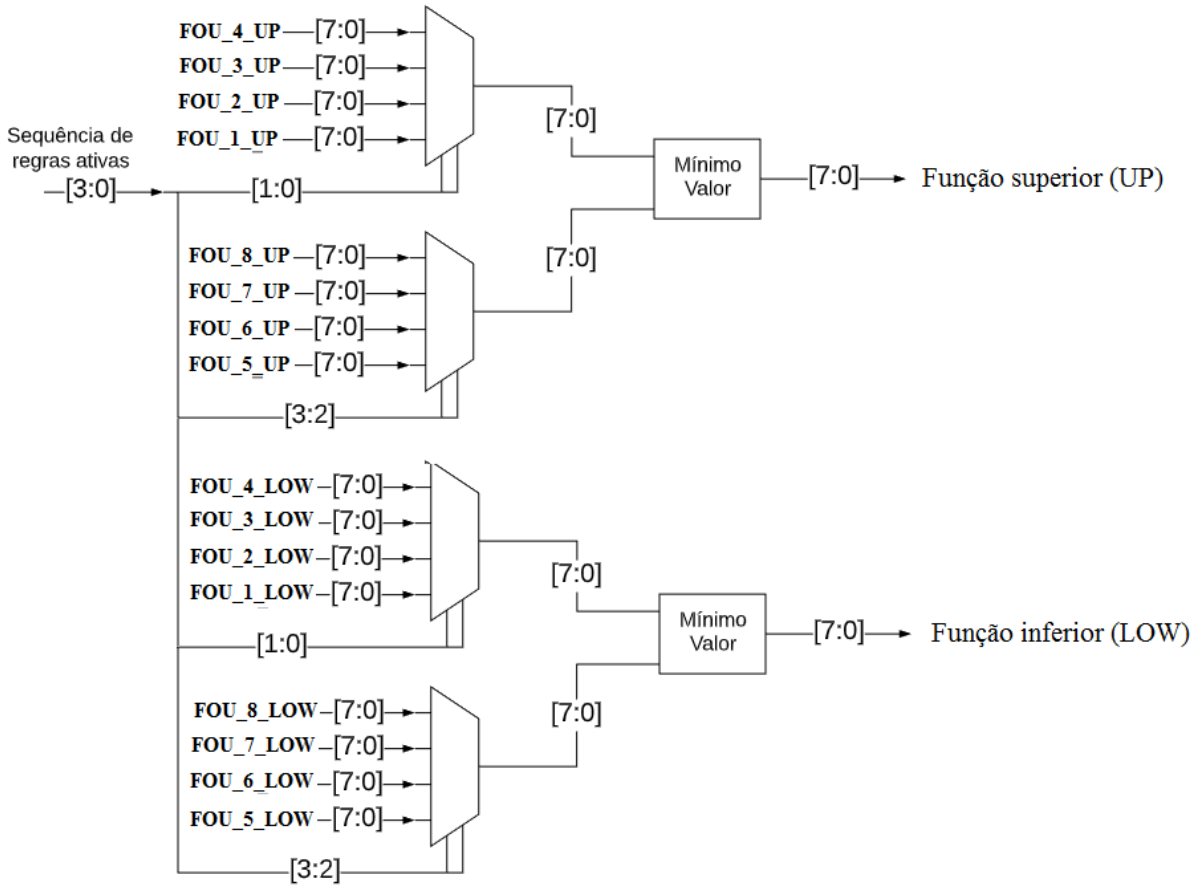


Figura 30: Circuito do bloco mínimo

O circuito de mínimo (Figura 30) define o mínimo grau de pertinência entre duas funções selecionadas por uma regra. O circuito em questão possui dois blocos de mínimo, uma para função gaussiana superior e o outro para a inferior. Os multiplexadores utilizados em cada bloco possuem quatro 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 utiliza na entrada de seleção. Os multiplexadores que recebem as funções de pertinência de 1 a 4 utilizam em sua entrada de seleção os bits de posição 0 e 1 da sequência de regras ativas. Já os multiplexadores que recebem as funções de pertinência de 5 a 8 utilizam os bits de posição 2 e 3 da sequência de regras ativas. A *Figura 31* apresenta a operação de mínimo, executada no circuito da Figura 30, onde a saída $\mu_{\overline{x}}(x)$ determina o mínimo valor para a função superior do FOU e a saída $\mu_{\underline{x}}(x)$ determina o mínimo valor para a função inferior do FOU. O bloco mínimo é o primeiro circuito que recebe e processa as

informações de maneira série, o sincronismo nos dados recebidos da entrada “Sequência de regras ativas” é feito pelo bloco base de regras apresentado na seção (3.2.3).

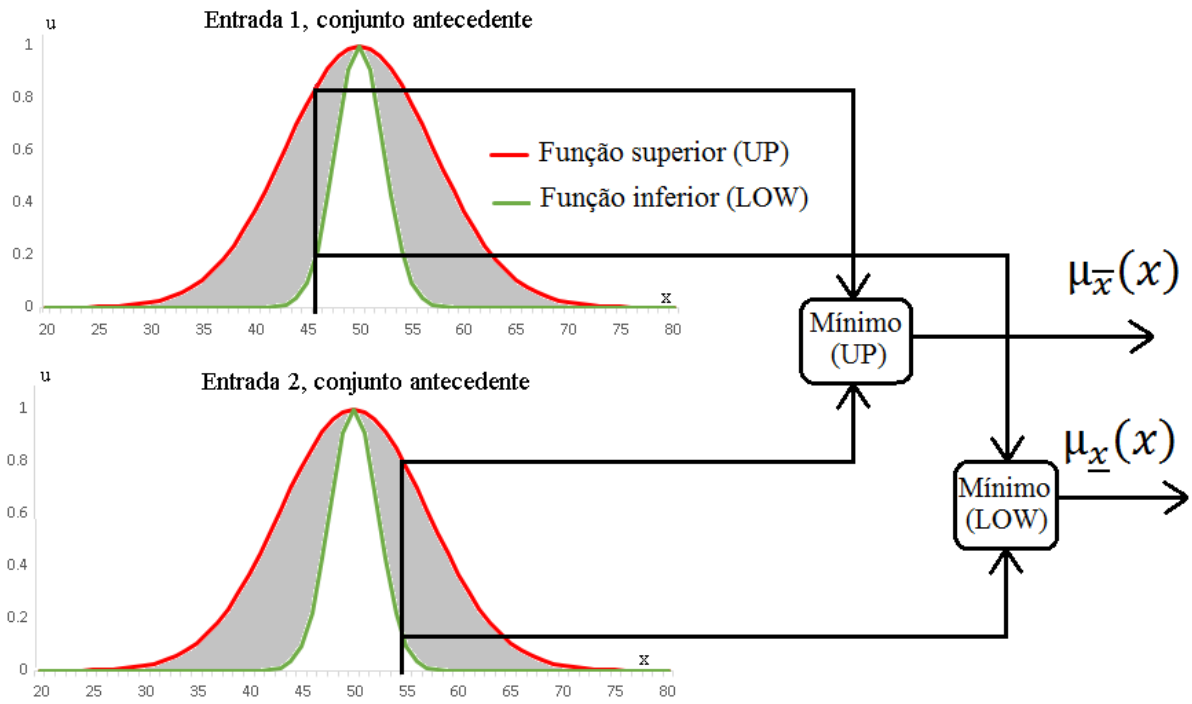


Figura 31: Função do bloco mínimo

3.2.2.2. Circuito de Máximo

A arquitetura proposta utiliza 16 regras que direcionam o grau de ativação em 7 funções consequentes, ou seja, algumas funções consequentes recebem o resultado de duas ou mais regras. Para cada função consequente, o circuito de máximo determina o maior grau de ativação entre as regras relacionadas.

O circuito da *Figura 32* determina o valor de cada função consequente, que é armazenado em registrados de 8 bit. 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 a função consequente é habilitado para atualização. Utilizando o codificado 02 (*Figura 32*), a própria informação de regra ativa habilita o registrado de destino. Por exemplo, caso a regra 6 esteja ativa, o codificador 02 recebe em sua entrada a informação binária 0110 e mantém em nível alto somente o bit de saída referente ao registrador R3, que é o registrado destinado a função consequente da regra 6. A relação de cada regra e sua função consequente é apresentada na Tabela 3.

O operador “Máximo” (*Figura 32*) determina o maior valor entre o grau de pertinência da função consequente, armazenado no registrado de destino, com o grau de pertinência

determinado pelo circuito mínimo para a regra ativa. A *Figura 33* apresenta a relação do mínimo grau de pertinência das funções antecedentes, relacionadas a regra ativa, que é comparada com o valor da função consequente utilizando o operador “Máximo”. Por exemplo, caso 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 se mantém a mesma.

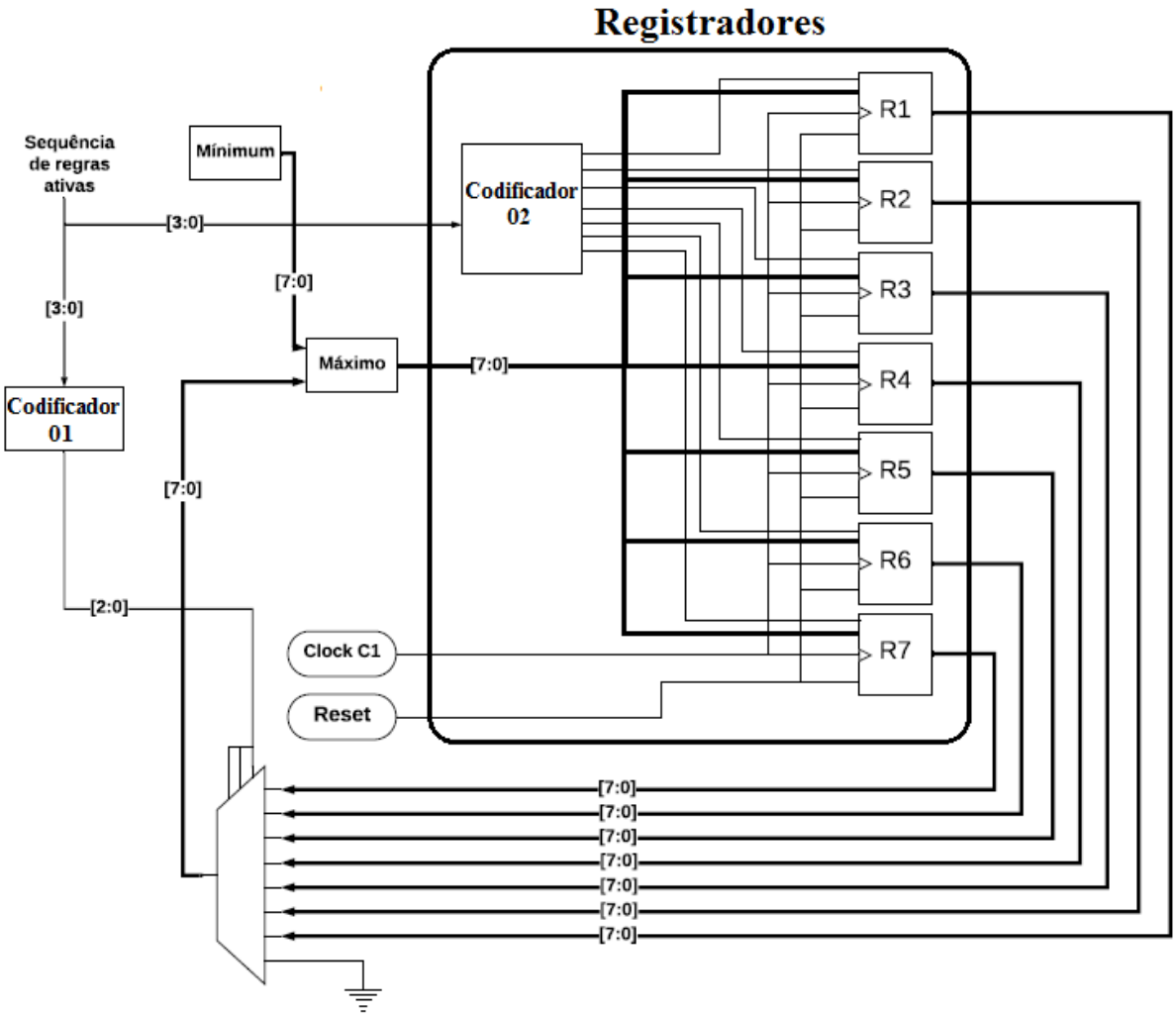


Figura 32: Circuito de máximo valor

A seleção do registrador a ser utilizado na entrada do circuito “Máximo” é feita utilizando o multiplexador de 8 canais apresentado na *Figura 32*. Onde a entrada de seleção é determinada pelo codificador 01, o qual utiliza a informação de regra ativa para determinar qual canal de entrada deve ser atualizado na saída do multiplexador. Por exemplo, caso a regra 6 esteja ativa, o codificador 01 recebe em sua entrada a informação binária 0110 e ativa o canal

3 do multiplexador, esse canal possui como entrada o registrador R3 referente a função consequente da regra 6.

Os registradores utilizam o *clock* c1 com um atraso de fase para que as informações sejam totalmente processadas de antes de atualizar o registrador de destino. A utilização, sincronismo das entradas de *clock* e o tempo de atraso de cada circuito será temas abordados na seção (3.3).

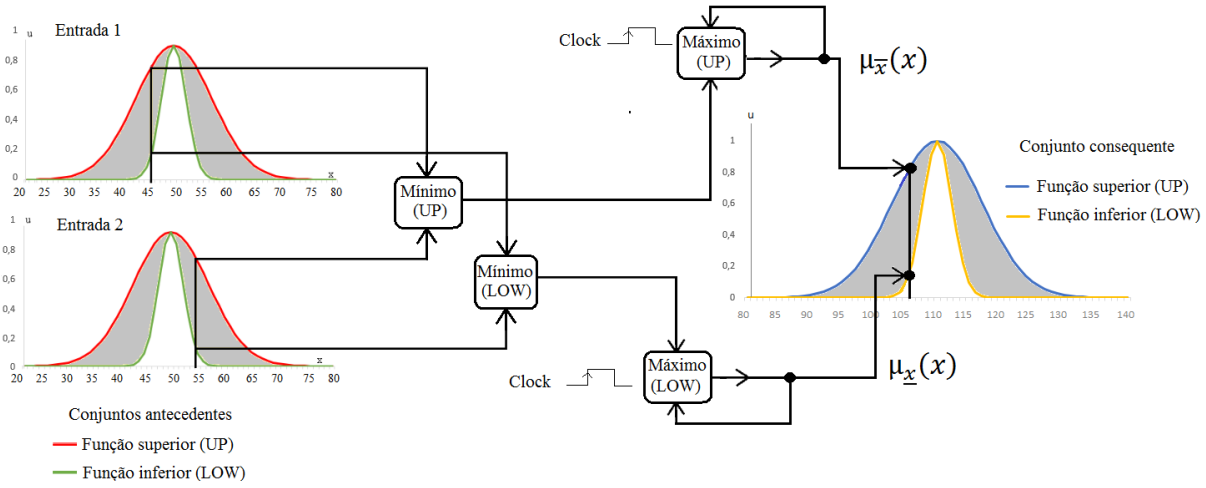


Figura 33: Operação de inferência.

3.2.2.3. Resultado do Circuito de Inferência

Um exemplo de operação para as funções de pertinência superiores $\mu_{\overline{x}}(x)$ do circuito de inferência é apresentado na Figura 34. Onde a posição de cada função de pertinência, a largura do FOU e o universo de discurso são ajustados para o controle de frenagem de uma locomotiva. A entrada *Crisp Input* 1 representa a velocidade da locomotiva com um universo de discurso que varia de 0 a 100Km/h, a entrada *Crisp Input* 2 representa a distância de parada da locomotiva com um universo de discurso de 0 a 100 metros e a saída *Crisp Output* representa a porcentagem de atuação dos freios da locomotiva com um universo de discurso que varia de 0 a 100%.

Caso a entrada *Crisp Input* 1 tenha seu valor fixado em 45Km/h, as funções de pertinência FOU_2, velocidade *Mean*, e o FOU_3, velocidade *High*, são ativas (Figura 34). Caso a entrada *Crisp Input* 2 tenha seu valor fixado em 13 metros, as funções de pertinência FOU_5, distância *Very short*, e o FOU_6, distância *Short*, são ativas (Figura 34).

Como o valor de 45 Km/h do *Crisp Input* 1 ativou duas funções e o valor de 13 metros do *Crisp Input* 2 também ativou duas funções, 4 regras são ativadas. Para as funções ativas as regras 4, 5, 8 e 9 serão selecionadas pela base de regras (Tabela 3).

Tabela 3: Base de Regras

		Input 2: Distance			
		Very Short	Short	Far	Very Far
Input 1: Speed	Low	Register 4 High Rule 0	Register 3 Mean Rule 1	Register 2 Low Rule 2	Register 1 Very Low Rule 3
	Mean	Register 5 Very High Rule 4	Register 4 High Rule 5	Register 3 Mean Rule 6	Register 2 Low Rule 7
	High	Register 6 Emergency Rule 8	Register 5 Very High Rule 9	Register 4 High Rule 10	Register 3 Mean Rule 11
	Very High	Register 7 Maximum Rule 12	Register 6 Emergency Rule 13	Register 5 Very High Rule 14	Register 4 High Rule 15

Na Figura 34, a saída de cada multiplexador do circuito mínimo é selecionada de acordo com o valor das regras que a Base de Regras gerencia. Por exemplo: o código binário definido para a regra 4 é 4'b0100, os bits das posições 3 e 2 (01) selecionam o FOU_2, que possui um grau de ativação de $\mu_{\overline{x}}(45) = 55$, já os bits das posições 1 e 0 (00) selecionam o FOU_5, que possui um grau de ativação de $\mu_{\overline{x}}(13) = 50$. Um circuito identificador de mínimo, posicionado para receber os dados de saída dos multiplexadores, estabelece o menor valor entre os FOU_5 e o FOU_2, que no exemplo em questão é 50. A operação de máximo é executada comparando o valor da função consequente 5 *Very High*, referente a regra 4, como o valor mínimo que acabou de ser definido. Como essa é a primeira regra executada, o valor armazenado no registrador R5 está em zero, portanto o valor máximo a ser atualizado na função Conseq_5 *Very High* é de $\mu_{\overline{x}}(13) = 50$.

O valor mínimo para regra 9 é estabelecido entre o FOU_6 distância *Short* igual a $\mu_{\overline{x}}(13) = 17$ e o FOU_2 velocidade High igual a $\mu_{\overline{x}}(45) = 10$, ou seja, o mínimo valor estabelecido para a regra 9 é de $\mu_{\overline{x}}(45) = 10$. Quando a regra 9 é executada, o registrador R5 tem o valor 50 registrado, referente a regra 4. Portanto, o registrador R5 é atualizado com o máximo grau de ativação estabelecido entre a regra 9 e a regra 4, que no exemplo em questão é o valor 50 apresentado na regra 4.

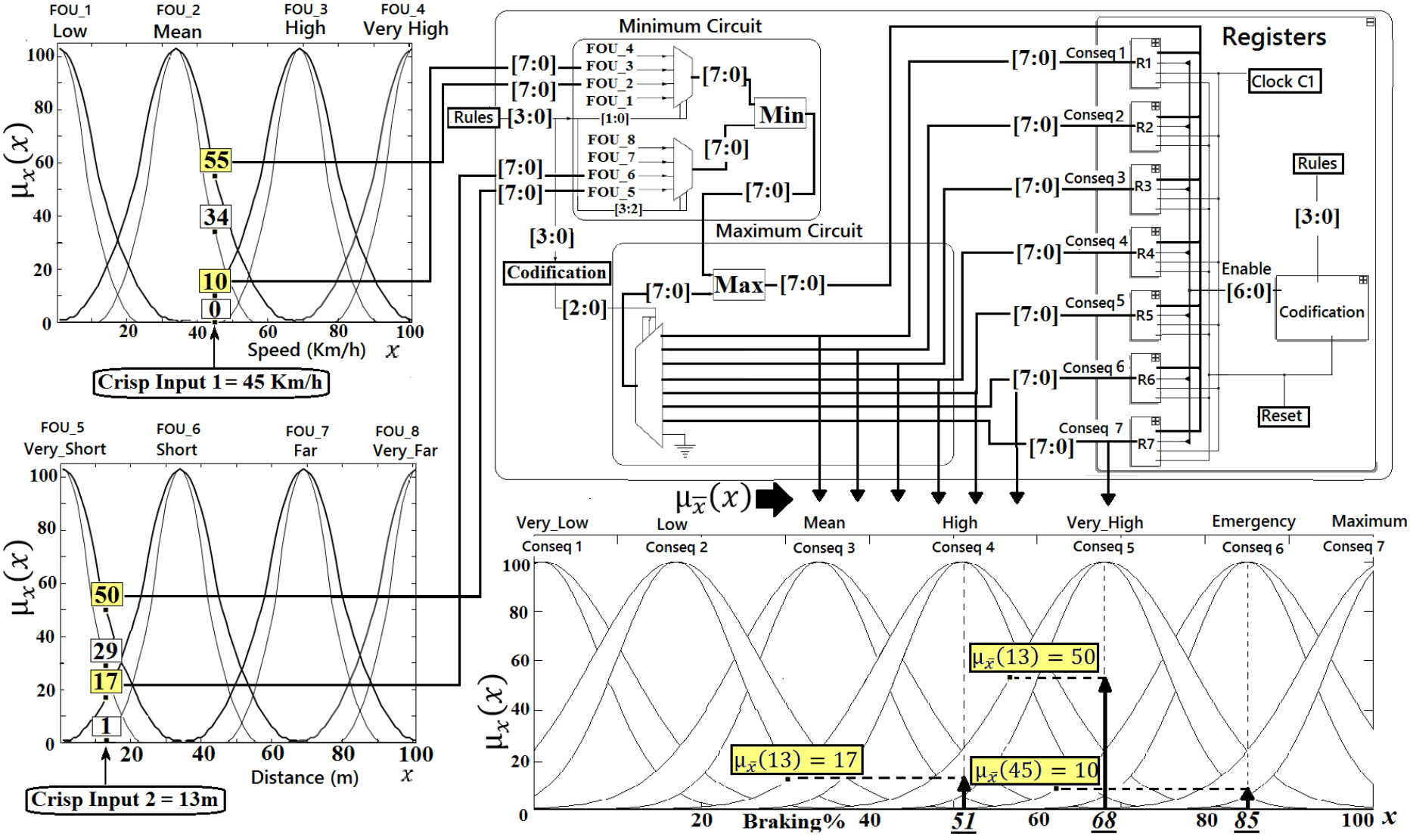


Figura 34: Circuito de inferência

3.2.3. Base de regras

A definição das regras pode ser feita por um especialista ou extraída de dados reais. Cada regra é uma declaração *IF-THEN*. A parte *IF* de uma regra é chamada antecedente e a parte *THEN* é chamada de consequente. A Figura 35 apresenta as regras antecedentes e consequentes utilizadas no modelo implementado neste trabalho.

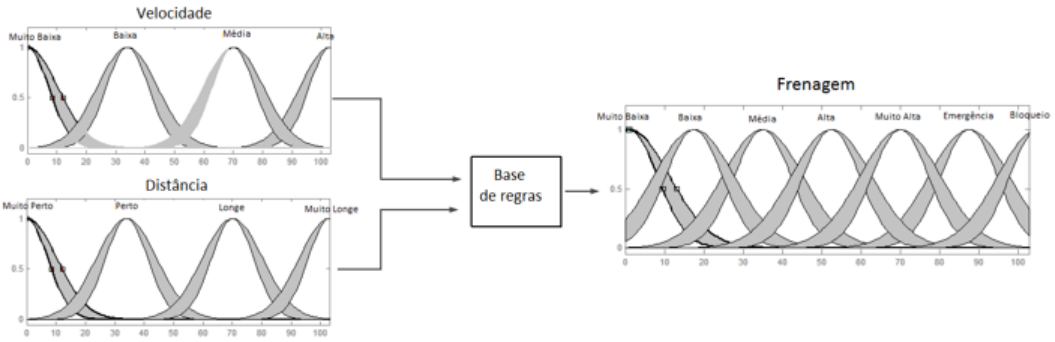


Figura 35: Funções antecedentes e consequentes

A base de regras proposta utiliza o exemplo abordado na seção (3.2.2.3) para um SIF tipo-2 intervalar que calcula a frenagem de uma locomotiva. O exemplo em questão possui quatro funções de pertinência para entrada velocidade e quatro funções de pertinência para a entrada distância. Para atender a todas as relações das funções de pertinência a base de regras implementa 16 regras, a Tabela 4 apresenta as definições de cada regra.

Tabela 4: Base de regras do SIF-T2

Regra 00:	SE (velocidade é muito baixa)	E (distância muito perto)	ENTÃO (Frenagem é alta)
Regra 01:	SE (velocidade é muito baixa)	E (distância perto)	ENTÃO (Frenagem é média)
Regra 02:	SE (velocidade é muito baixa)	E (distância longe)	ENTÃO (Frenagem é baixa)
Regra 03:	SE (velocidade é muito baixa)	E (distância muito longe)	ENTÃO (Frenagem é muito baixa)
Regra 04:	SE (velocidade é baixa)	E (distância muito perto)	ENTÃO (Frenagem é muito alta)
Regra 05:	SE (velocidade é baixa)	E (distância perto)	ENTÃO (Frenagem é alta)
Regra 06:	SE (velocidade é baixa)	E (distância longe)	ENTÃO (Frenagem é média)
Regra 07:	SE (velocidade é baixa)	E (distância muito longe)	ENTÃO (Frenagem é baixa)
Regra 08:	SE (velocidade é média)	E (distância muito perto)	ENTÃO (Frenagem é de emergência)
Regra 09:	SE (velocidade é média)	E (distância perto)	ENTÃO (Frenagem é muito alta)
Regra 10:	SE (velocidade é média)	E (distância longe)	ENTÃO (Frenagem é alta)
Regra 11:	SE (velocidade é média)	E (distância muito longe)	ENTÃO (Frenagem é média)
Regra 12:	SE (velocidade é alta)	E (distância muito perto)	ENTÃO (Frenagem é de bloqueio)
Regra 13:	SE (velocidade é alta)	E (distância perto)	ENTÃO (Frenagem é de emergência)
Regra 14:	SE (velocidade é alta)	E (distância longe)	ENTÃO (Frenagem é muito alta)
Regra 15:	SE (velocidade é alta)	E (distância muito longe)	ENTÃO (Frenagem é alta)

Cada entrada *Crisp* pode ativar apenas uma ou duas funções de pertinência. Portanto, apesar de existirem 16 regras nem todas são ativas ao mesmo tempo, por exemplo: 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 entrada ative apenas uma função de pertinência duas regras são selecionadas e caso ambas entradas ativem duas funções de pertinência quatro regras são selecionadas.

A base de regras implementada neste trabalho utiliza uma máquina de estado (Figuras 36 e *37*) 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 funções de pertinência, onde "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 36) gerencia 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 16 regras que estão relacionadas com suas respectivas funções antecedentes e consequentes nas Tabelas 3 e *4*. Para identificar e gerenciar as regras ativas a máquina de estado é projetada com 18 estados (*Figura 37*). Um estado limpa as memórias, dezesseis estados são dedicados a gerar o código das 16 regras e um estado habilita as memórias para atualização de valores.

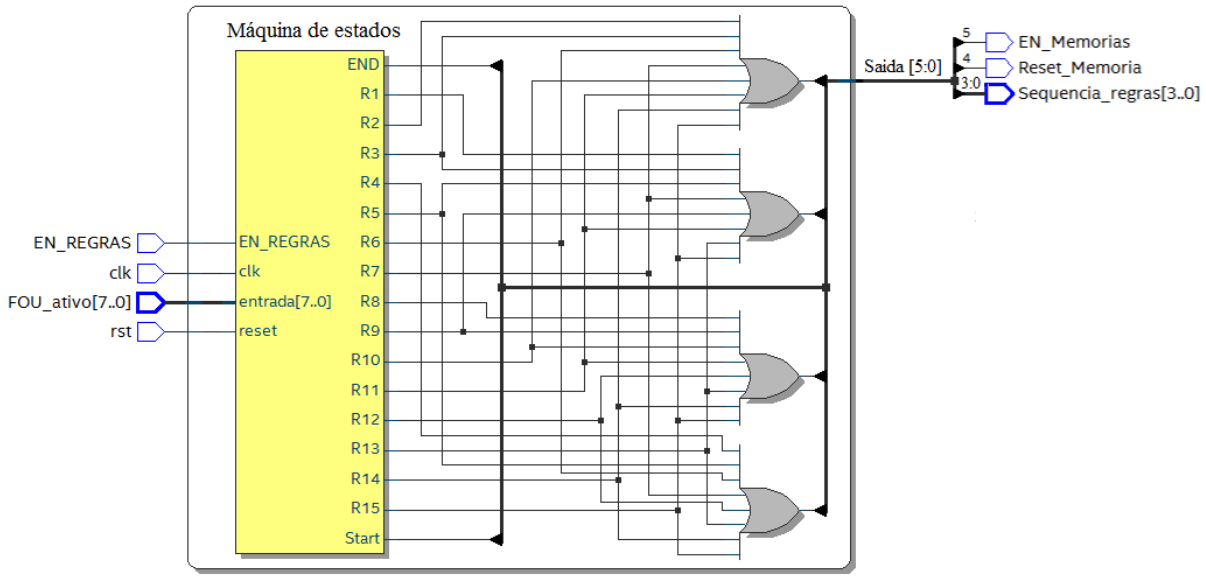


Figura 36:Circuito da base de regras

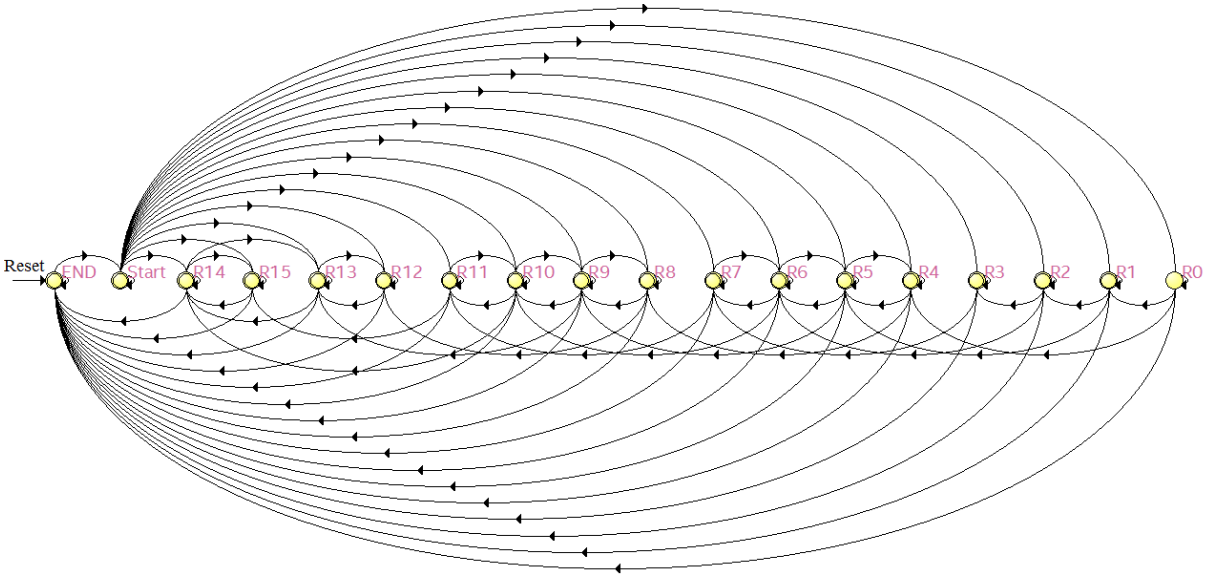


Figura 37: Máquina de estado para base de regras

A Figura 38 apresenta um fluxograma que define a sequência lógica dos estados. 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, R8, R9, R10, R11, R12, R13, R14 e R15) dedicados as 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.
- 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).

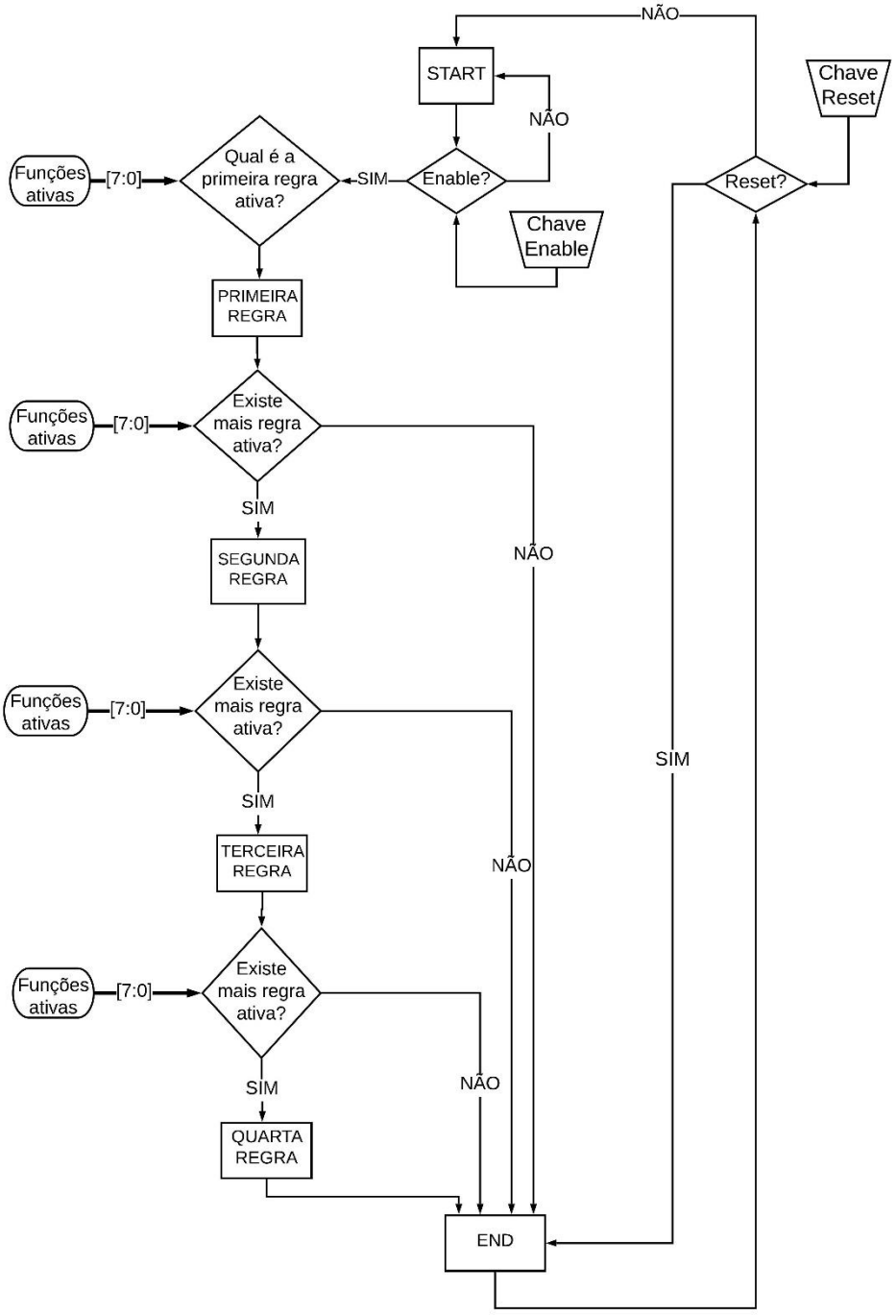


Figura 38: Fluxograma da máquina de estado

3.2.4. Tipo redutor e defuzificador

Neste trabalho é adotado o algoritmo Nie-Tan (NT) [14] para calcular a saída *crisp* utilizando as funções consequentes *singleton* de saída do bloco de Inferência, seção (3.2.2).

O algoritmo de Nie-Tan define que a operação de redução de tipo é feita com a média entre os graus de ativação superior e inferior das funções de pertinências (*Figura 39*). Onde $\overline{f}$ representa a função de pertinência superior e $\underline{f}$ representa a função de pertinência inferior.

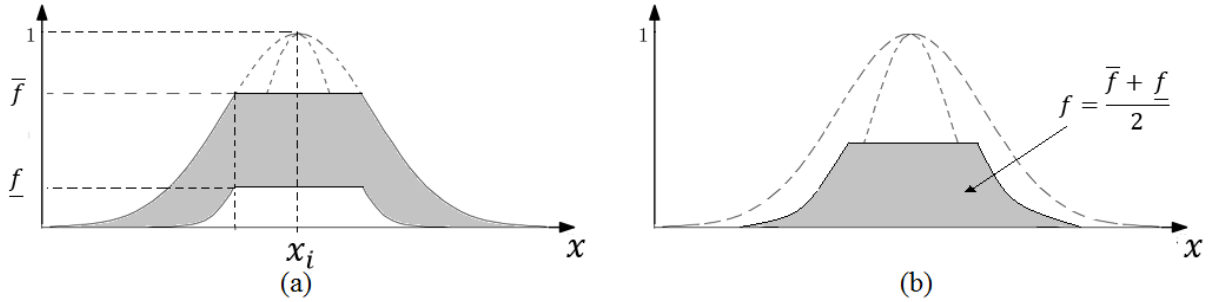


Figura 39: (a) Função de pertinência tipo-2 (b) Função de pertinência tipo-1 resultante do processo de redução de tipo NT.

Desta forma a redução de tipo NT para n-ésima função de pertinência é definida na equação (3.6).

$$f_n = \frac{\bar{f}^n + \underline{f_n}}{2} \tag{3.6}$$

As funções de pertinências tipo-1, obtidas pela operação de redutor de tipo apresentada na equação (3.6), são utilizadas para determinar o valor de saída *crisp* utilizando o método de defuzificação da Altura **(Apêndice -C).** Desta forma a equação (3.7) define a operação para determinação do valor *crisp* de saída pelo algoritmo de Nie-Tan.

$$Y_{NT} = \frac{\sum_{i=1}^{i=N} x_i f_n}{\sum_{i=1}^{i=N} f_n} \tag{3.7}$$

Onde 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. Substituindo a equação (3.6) na (3.7) obtém a equação (3.8).

$$Y_{NT} = \frac{\sum_{i=1}^{M} y^i(\bar{f}^i + \underline{f}^i)}{\sum_{i=1}^{M} (\bar{f}^i + \underline{f}^i)} \tag{3.8}$$

A equação (3.8) 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 circuito dedicado a redutor tipo, isso reduz o número de blocos do processamento de saída.

O processamento de saída proposto neste trabalho opera com sete funções de pertinência. A equação (3.9) apresenta a expansão da equação (3.8) onde a operação $(y.K_n)$ determina a posição do centro de cada função de pertinência "n" no universo de discurso.

$$Y_{NT} = y\left[\frac{K_1(\bar{f}^1 + \underline{f}^1) + K_2(\bar{f}^2 + \underline{f}^2) + K_3(\bar{f}^3 + \underline{f}^3) + K_4(\bar{f}^4 + \underline{f}^4) + K_5(\bar{f}^5 + \underline{f}^5) + K_6(\bar{f}^6 + \underline{f}^6) + K_7(\bar{f}^7 + \underline{f}^7)}{(\bar{f}^1 + \underline{f}^1) + (\bar{f}^2 + \underline{f}^2) + (\bar{f}^3 + \underline{f}^3) + (\bar{f}^4 + \underline{f}^4) + (\bar{f}^5 + \underline{f}^5) + (\bar{f}^6 + \underline{f}^6) + (\bar{f}^7 + \underline{f}^7)}\right] \tag{3.9}$$

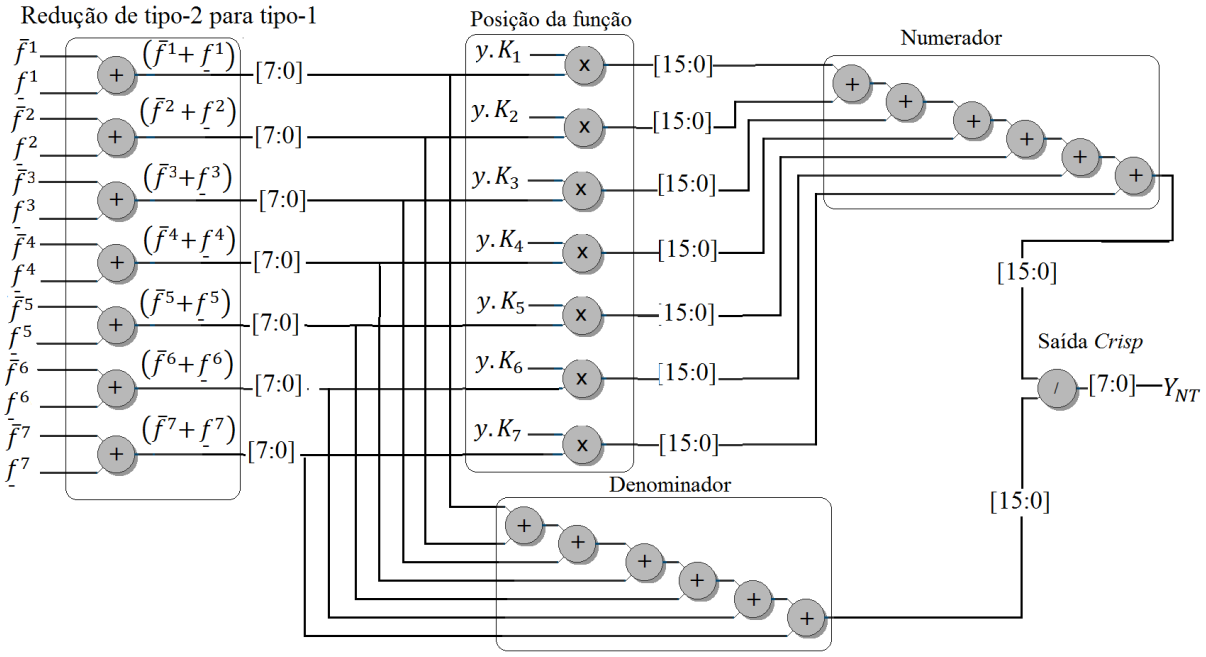


Figura 40: Circuito para implementação do método de Nie-Tan

Para o exemplo apresentado na Figura 34 e discutido na seção (3.2.2.3), as funções consequentes da parte superior do FOU apresentam os seguintes valores: Conseq_4 = 17, Conseq_5 = 50 e Conseq_6 = 10. Já os valores definidos para a parte inferior do FOU são: Conseq_4 = 1, Conseq_5 = 29 e Conseq_6 = 0. Substituindo esses valores na Equação (3.9), os cálculos são apresentados nas Equações (3.10) e (3.11).

$$Y_{NT} = \left[\frac{1(0+0)+17(0+0)+34(0+0)+51(17+1)+68(50+29)+85(10+0)+102(0+0)}{(0+0)+(0+0)+(0+0)+(17+1)+(50+29)+(10+0)+(0+0)}\right] \quad (3.10)$$

$$Y_{NT} = 17\left[\frac{51(18)+68(79)+85(10)}{18+79+10}\right] = 66{,}7\% \quad (3.11)$$

Portanto, para o exemplo apresentado na seção (3.2.2.3), onde 45Km/h é o valor estabelecido para a entrada *Crisp Input 1* e 13 metros para a entrada *Crisp Input 2*, a saída *Crisp* vai apresentar 66% de atuação dos freios, como pode ser observado na Equação (3.11). A saída *Crisp* utiliza apenas a parte inteira do valor calculado na Equação (3.11) representado em 8 bits.

3.3. *Hardware* completo para SIF tipo-2 intervalar

A Figura 41 apresenta a arquitetura implementada em FPGA para um Sistema de Inferência Fuzzy (SIF) tipo-2 intervalar. Os blocos Fuzificador UP e Fuzificador LOW implementam as funções gaussianas *singleton* tipo-1 que determinam a parte superior e inferior da mancha de incerteza FOU, seção (3.2.1). Para cada bloco de fuzificação é utilizado um circuito de Inferência (Figura 29) que relaciona as funções antecedentes com as funções consequentes, seção (3.2.2). As saídas dos circuitos de Inferência são conectadas no circuito aritmético Tipo-

Redutor (*Figura 40*) para determinar a saída *Crisp* utilizando o algoritmo de Nie-Tan, seção (3.2.4). A base de regras identifica e gerencia as regras ativas utilizando como entrada o vetor de 8 bits “FOU_Ativo” da saída do Fuzificador UP, o qual indica quais funções de entrada estão ativas, seção (3.2.3).

O processamento dos dados da arquitetura proposta na Figura 41 é apresentado no fluxograma da Figura 42. A atualização dos dados das duas entradas e do valor calculado na saída *crisp* é feita na entrada *clock* dos Flip Flops utilizando o *bit Enable* da saída da Base de Regras. Os blocos de Fuzificação e Tipo-Redutor utilizam processamento paralelo e não necessitam de atualização por *clock*. Já o Bloco de Inferência processa as regras de modo sequencial utilizando o *clock* C1 que é sincronizado com o *clock* C0 da Base de Regras. Os dois *clocks* são obtidos com a *Phase Locked Loop* (PLL) apresentada na Figura 43, onde a entrada “*inclok0”* utiliza o *clock* de 50 Mhz, disponível na FPGA, para obter os clocks C0 e C1 com frequência de 30Mhz. O *clock* C1 é configurado na PLL com 135°, ou 12,5 ns, de atraso em relação ao *clock* C0.

Os tempos de atraso, apresentados em vermelho na Figura 42, são determinados de maneira individual com o analisador lógico *Signal Tap.*

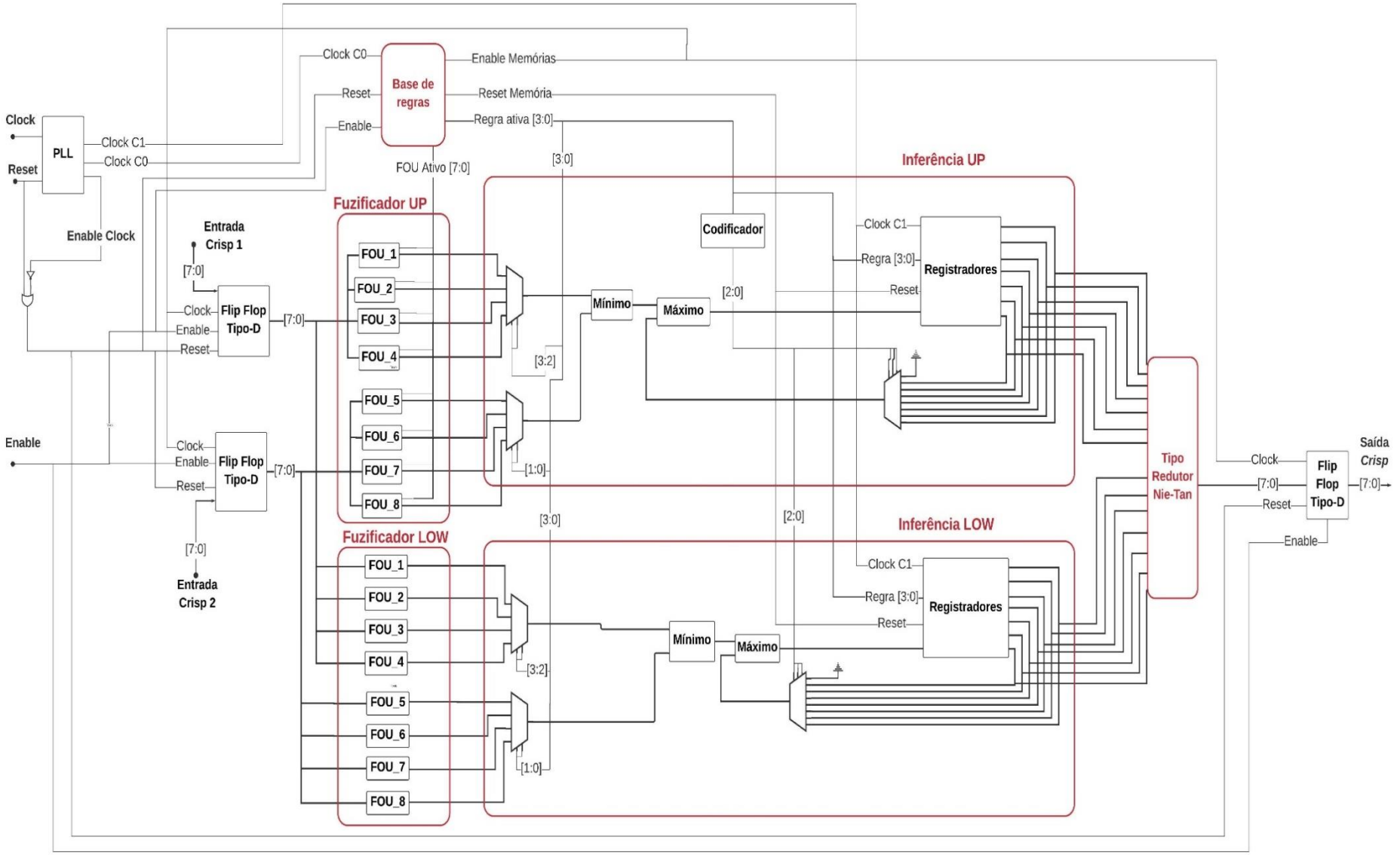


Figura 41:Arquitetura do hardware em FPGA

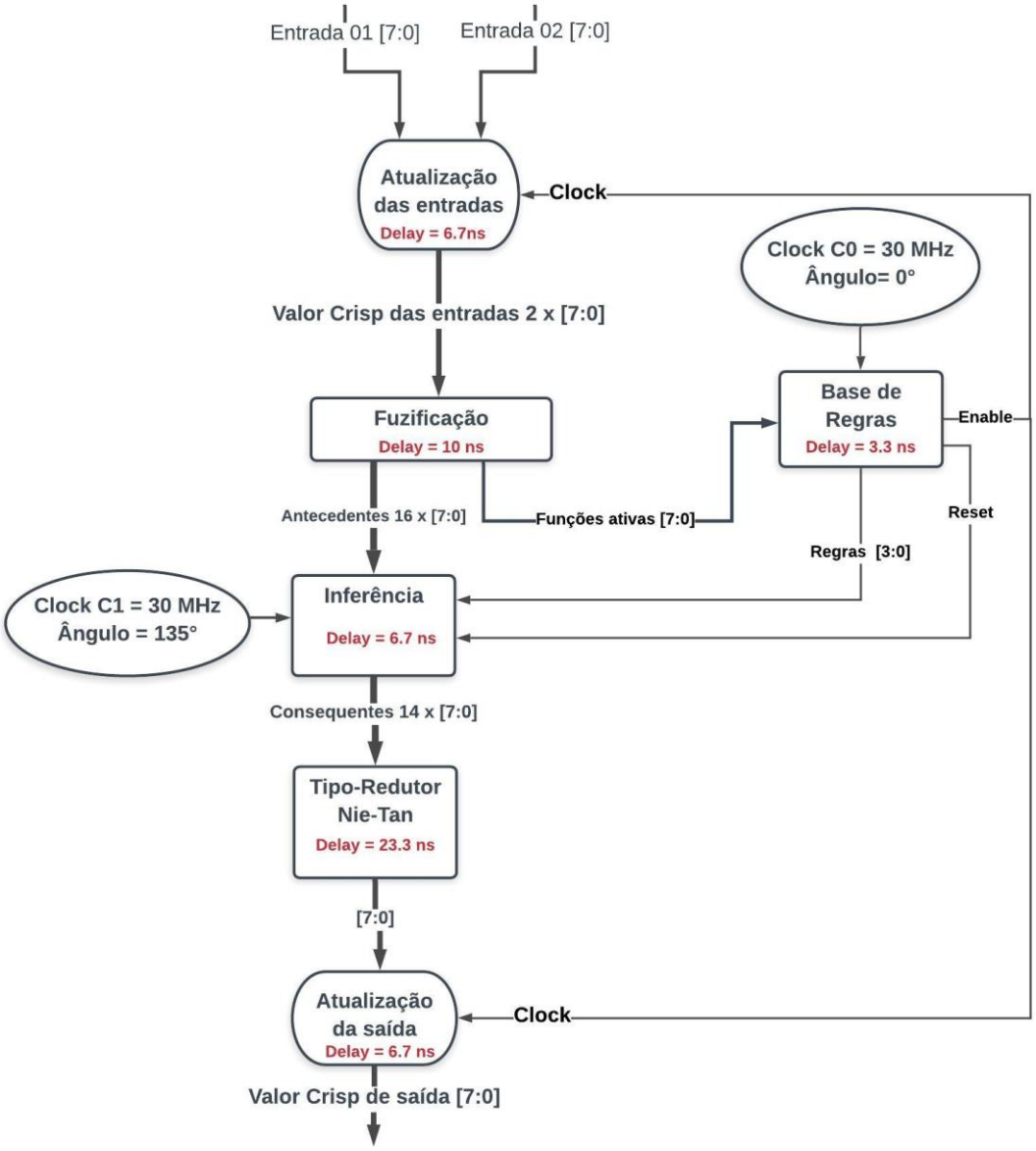


Figura 42: Fluxograma dos dados do hardware

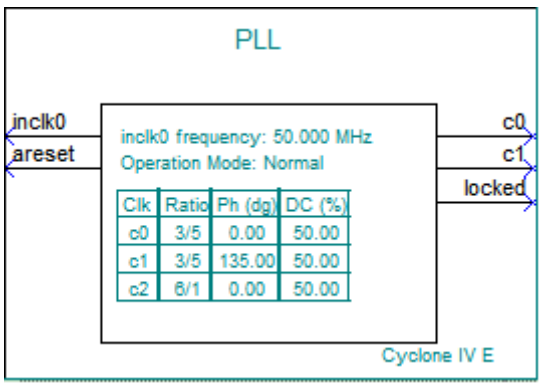


Figura 43: PLL

4. Resultados Encontrados

4.1. Introdução

Essa seção apresenta os resultados encontrados durante os testes e simulações feitas no *hardware* de Sistema de Inferência Fuzzy (SIF) tipo-2 intervalar, implementado na plataforma FPGA DE2 115. As configurações utilizadas durante os teste para o SIF tipo-2 intervalar são as mesmas utilizadas no exemplo de controle de frenagem de locomotiva apresentado na seção (3.2.1.3).

4.2. Resultado das simulações

4.2.1. Ferramenta

O analisador lógico *Signal Tap*, disponível no *software* Intel Quartus® Prime, é utilizado neste trabalho para sondar e depurar o comportamento dos sinais internos em tempo real durante a operação do dispositivo. Esse recurso tem a vantagem de não necessitar de pinos extras de saída ou equipamentos externos de laboratório para o monitoramento dos sinais. A *Figura 44* apresenta o funcionamento da ferramenta que utiliza os seguintes blocos:

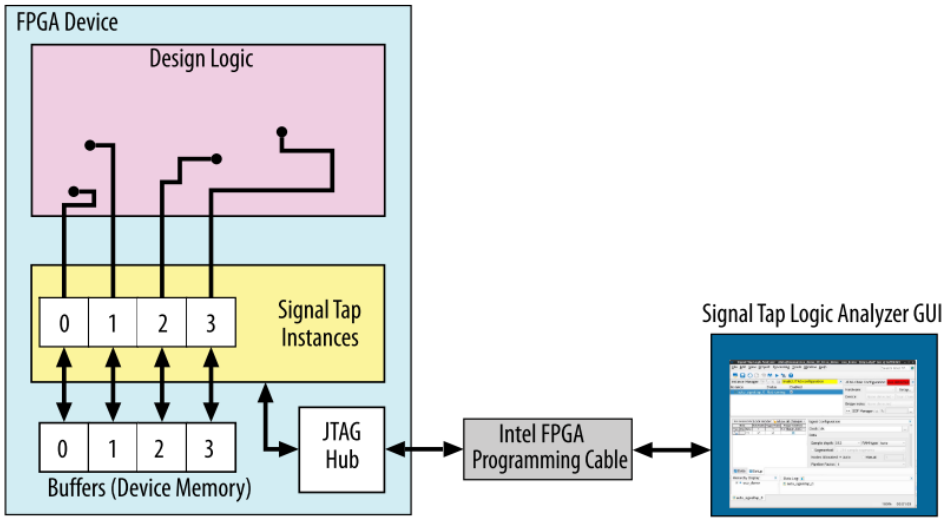


Figura 44: Diagrama de blocos do *Signal Tap*. Fonte: [37].

Design Logic: é o circuito desenvolvido em FPGA que gera os sinais de origem.

Signal Tap Instances: habilita as funcionalidades do analisador lógico definindo as instâncias que serão utilizados. As propriedades das instâncias são definidas no *Signal Tap Logic Analyzer* GUI.

Signal Tap Logic Analyzer GUI: define as configurações das instâncias, como por exemplo: quantidade de amostra, tipo de memória, sinal de *trigger* e frequência de amostragem (Figura 45). Os dados das configurações são salvos e gerenciados em arquivo (.stp).

Buffers (Device Memory): é a memória configurada para salvar os dados capturados do dispositivo.

JTAG Hub: após configurar das instâncias, compilar o projeto e programar a FPGA, a comunicação para captura dos dados é feita por meio da conexão JTAG.

Os dados capturados na memória do dispositivo podem ser analisados e filtrados utilizando a interface gráfica *Signal Configuration.* A *Figura 52* apresenta uma análise de todos os sinais aquisitados no *hardware* de SIF tipo-2 intervalar utilizando a interface gráfica *Signal Configuration*. Para exportar os dados, a opção *Create Signal Tap Liste File* cria um arquivo (.txt) na pasta raiz do projeto, o qual pode ser trabalhado em outras plataformas gráficas.

Para um estudo mais aprofundado da ferramenta *Signal Tap*, é aconselhável uma consulta no capítulo 2 *Design Debugging with the Signal Tap Logic Analyzer* do guia de utilização *Intel® Quartus® Prime Pro Edition User Guide - Platform Designer* [37].

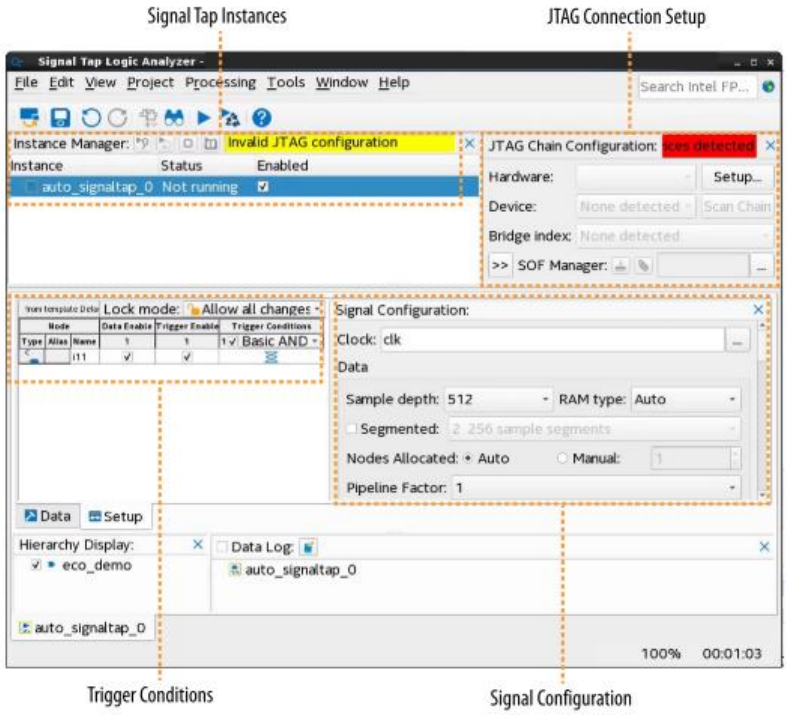


Figura 45: Signal Tap Logic Analyzer GUI. Fonte: [37]

4.2.2. Tempo de atraso “Delay”

Para estabelecer a máxima frequência de operação é necessário conhecer o tempo de atraso *“Delay”* em cada circuito. O fluxograma da Figura 42 apresenta o tempo de atraso dos seguintes circuitos: registradores de entrada, registradores de saída, Fuzificador, Base de Regras, Inferência e Tipo-Redutor. Os resultados apresentados no fluxograma são obtidos utilizando o analisador lógico *Signal Tap* configurado com uma frequência de amostragem de 300Mhz e capacidade de armazenamento de mil amostras. A frequência de atualização das amostras é estabelecida na PLL com uma saída extra c2 (*Figura 46*), que é dedicada exclusivamente ao analisador lógico *Signal Tap*.

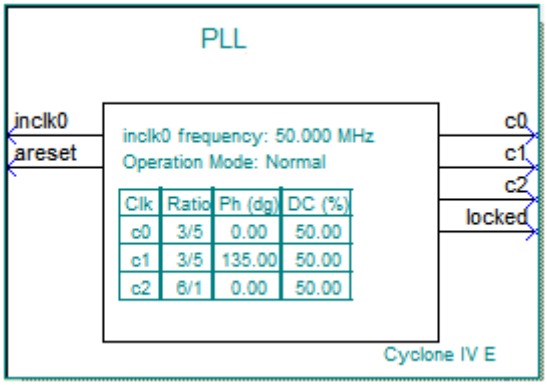


Figura 46: PLL ajustado com clock c0, c1 e com a frequência de aquisição C2

A *Figura 47* apresenta a medição do tempo de atraso para os blocos de *Flip Flops* utilizados com registradores das entradas e da saída do *hardware*. O tempo de +6,67ns, apresentado no segundo cursor, é o tempo de atraso *“Delay”* com relação ao sinal “EN_Memorias” da unidade de controle de regras. O sinal “EN_Memorias” é utilizado como entrada *clock* nos registradores.

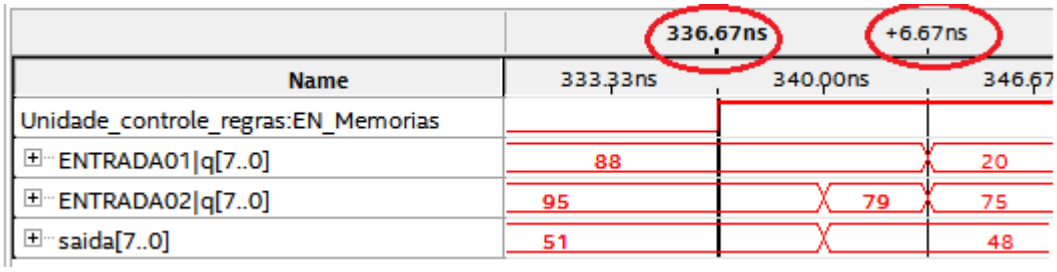


Figura 47: Tempo de atraso Flip Flop

A *Figura 48* apresenta a medição do tempo de atraso para o circuito da Base de Regras (Figura 36). O tempo de +3,33ns, apresentado no segundo cursor, é o tempo que o circuito leva para atualizar o código das regras “Sequencia_regras”, o bit “EN_Memorias” e o bit “Reset_Memoria” com relação ao sinal *clock* nomeado como “clk_out”. A entrada *clock* do circuito “clk_out” é a saída c0 da PLL apresentada na *Figura 46*.

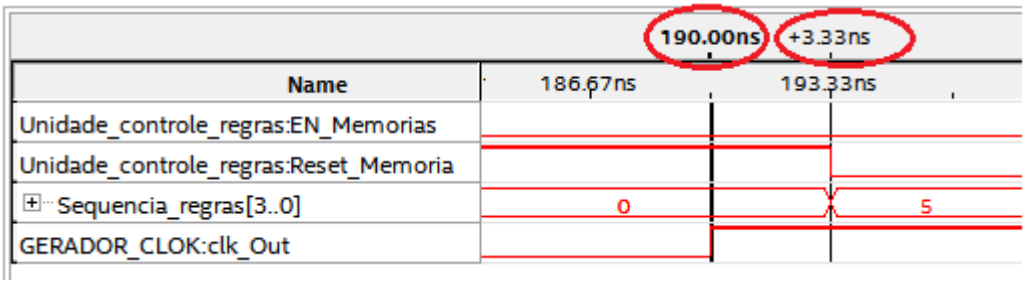


Figura 48: Tempo de atraso base de regras

A *Figura 49* apresenta a medição do tempo de atraso para o circuito Fuzificador (Figuras 21 e 28). O tempo de +10ns, apresentado no segundo cursor, é o tempo máximo que o circuito leva para processar as duas entradas e atualizar as 16 saídas FOU. Com o circuito processa as informações de maneira paralela não existe a necessidade de um pulso de *clock* para sincronismo dos dados.

Figura 49: Tempo de atraso de fuzificação

A Figura 50 apresenta a medição do tempo de atraso para o circuito de Inferência (Figura 29). O tempo de +6,67ns, apresentado no segundo cursor, é o tempo máximo que o circuito leva para processar cada regra com relação ao sinal *clock* nomeado como “clk_out1”. A entrada de *clock* do circuito “clk_out1” está conectada a saída c1 da PLL apresentada na *Figura 46*.

.

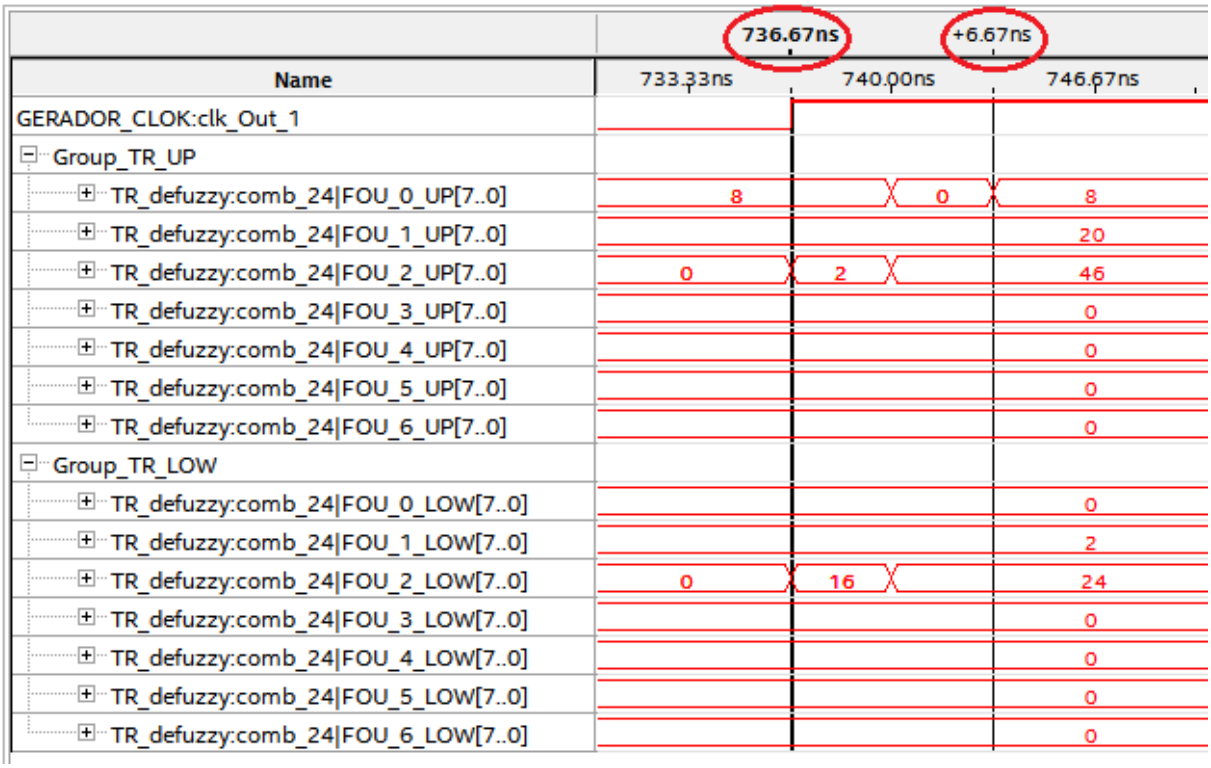


Figura 50: Tempo de atraso Inferência

A Figura 51 apresenta a medição do tempo de atraso para o circuito Tipo-Redutor (*Figura 40*). O tempo de +23,33ns, apresentado no segundo cursor, é o tempo máximo que o circuito leva para processar as 14 entradas, referentes ao grau de pertinência de cada função consequentes, e atualizar a saída de 8bits nomeado com "TR_defuzzy:saida". Com o circuito processa as informações de maneira paralela não existe a necessidade de um pulso de *clock* para sincronismo dos dados.

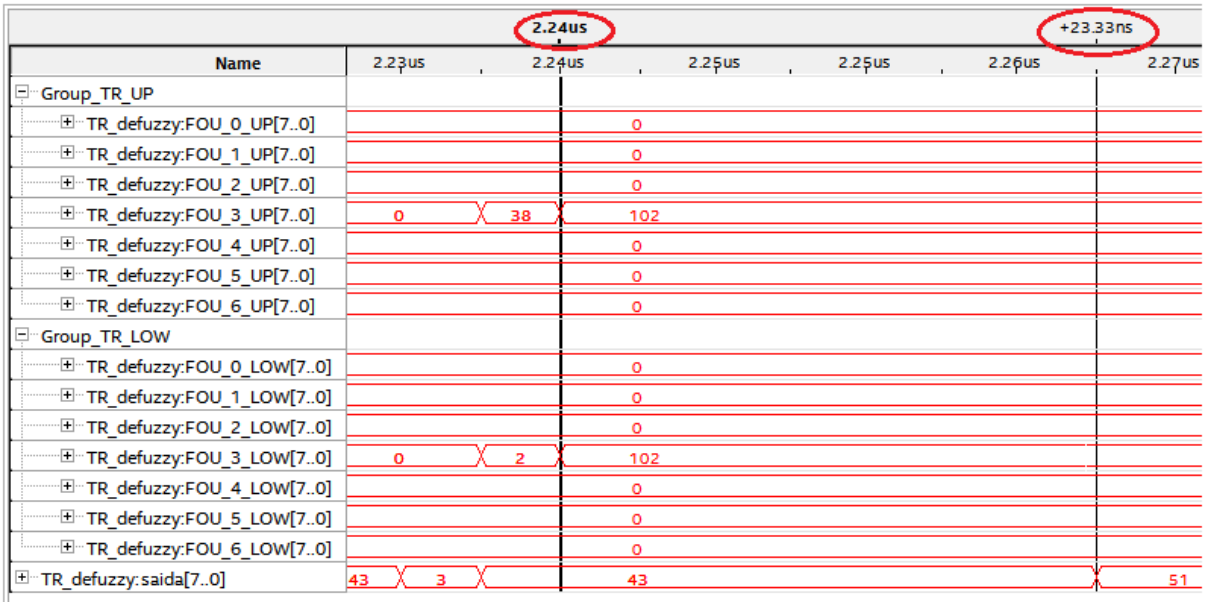


Figura 51:Tempo de atraso Tipo-Redutor

O valor da frequência de *clock* utilizada no circuito depende do maior tempo de atraso no processamento dos dados. Esses "gargalo" é identificado entre a entrada do dado no mecanismo de inferência e a saída do valor *Crisp* do bloco Tipo-Redutor. O tempo de atraso total entre os dois circuitos é determinado por, 6,7ns na Inferência + 23,33ns no Tipo-Redutor, o que totaliza 30ns. Portanto o clock é estabelecido com um período de 33,33ns, ou uma frequência de 30Mhz.

Para atende todo processamento do circuito são utilizados dois sinais de *clocks*, os quais estão nomeados com GERADOR_DE_CLOCK:clk_Out e GERADOR_DE_CLOCK:clk_Out1 (Figura 52). O clock "clk_Out" é aplicado na Base de Regras e não possui atraso de fase. O clock "clk_Out1" é aplicado no mecanismo de inferência e possui um atraso na fase de 135° ou 12,5ns. O atraso no clock "clk_Out1" é necessário porque o mecanismo de Inferência depende dos dados de saída da Base de Regras, a qual possui um tempo de atraso de 3,33ns com relação ao momento de subida do *clock* "clk_Out".

A Figura 52 apresenta todos os sinais que são monitorados no *Signal Tap* para medição de tempo de atraso. O período de 33,33ns estabelecido para ambos os *clocks* é apresentado no segundo cursor.

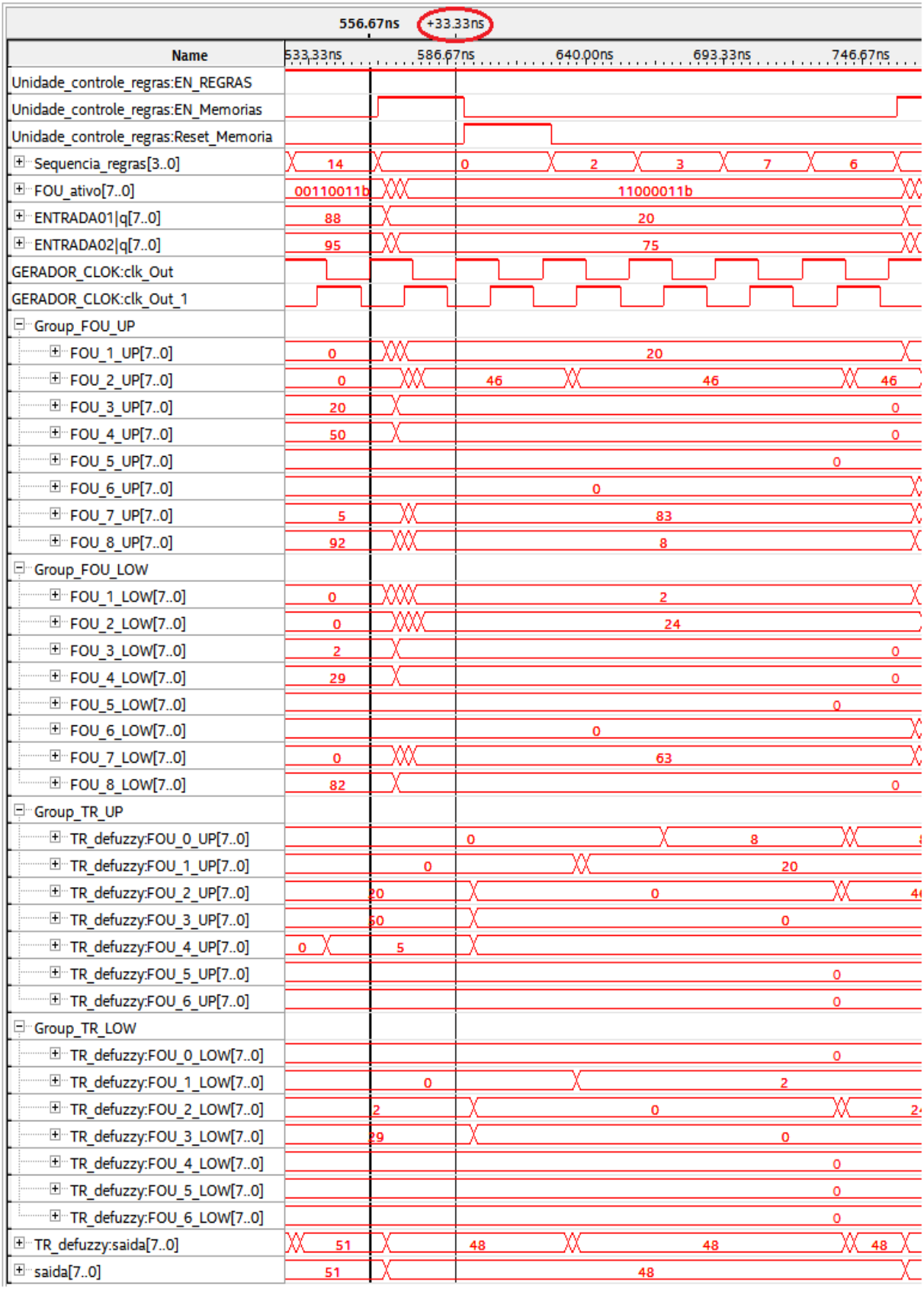


Figura 52: Dados capturados no *Signal Tap*

4.2.3. *Fuzzy logic Inference Per Second* (FLIPS)

A arquitetura implementada utiliza 16 regras, porém, somente uma, duas ou quatro regras podem ser ativas a cada inferência *fuzzy*.

As figuras 53, 54 e apresentam os sinais de saída da base de regras (EN_Memorias, Reset_Memoria e Sequencia_regras), o barramento de 8bit da saída do fuzificador (FOU_ativo), os dois sinais de clock, os valores das entras e o valor da saída do *hardware*. O dados foram aquisitados utilizando o analisador lógico *Signal Tap* ajustado para monitorar os sinais durante o processamento da inferência *fuzzy*.

O bit "EN_Memorias" atualiza as duas entradas e a saída, porém, a saída é atualizada sempre com o resultado da inferência anterior. O bit "Reset_Memoria" limpa os dados registrados referentes a inferência anterior antes de iniciar o processamento das regras atuais. O barramento de 8bits "FOU_ativo" é utilizado como entrada na base de regras para indicar quais funções de pertinência estão ativas. E por fim, o barramento de 4 bits "Sequencia_regras" indica de maneira sequencial as regras ativas. O tempo de processamento do *hardware* é medido entre o momento de atualização das entradas 01 e 02 com relação ao momento de atualização do valor na saída.

No exemplo da Figura 53 a entrada 01 é atualizada com valor de 88 e a entrada 02 é atualizada com o valor de 95, as duas entradas são atualizadas no instante 636.67ns. O barramento FOU_ativo indica que duas funções de pertinência são ativas pela entrada 01 e duas funções de pertinência são ativa pela entrada 02. As regras 10, 11, 15 e 14 são indicadas no barramento Sequencia_regras como ativas. O tempo de processamento é indicado no segundo cursor como 200ns. Esse tempo é constituído por seis pulsos de clock de 33.33ns, um clock atualiza as entradas e saída, um clock limpa os dados dos registradores e quatro clocks são utilizados para processar as quatro regras ativas. Nessas condições o hardware tem capacidade de processar 5 milhões de inferências *fuzzy* por segundo ou 5M FLIPS.

No exemplo da Figura 54 a entrada 01 é atualizada com valor de 34 e a entrada 02 é atualizada com o valor de 50, as duas entradas são atualizadas no instante 403.33ns. O barramento FOU_ativo indica que uma função de pertinência é ativas pela entrada 01 e duas funções de pertinência são ativas pela entrada 02. As regras 5 e 6 são indicadas no barramento Sequencia_regras como ativas. O tempo de processamento é indicado no segundo cursor como 133,33ns. Esse tempo é constituído por quatro pulsos de clock de 33.33ns, um clock atualiza as entradas e saída, um clock limpa os dados dos registradores e dois clocks são utilizados para

processar as duas regras ativas. Nessas condições o hardware tem capacidade de processar 7,5 milhões de inferências *fuzzy* por segundo ou 7,5M FLIPS.

No exemplo da Figura 55 a entrada 01 é atualizada com valor de 34 e a entrada 02 é também é atualizada com o valor de 34, as duas entradas são atualizadas no instante 536,67ns. O barramento FOU_ativo indica que apenas uma função de pertinência é ativa para cada entrada. Portanto, somente a regra 5 é indicada no barramento Sequencia_regras como ativa. O tempo de processamento é indicado no segundo cursor como 100ns. Esse tempo é constituído por três pulsos de clock de 33.33ns, um clock atualiza as entradas e saída, um clock limpa os dados dos registradores e um clocks é utilizado para processar a única regra ativa. Nessas condições o hardware tem capacidade de processar 10 milhões de inferências *fuzzy* por segundo ou 10M FLIPS.

Portanto, a quantidade de inferência *fuzzy* por segundo, FLIPS, é determinada pela frequência do *clock* utilizado e a quantidade de regras ativas. A Tabela 5 apresenta a quantidade de FLPIS que o *hardware* oferece.

Tabela 5: Quantidade de FLIPS

Regras ativas	Nº de Clocks	Tempo de processamento	FLIPS
1	3	100 ns	10 M
2	4	133,33 ns	7,5 M
4	6	200 ns	5 M

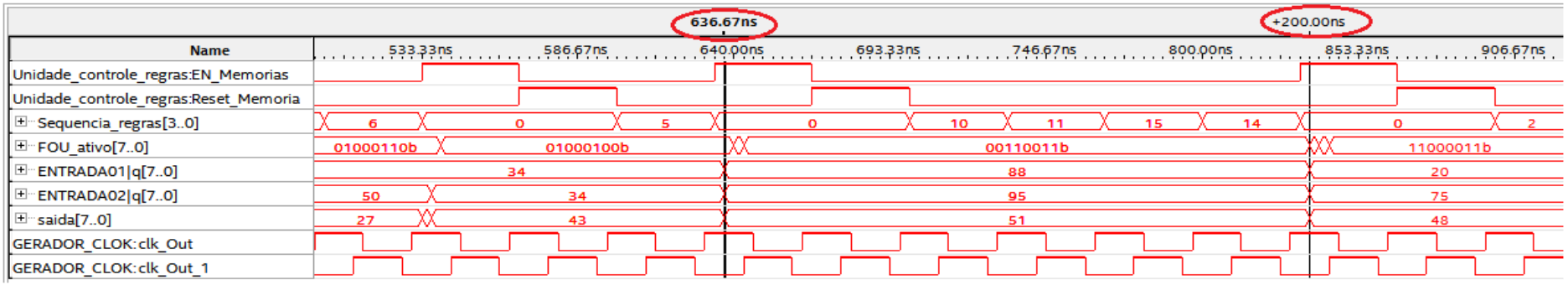


Figura 53: Tempo de processamento para 4 regras ativas

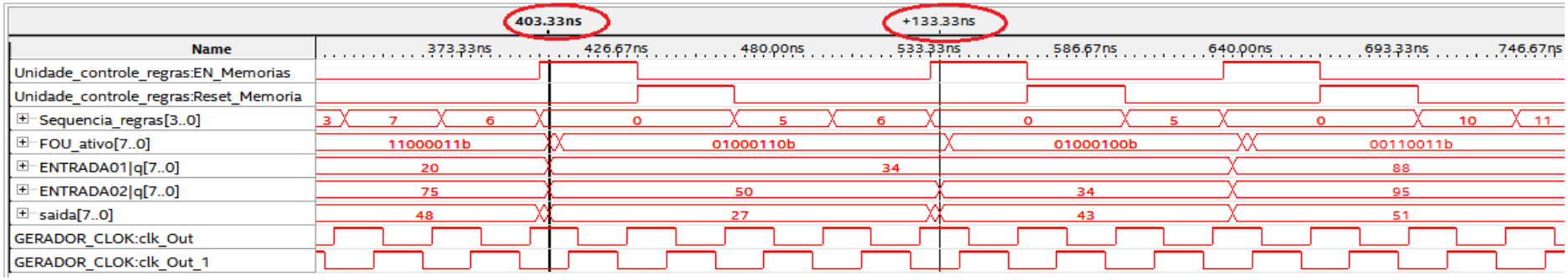


Figura 54:Tempo de processamento 2 regras ativas

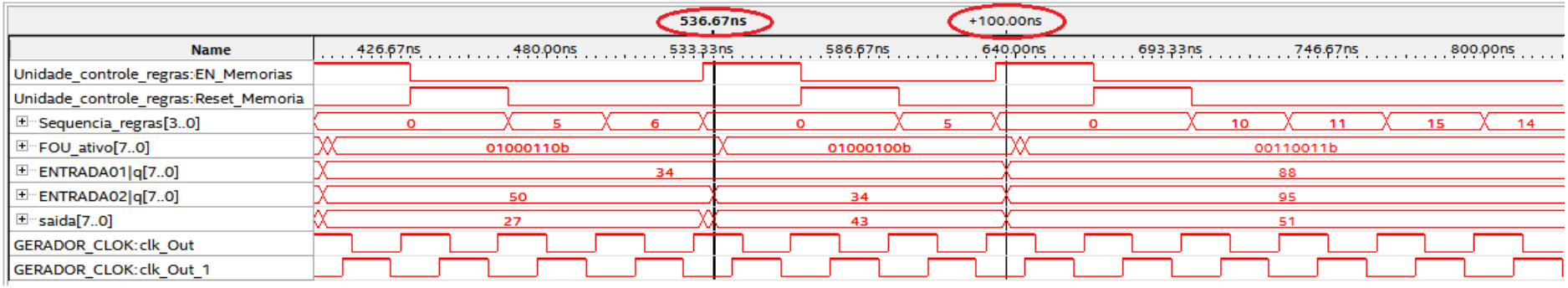


Figura 55: Tempo de processamento 1 regras ativas

4.2.4. Validação de resultados

Para validar o *hardware* de Sistema de Inferência Fuzzy (SIF) tipo-2 intervalar, implementado na FPGA DE2-115, uma plataforma de teste (*Figura 56*) é programada para gerar os dados nas entradas 01 e 02 e monitorar as variáveis internas do circuito. Com pode ser observado na *Figura 56*, a entrada 01 do SIF recebe os dados do contador 1 que inicia sua contagem em 0 e finaliza em 100, já a entrada 02 recebe os dados do contador 2 que também é configurado para iniciar a contagem em 0 e finaliza em 100. Sendo assim, cada contador gera um total de 101 valores. A atualização do contador 1 é feita por um *clock* de 1Mhz, já o contador 2 só recebe pulso de *clock* quando o contado 1 atinge o valor 100. Dessa maneira, o contador 2 só atualiza seu valor de saída após o contador 1 reiniciar sua contagem em 0. A combinação dos resultados de ambos contadores gera 10201 possibilidades de pares de entrada. Com o SIF é configurado previamente para trabalhar com o universo de discurso variando de 0 a 100 em cada entrada, os contadores 1 e 2 cobrem todas as possíveis combinações de valores das duas entradas. Ambos contadores possuem os mesmos botões de *Enable e Reset,* possibilitando assim, parar a ou reinicia a contagem a qualquer momento.

O monitoramento das variáveis é feito de duas maneira. A primeira é utilizando os display de 7segmentos, disponíveis na placa da FPGA, para visualizar os dados de entrada e saída do circuito. A segunda é utilizando o analisador lógico *Signal Tap,* o qual é apresentado na *Figura 56* e é discutido na seção (4.2.1). O *clock* utilizado para aquisição dos dados é o mesmo utilizado para atualizar as entradas, dessa forma, a cada mudança de dado na entrada uma amostra da variável monitorada é aquisitada. Os dados são armazenados na memória do *Signal Tap e* transmitido para o computador via cabo USB. A análise pode ser feita pela interface gráfica *Signal Configuration* ou os dados podem ser baixados em forma de arquivo (.txt).

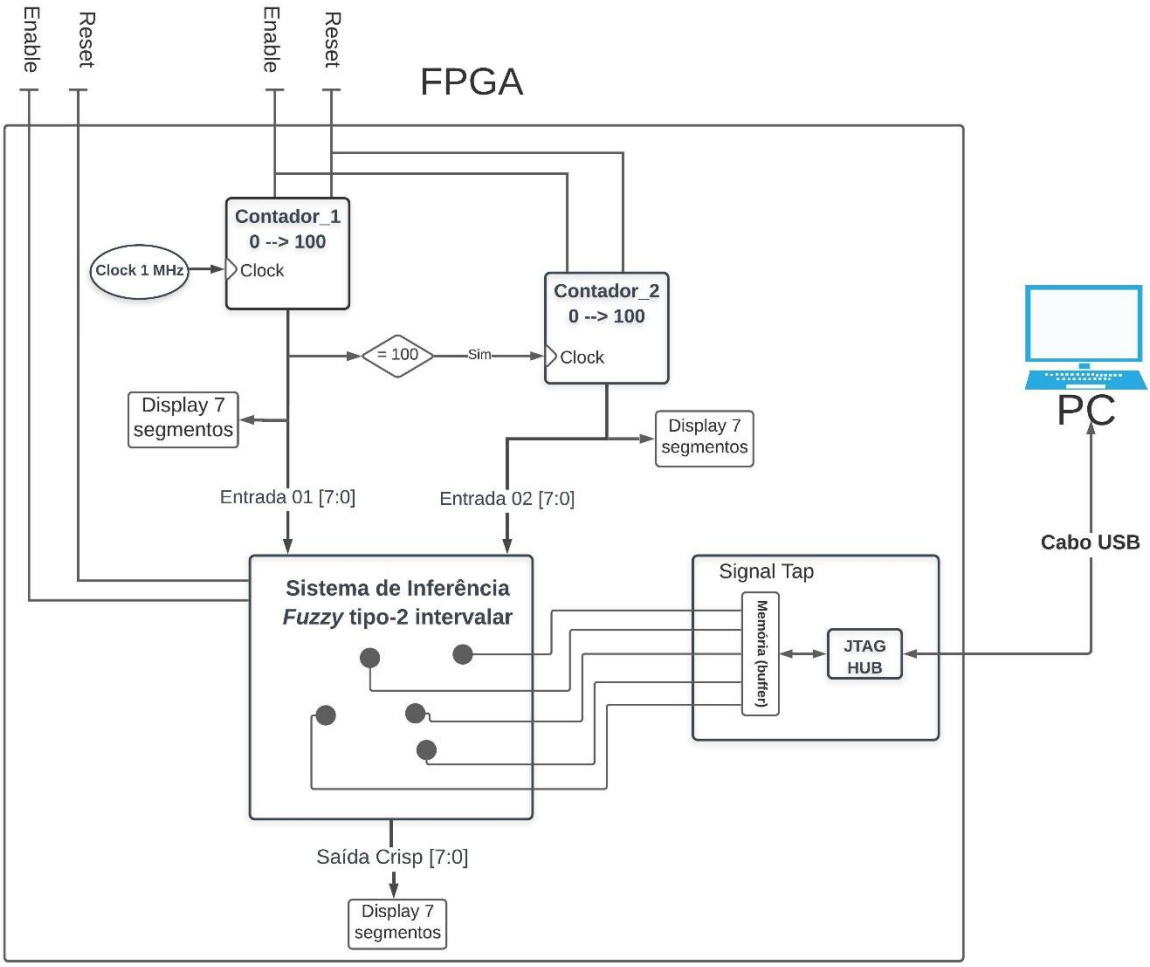


Figura 56:Plataforma de teste

4.2.4.1. Funções antecedentes

A Figura 57 apresenta as funções gaussianas antecedentes implementadas no circuito fuzificador do *hardware* de SIF tipo-2 intervalar. O exemplo adotado para o teste é o sistema de controle para frenagem de locomotiva, abordado na seção (3.2.2.3), onde a entrada 01 representa o valor da velocidade (km/h) da locomotiva e a entrada 02 representa a distância (metros) para parada da locomotiva. Os dados são obtidos com plataforma de teste, apresentada na *Figura 56*, em dois momentos. Para as funções FOU_1, FOU_2, FOU_3 e FOU_4, referentes a entrada 01, o analisador lógico *Signal Tap* é configurado com a frequência de atualização da entrada 01. Para as funções FOU_5, FOU_6, FOU_7 e FOU_8, referentes a entrada 02, o analisador lógico *Signal Tap* é configurado com a frequência de atualização da entrada 02. Dessa maneira é possível observar as características de resposta de cada circuito FOU com sua entrada variando em todo universo de discurso.

Os dados são exportado para Matlab® e filtrados no intervalo do universo de discurso para a entrada 01 de 0 a 100 Km/h e para a entrada 02 de 0 a 100 metros.

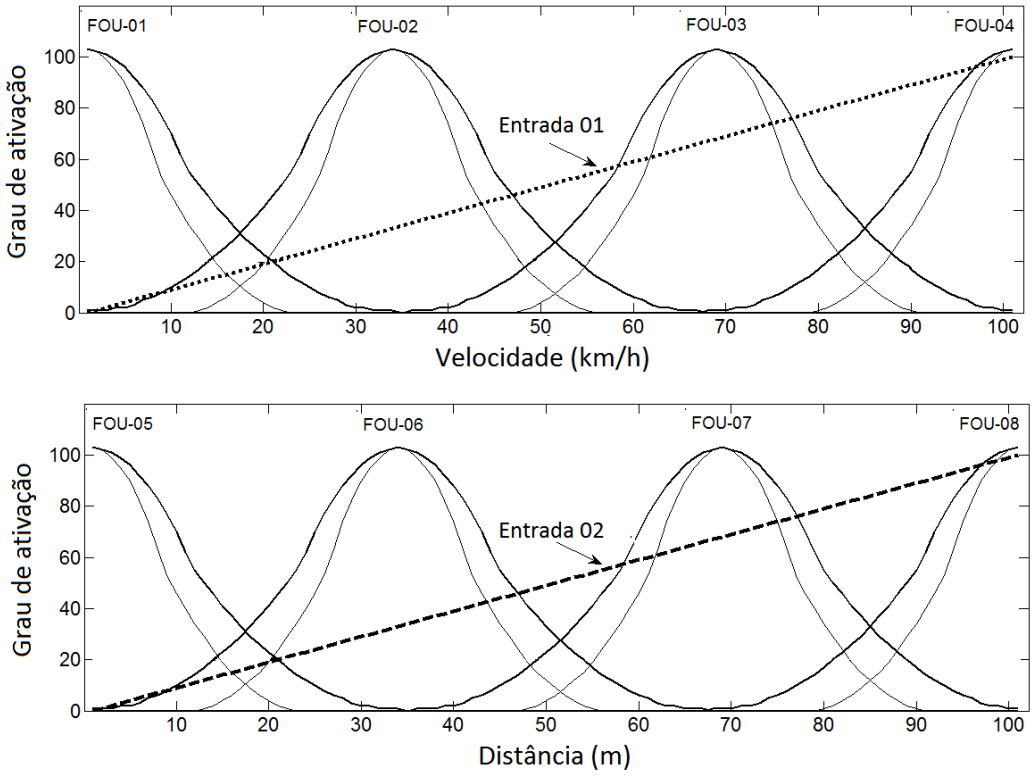


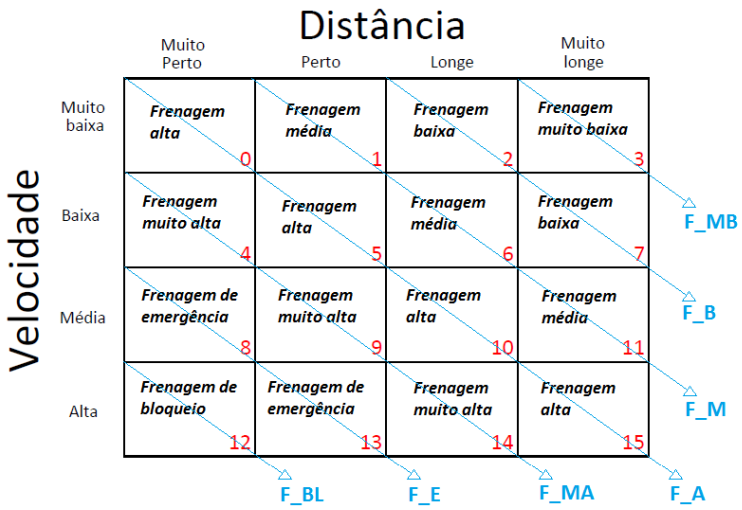
Figura 57: Funções antecedentes

4.2.5. Funções consequentes

O analisador lógico *Signal Tap*, é configurado para monitorar as sete saídas das funções superiores do bloco de inferência utilizando a plataforma de teste apresentada na *Figura 56*. Para ter acesso a uma interface com mais recursos gráficos, os dados obtidos são exportados para o Matlab®. Os valores referentes a entrada 01, que no exemplo da seção (3.2.2.3) representa velocidade da locomotiva, são alocados no eixo Y dos gráficos. Os dados da entrada 02, que são referentes a distância de parada da locomotiva, são alocados no eixo X dos gráficos. Já o grau de ativação da função consequente é alocado no eixo Z. Dessa forma é possível visualizar em um gráfico de 3 dimensões todas as possibilidade de ativação de cada função consequente.

É importante ressaltar que uma função consequente pode ser relacionada com mais de uma regras, por exemplo: a função “Frenagem de Emergência”, Figura 63, é ativada pelas regras 8 e 13, já a função “Frenagem Alta”, Figura 61, é ativada pelas regras 0, 5, 10 e 15. A Tabela 6 apresenta a relação de toda as regras com suas respectivas funções antecedentes e consequentes.

Tabela 6: Base de regras para sistema de frenagem



Na Figura 58 é apresentada a superfície da função consequente frenagem muito baixa "F_MB". Essa função é relacionada a regra 3 (Tabela 6).

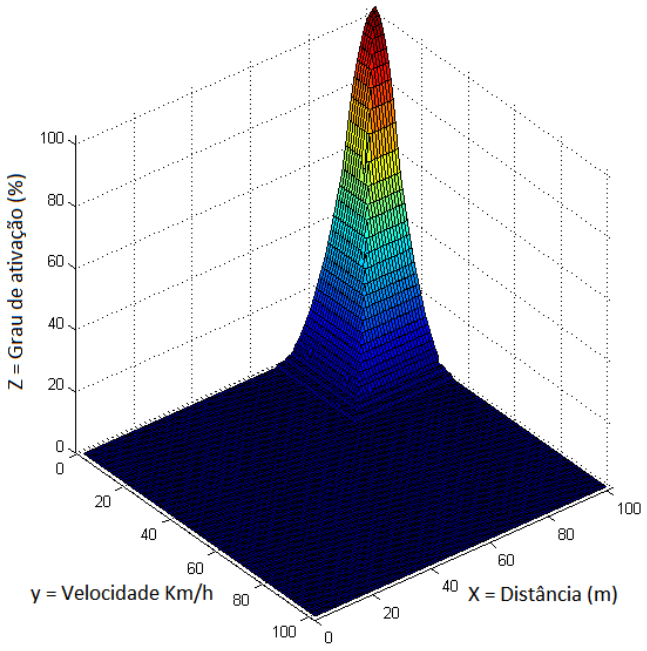


Figura 58: Função consequente Frenagem muito baixa "F_MB"

A superfície apresentada na *Figura 59* é o resultado da função consequente frenagem baixa "F_B". Essa função é relacionada as regras 2 e 7 (Tabela 6).

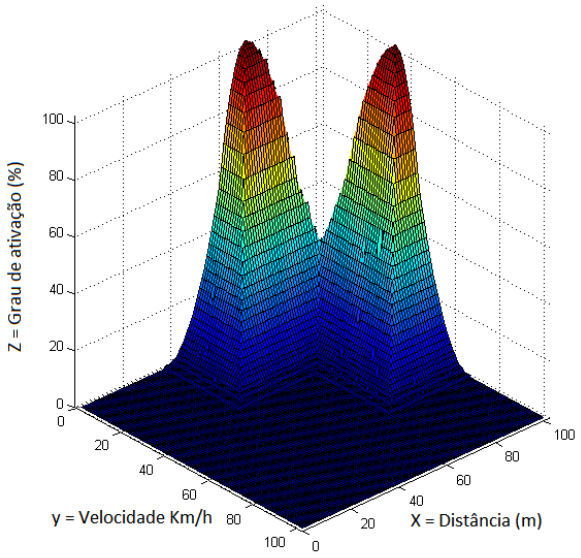


Figura 59: Função consequente frenagem baixa “F_B”

A superfície apresentada na *Figura 60* é o resultado da função consequente frenagem média “F_M”. Essa função é relacionada as regras 1, 6 e 11 (Tabela 6).

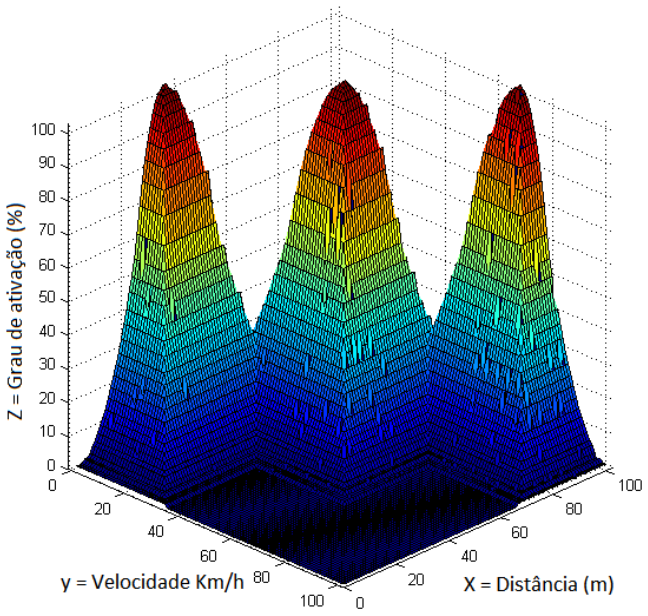


Figura 60: Função consequente frenagem média “F_M”

A superfície apresentada na *Figura 61* é o resultado da função consequente frenagem alta “F_A”. Essa função é relacionada as regras 0, 5, 10 e 15 (Tabela 6).

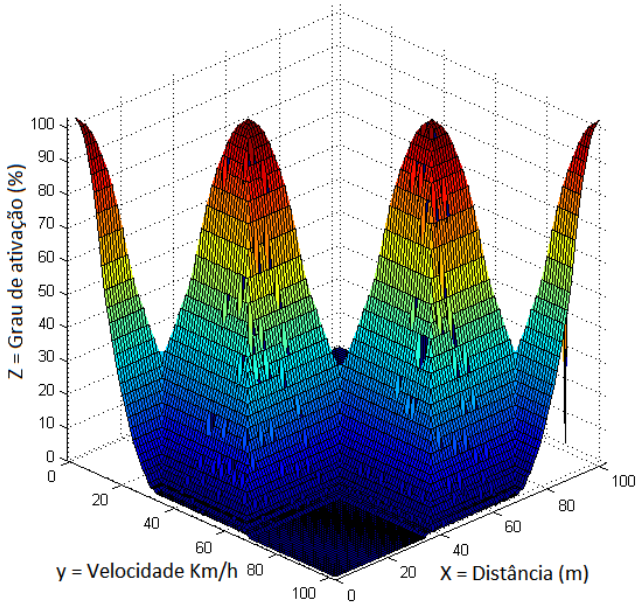


Figura 61: Função consequente frenagem alta “F_A”

A superfície apresentada na *Figura 62* é o resultado da função consequente frenagem muito alta “F_MA”. Essa função é relacionada as regras 4, 9 e 14 (Tabela 6).

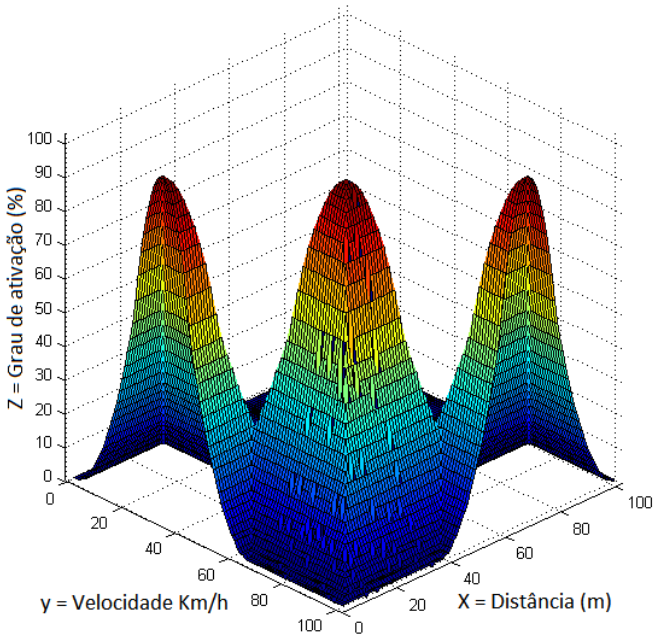


Figura 62: Função consequente frenagem muito alta “F_MA”

A superfície apresentada na *Figura 63* é o resultado da função consequente frenagem de emergência “F_E”. Essa função é relacionada as regras 8 e 13 (Tabela 6).

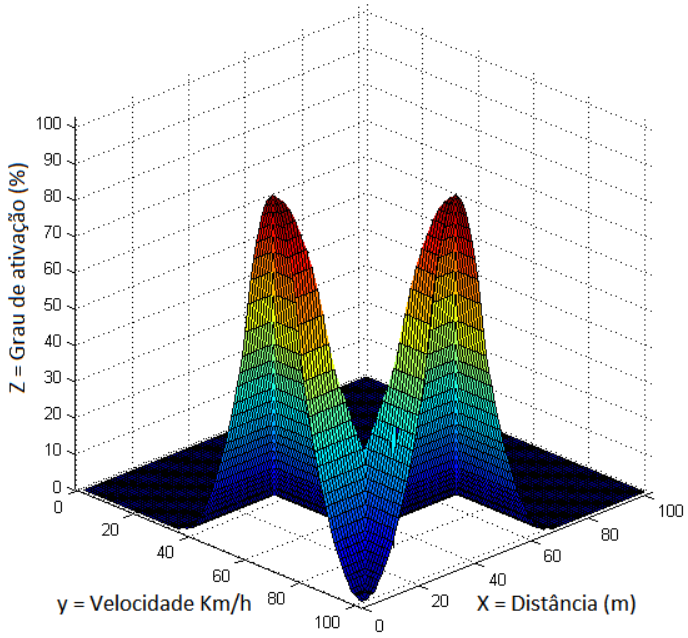


Figura 63: Função consequente frenagem de emergência “F_E”

Na *Figura 64* é apresentada a superfície da função consequente frenagem de bloqueio “F_BL”. Essa função é relacionada a regra 12 (Tabela 6).

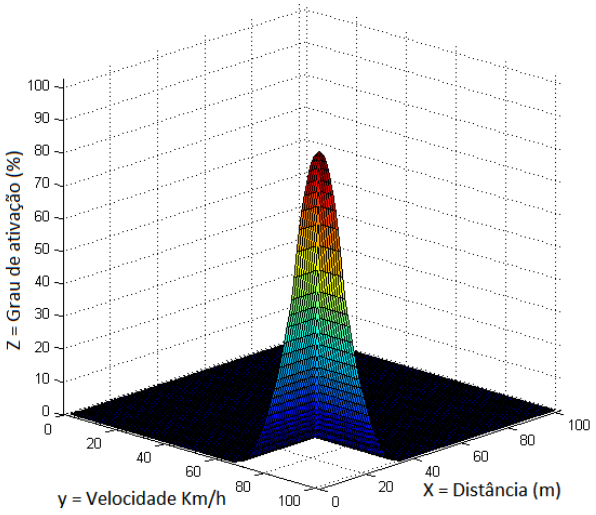


Figura 64: Função consequente frenagem de bloqueio “F_BL”

4.2.6. Superfície de controle

Para validar os resultados da arquitetura em FPGA, é implementado no Matlab®, utilizando o *Interval Type-2 Fuzzy Logic Toolbox* [24], a mesma configuração do Sistema de Inferência Fuzzy (SIF) tipo-2 intervalar apresentada no exemplo de controle de frenagem de

locomotiva abordado na seção (3.2.2.3). Os dados obtidos na saída do Matlab® são utilizados como referência de resposta para a saída *Crisp* da FPGA.

O analisador lógico Signal Tap é configurado para monitorar as duas entradas e saída *Crisp* da arquitetura implementada em FPGA. A geração dos sinais de entrada é feita pela plataforma de teste apresentada na Figura 56. Os dados obtidos são exportados para o Matlab® e as duas entradas da arquitetura, implementada na Toolbox para fuzzy tipo-2 do Matlab® (Figura 65), utilizam os resultados dos dois contadores aplicado na plataforma de teste (Figura 56).

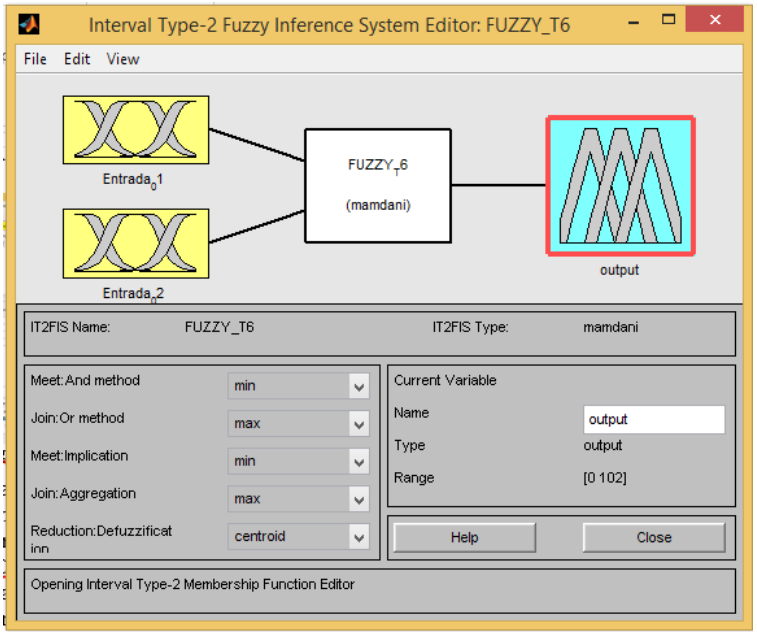


Figura 65: Tool box Matlab® fuzzy tipo-2

Para visualizar os resultados em uma superfície de controle os valores da entrada 01, que representa a velocidade da locomotiva, são alocados no eixo Y. Os dados da entrada 02, que são referentes a distância de parada da locomotiva, são alocados no eixo X. Já os valores da saída *Crisp*, que se referem a porcentagem de frenagem da locomotiva, são alocados no eixo Z.

Dessa forma é possível visualizar em um gráfico de 3 dimensões todas as possibilidade de entrada nos eixo X e Y com a resposta do sistema representada no eixo Z. A Figura 66 apresenta a resposta de controle para os dois sistemas. Observa-se que a resposta de controle gerada pela FPGA é coerente com a resposta de controle gerada pelo SIF tipo-2 intervalar do Matlab®, validando a arquitetura de hardware em FPGA apresentada neste trabalho

.

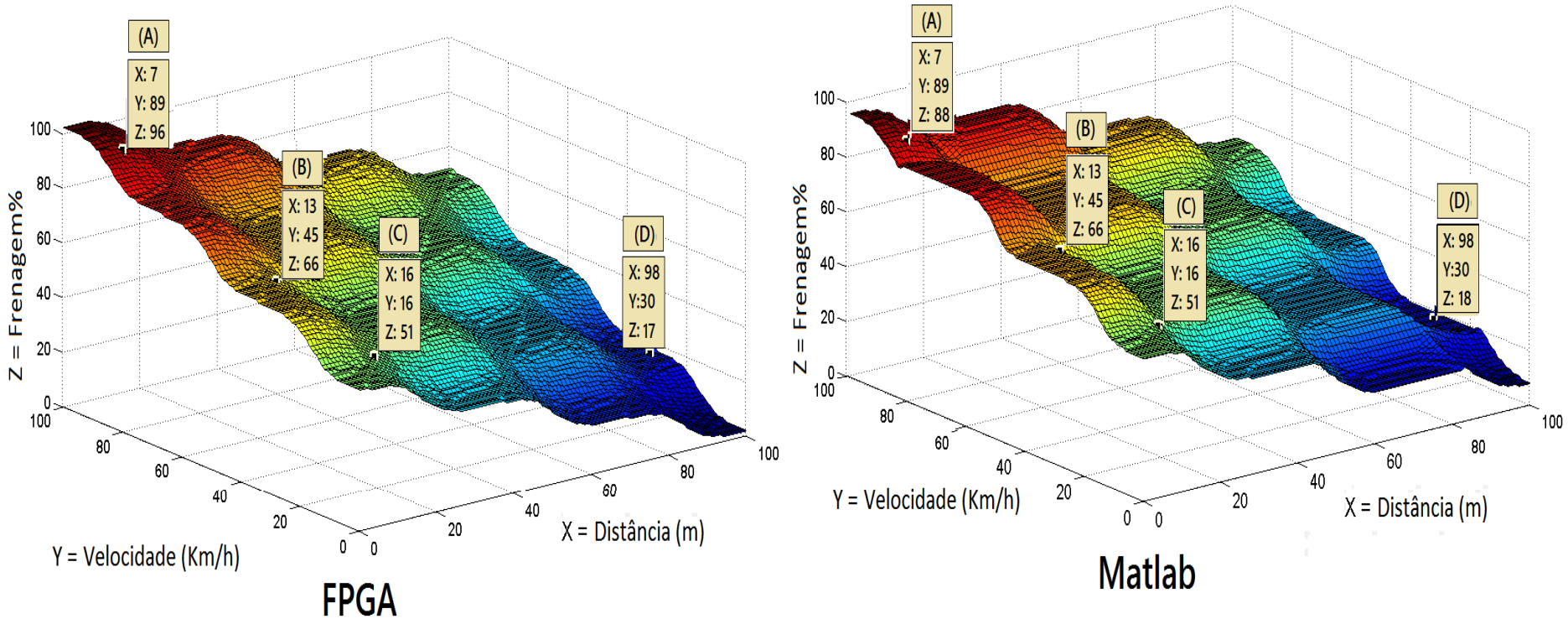


Figura 66: Superficie de controle FPGA x Matlab®

4.2.7. Característica do *hardware*

O hardware implementado apresenta uma taxa de ocupação da FPGA DE2 115 que é apresentada na Tabela 7.

Tabela 7: Taxa de ocupação EP4CE115F29C7 FPGA

Parameter	Value
Operation Frequenncy	30 Mhz
PLL	1(25%)
Elements logic	4095 (4%)
Registers	2306 (2%)
Memory bits	295912 (7%)

A redução de velocidade em processamento série, fez com que os principais trabalhos já publicados (Tabela 8) optassem por um processamento totalmente paralelo para atende aplicações de controle em tempo real.

As arquiteturas com processamento paralelo, no mecanismo de inferência, necessitam de um circuito dedicado a cada regra existente, ou seja, quanto maior o número de regras maior é o número de circuitos dedicados no mecanismo de inferência. Por exemplo, o trabalho apresentado na referência [16] (Tabela 8) possui 16 regras, porém, por conta do processamento paralelo no mecanismo de inferência utiliza 16 circuito para processar as 16 regras.

Tabela 8: Trabalhos publicados sobre SIF tipo-2 intervar em FPGA

Ref.	Modelo da FPGA	N° de FLIPS	Tipo-Redutor	Entradas	Saídas	Regras
[16]	Virtex XC2V3000FF1152	32 M	WM	2	1	16
[17]	Spartan 3 e Virtex 5	12,5 M 25 M	KM	2	1	25
[18]	Nexys3 Spartan-6	16.67 M	BMM	2	1	9
[19]	Spartan-6 XC6SLX75	13.33 M	NT	2	1	25
		13.33 M	BMM			
		10 M	WM			
		8M	KM			
[20]	Spartan 6	44.2 M	WM	4	1	9
[36]	Virtex XC2V3000FF1152	30M	WM	2	1	9
Trabalho apresentado	DE2 EP4CE115F29C7N	5M – 10M	NT	2	1	16

O gerenciamento das regras ativas, apresentado neste trabalho, possibilita integrar o mecanismo de inferência, com processamento série, aos blocos de fuzificação e tipo-redutor,

com processamento paralelo. Dessa forma, a inferência processa 16 regras em um único circuito série, mantendo a quantidade de *Fuzzy logic Inference Per Second* (FLIPS) de 5M a 10M, dependendo do número de regras ativas.

Um outro fator que limita a quantidade de FLIPS é o algoritmo de tipo-redutor utilizado na arquitetura. O algoritmo de Nie-Tan, adotado neste trabalho, também é utilizado na referência [19] (Tabela 8), a qual possui processamento totalmente paralelo e apresenta uma quantidade de FLIPS de 13,33M.

5. Conclusões e Sugestões para trabalhos futuros

5.1. Conclusão

Este trabalho propôs e validou uma arquitetura para um Sistemas de Inferência Fuzzy (SIF) tipo-2 intervar implementada em FPGA. O hardware apresentado possui como característica duas entradas e uma saída, sendo composto pelos seguintes módulos: fuzificador tipo-2 com processamento paralelo, inferência baseado no método de Mamdani com processamento série, base de regras implementada em máquina de estados e tipo-redutor baseado no algoritmo de Nie-Tan com processamento paralelo.

A arquitetura proposta permite programar a largura da mancha incerteza *Footprint Of Uncertainty* (FOU) e a possição de cada função de pertinência no universo de discurso. A largura do FOU também pode ser ajustada para operar como um sistema de inferencia *fuzzy* tipo-1. Desta maneira a arquitetura proposta tem a vantagem obter tanto um SIF tipo-2 como um SIF tipo-1.

Os resultados das simulações dos módulos e da arquitetura completa deste trabalho, mostram que o *hardware* apresenta velocidade de processamento de 5M a 10M *Fuzzy logic Inference Per Second* (FLIPS), dependendo do número de regras ativas, o que o torna compatível para aplicações de controle em tempo real.

Por fim, a arquitetura com os módulos de fuzificação, 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*.

5.2. Sugestões para trabalhos futuros

- O principal limitador na velocidade de processamento do *hardware* é o tempo de resposta do circuito Tipo-Redutor. A quantidade de circuito multiplicadores utilizados para calcular o valor *crisp* de saída faz com que o "*Delay*" do circuito fique em 23,33ns. Para um trabalho futuro é interessante que esse tempo seja reduzido com propostas de circuitos multiplicadores mais rápidos. Essa solução abre a possibilidade de um aumento no número de FLIPS do circuito.
- O *hardware* apresentado possui liberdade na configuração do sistema, entretanto, as configurações só podem ser feitas ajustando os paramentos do código em *Verilog*.

Para facilitar a operação do *hardware,* é importante que uma *interface* gráfica, semelhante a utiliza no Matlab®, seja elaborada para que a configuração do hardware se torne mais fácil.

- Implementar o sistema de inferência *fuzzy* tipo-2 intervalar proposto em uma planta de controle em tempo real.

Referências Bibliográficas

[1] L. A. Zadeh, “Fuzzy Sets,” *Inf. CONTRO*, vol. 90, no. 1, pp. 103–107, 1965.

[2] T. J.Ross, *Fuzzy Logic With Engineering Applications*. 2010.

[3] E. H. Mamdani and S. Assilian, “An experiment in linguistic synthesis with a fuzzy logic controller,” *Int. J. Man. Mach. Stud.*, vol. 7, no. 1, pp. 1–13, 1975.

[4] Z. Kovacic and S. Bogdan, *Fuzzy Controller Design Theory and Applications*, CRC Press. Broken Sound Parkway NW, 2005.

[5] S. A. Loan, A. M. Murshid, S. A. Abbasi, and A. R. M. Alamoud, “A novel VLSI architecture for a fuzzy inference processor using Gaussian-shaped membership function,” *J. Intell. Fuzzy Syst.*, vol. 24, no. 1, pp. 5–19, 2013.

[6] L. A. Zadeh, “The concept of a linguistic variable and its application to approximate reasoning-I,” *Inf. Sci. (Ny).*, vol. 8, no. 3, pp. 199–249, 1975.

[7] P. M. S. R. Rizol, L. Mesquita, and O. Saotome, “Lógica Fuzzy tipo-2,” *Sodebras*, vol. 6, pp. 27–46, 2011.

[8] J. M. Mendel, “Type-2 fuzzy sets as well as computing with words,” *IEEE Comput. Intell. Mag.*, vol. 14, no. 1, pp. 82–95, 2019.

[9] N. N. Karnik and J. M. Mendel, “Introduction to type-2 fuzzy logic,” *Introd. to type-2 fuzzy Log. Syst.*, vol. 223, pp. 1–4, 1998.

[10] Q. Liang and J. M. Mendel, “Interval type-2 fuzzy logic systems,” *IEEE Int. Conf. Fuzzy Syst.*, vol. 1, pp. 328–333, 2000.

[11] D. Wu and J. M. Mendel, “Enhanced Karnik-Mendel algorithms,” *IEEE Trans. Fuzzy Syst.*, vol. 17, no. 4, pp. 923–934, 2009.

[12] H. Wu and J. M. Mendel, “Uncertainty bounds and their use in the design of interval type-2 fuzzy logic systems,” *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 5, pp. 622–639, 2002.

[13] M. A. Melgarejo, G. R. Antonio, and C. A. Peña-Reyes, “Pro-Two: A hardware based platform for real time type-2 fuzzy inference,” *IEEE Int. Conf. Fuzzy Syst.*, vol. 2, pp. 977–982, 2004.

[14] M. Nie and W. W. Tan, “Towards an efficient type-reduction method for interval type-2 fuzzy logic systems,” *IEEE Int. Conf. Fuzzy Syst.*, vol. 2, pp. 1425–1432, 2008.

[15] M. Biglarbegian, W. W. Melek, and J. M. Mendel, “On the stability of interval type-2 tsk fuzzy logic control systems,” *IEEE Trans. Syst. Man, Cybern. Part B Cybern.*, vol.

40, no. 3, pp. 798–818, 2010.

[16] M. Melgarejo and C. A. Pena-Reyes, “Implementing Interval Type-2 Fuzzy Processors,” *IEEE Comput. Intell. Mag.*, vol. 2, no. 1, pp. 63–71, 2007.

[17] R. Sepúlveda, O. Montiel, O. Castillo, and P. Melin, “Embedding a high speed interval type-2 fuzzy controller for a real plant into an FPGA,” *Appl. Soft Comput. J.*, vol. 12, no. 3, pp. 988–998, 2012.

[18] M. D. Schrieber and M. Biglarbegian, “Hardware implementation of a novel inference engine for interval type-2 fuzzy control on FPGA,” *IEEE Int. Conf. Fuzzy Syst.*, pp. 640–646, 2014.

[19] M. D. Schrieber and M. Biglarbegian, “Hardware implementation and performance comparison of interval type-2 fuzzy logic controllers for real-time applications,” *Appl. Soft Comput. J.*, vol. 32, pp. 175–188, 2015.

[20] E. Ontiveros-Robles, J. L. Gonzalez-Vazquez, J. R. Castro, and O. Castillo, “A hardware architecture for real-Time edge detection based on interval type-2 fuzzy logic,” *IEEE Int. Conf. Fuzzy Syst. FUZZ-IEEE 2016*, pp. 804–810, 2016.

[21] A. Gaona, D. Olea, and M. Melgarejo, “Distributed arithmetic in the design of high speed hardware fuzzy inference systems,” *IEE Xplore Annu. Conf. North Am. Fuzzy Inf. Process. Soc. - NAFIPS*, pp. 116–120, 2003.

[22] E. Frías-Martínez, “Real-time fuzzy processor on a DSP,” *IEEE Int. Conf. Emerg. Technol. Fact. Autom. ETFA*, vol. 1, pp. 403–408, 2001.

[23] D. Wu and J. M. Mendel, “Recommendations on designing practical interval type-2 fuzzy systems,” *IEEE Int. Conf. Fuzzy Syst.*, vol. 85, no. February, pp. 800–807, 2019.

[24] J. R. Castro, O. Castillo, and L. G. Martínez, “Interval Type-2 Fuzzy Logic Toolbox,” *IEEE Int. Fuzzy Syst. Conf.*, 2007.

[25] M. Olarte, F. Ladino, P. Melgarejo, “Hardware realization of Fuzzy Adaptive Filters for non linear channel equalization,” *IEEE Int. Symp. Circuits Syst.*, pp. 932–935, 2005.

[26] P. M. Silva Rocha Rizol and R. Alves Dias, “DESMISTIFICANDO A LÓGICA FUZZY.” Congresso Brasileiro de Educação em Engenharia (COBENGE), 2014.

[27] O. Castillo and P. Melin, *Type-2 Fuzzy Logic: Theory and Applications*, Springer. Springer-Verlag Berlin Heidelberg, 2008.

[28] O. Castillo, P. Melin, J. Kacprzyk, and W. Pedrycz, “Type-2 fuzzy logic: Theory and applications,” *Proc. - 2007 IEEE Int. Conf. Granul. Comput. GrC 2007*, pp. 145–150, 2007.

[29] H. Y. Ying, J. M. Mendel, H. Hagras, W. Tan, W. W. Melek, *Introduction to type-2 fuzzy*

logic control theory and applications, vol. 1. 2014.

[30] J. M. Mendel, "General type-2 fuzzy logic systems made simple: A tutorial," *IEEE Trans. Fuzzy Syst.*, vol. 22, no. 5, pp. 1162–1182, 2014.

[31] J. M. Mendel, R. I. John, and F. Liu, "Interval type-2 fuzzy logic systems made simple," *IEEE Trans. Fuzzy Syst.*, vol. 14, no. 6, pp. 808–821, 2006.

[32] S. A. Lopez and M. A. Melgarejo, "Hardware Based Fuzzy Logic Controllers Using Frequency Domain Singleton Fuzzification," pp. 731–736, 2005.

[33] H. A. Hagras, "A hierarchical type-2 fuzzy logic control architecture for autonomous mobile robots," *IEEE Trans. Fuzzy Syst.*, vol. 12, no. 4, pp. 524–539, 2004.

[34] D. Wu and W. W. Tan, "Computationally efficient type-reduction strategies for a type-2 fuzzy logic controller," *IEEE Int. Conf. Fuzzy Syst.*, pp. 353–358, 2005.

[35] R. J. Contreras, "Modelos Neuro-Fuzzy Hierárquicos BSP do Tipo 2," PUC -RJ, 2007.

[36] M. Melgarejo, A. García, C. A. Peña-reyes, M. Melgarejo, A. García, and C. A. Peña-reyes, "Architectural Proposal and Hardware Realization of a Type-2 Fuzzy Inference Sytem," no. 1, 2004.

[37] Intel Corporation, "Intel® Quartus® Prime Pro Edition User Guide - Platform Designer," 2019. [Online]. Available: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug-qpp-platform-designer.pdf. [Accessed: 21-Oct-2020].

APÊNDICE – A: Artigo publicado

R. Maciel; R. L. Moreno; P. M. S. R. Rizol; T. C. Pimenta. FPGA Implementation of Interval Type-2 Fuzzy System Based on Nie-Tan Algorithm. IEEE INTERNATIONAL CONFERENCE ON MICROELECTRONICS – ICM , 2020, Jordan.

APÊNDICE – B: Base de dados dos códigos em verilog

Toda base de dados do código em Verilog para a implementação do sistema de inferência *fuzzy* tipo-2 intervalar implementado na FPGA EP4CE115F29C7 está disponível no link abaixo:
https://drive.google.com/drive/folders/1dqPYeatFR54Bttr0qWZEcXVOrISXGGhe?usp=sharing

O *software* utilizado para edição do código verilog é o Quartus® prime lite versão 18.1. Segue o l*ink* para *download* do software:
https://fpgasoftware.intel.com/18.1/?edition=lite

APÊNDICE – C: Métodos de Defuzificação

Defuzificação é a conversão de um conjunto *fuzzy* em um valor *crisp*, assim como fuzificação é a conversão de valores precisos em um conjunto *fuzzy*. A saída de um processo *fuzzy* pode ser a união de duas ou mais funções de pertinência *fuzzy* definidas no universo do discurso da variável de saída. Por exemplo, suponha que uma saída *fuzzy* compreenda duas partes: uma forma trapezoidal Figura 67 (a) e uma forma triangular Figura 67 (b). A união dessas duas funções, envolvendo o operador máximo, resulta na forma apresentada na Figura 67 (c).

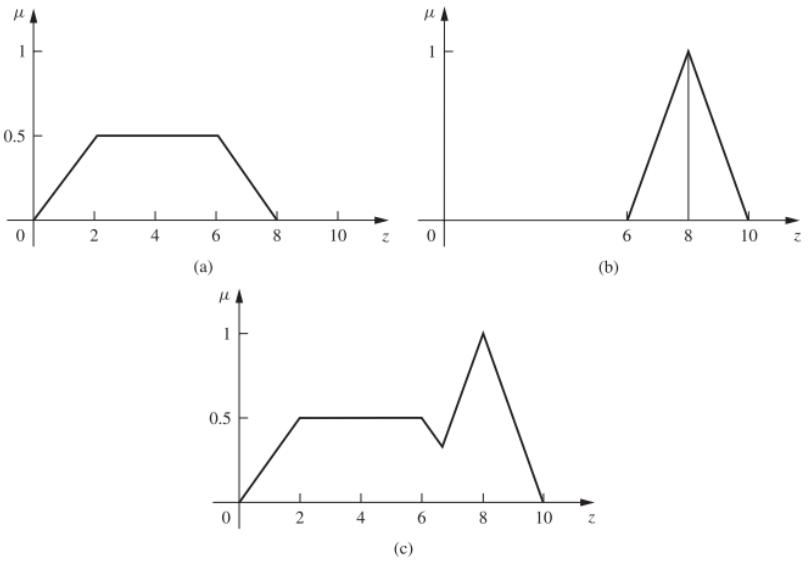


Figura 67: Saída típica do processo fuzzy: (a) primeira parte da saída fuzzy; (b) segunda parte da saída fuzzy; e (c) união de ambas as partes. Fonte:[2].

Um processo de saída *fuzzy* pode envolver várias partes de funções antecedentes. A função representada na saída pode ter formas além de triângulos, trapézios ou gaussianas. Além disso, como mostra a Figura 67 (a), as funções de pertinência nem sempre podem ser normais [2].

Para se converter a informação dos conjuntos *fuzzy* de saída para o domínio real, diversos métodos de defuzificação foram propostos na literatura nos últimos anos, entretanto, somente os mais utilizados serão abordados.

Método da Altura: esse método também é conhecido como centro dos máximos e é uma opção de cálculo mais simples e com menor número de operações. A Equação (2.1) apresenta as operações aplicadas para sistemas discretos, onde:

$c_i =$ a posição da altura máxima da função de pertinência

$h_i =$ a altura da função de pertinência em grau de ativação c_2

n = número de funções de pertinência

$$u^* = \frac{\sum_{i=1}^{n} h_i \,.\, c_i}{\sum_{i=1}^{n} h_i} \qquad (2.1)$$

A Figura 68 apresenta um exemplo de saída *fuzzy* que é defuzificada pelo método da altura na Equação (2.2).

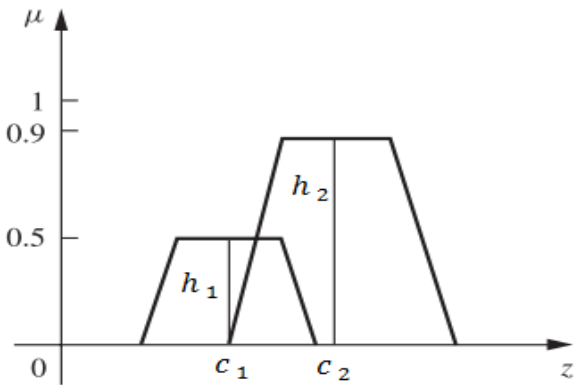


Figura 68: Defuzificação pelo método da altura

$$u^* = \frac{h_1 . c_1 + h_2 . c_2}{h_1 + h_2} \qquad (2.2)$$

Método centro de área: ou centro de gravidade é um dos método mais utilizado, esse cálculo define o centro da função resultante. A Equação (2.3) apresenta sua definição um ambiente contínuo.

$$u^* = \frac{\int_u u \cdot u_u du}{\int_u u_u du} \qquad (2.3)$$

Onde: "u_u" união das funções de pertinência resultante e "u" é a posição da função de pertinência.

Para aplicações em ambiente discretos os cálculos são definidos pela Equação(2.4). Onde n é o número de amostras ou pontos.

$$u^* = \frac{\sum_{i=1}^{n} u \cdot u_u(u_i)}{\sum_{i=1}^{n} u_u(u_i)} \qquad (2.4)$$

Primeiro do Máximo: é o método que usa a saída dos conjuntos *fuzzy* de maneira individuais para determinar o menor valor do domínio com o máximo grau de ativação. Essa operação é feita em três partes

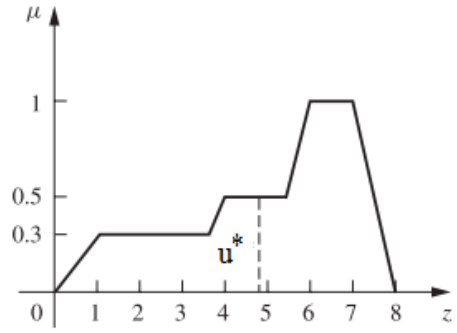


Figura 69: Defuzificação por centro de área

Primeiro, a maior altura da união é determinada:

$$hgt(U) = sup\mu_u(u) \quad onde: u \in U \tag{2.5}$$

Segundo, o conjunto com os graus de pertinência sofre uma comparação, como mostrado na (2.6).

$$hgt(U) = \{u \in U | \mu_U(u) = hgt\ (u)\} \tag{2.6}$$

Então o valor de saída para o primeiro máximo é dado na Equação (2.7).

$$u^* = inf\{u \in U | \mu_U(u) = hgt\ (u)\} \tag{2.7}$$

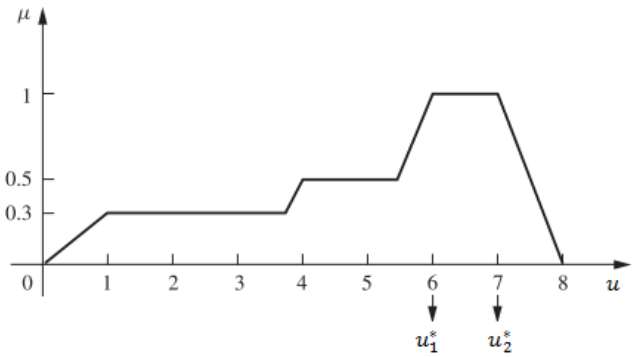


Figura 70: Primeiro máximo = 6 e último máximo = 7.

Último do Máximo: Por analogia as equações que representam o primeiro máximo, também é determinado o método para o ultimo máximo. A Figura 70 representa a formas de saída *fuzzy* com ambos valores *crisp* destacados. A Equação (2.8) define a identificar do último máximo.

$$u^* = sup\{u \in U | \mu_U(u) = hgt\ (u)\} \tag{2.8}$$

Média do Máximo: esse método faz a média do primeiro máximo e o último máximo, os quais acabaram de ser retratados. A Figura 71 apresenta os dados gráficos desse método. A Equação (2.9) apresenta as operações matemáticas.

$$u^* = \frac{inf\{u \in U | \mu_U(u) = hgt\ (u)\} + \ sup\{u \in U | \mu_U(u) = hgt\ (u)\}}{2} \qquad (2.9)$$

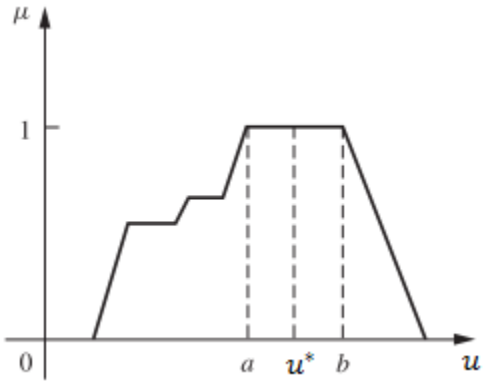


Figura 71: Média dos máximos

Centro de soma: é um método mais rápido se comparado a muitos outros métodos, seu processo de cálculo é parecido como o de centro de área, porém, é utilizado a área de cada função resultante de maneira individual. As desvantagens desse método são que as áreas de intersecção são adicionadas duas vezes e o método também envolve encontrar os centroides das funções individuais de associação. A Figura 72 apresenta dois conjuntos de saídas *fuzzy*, C1 e C2. Como pode ser observado na Figura 72 (c), o resultado da soma das áreas apresenta um intersecção com resultado duplicado.

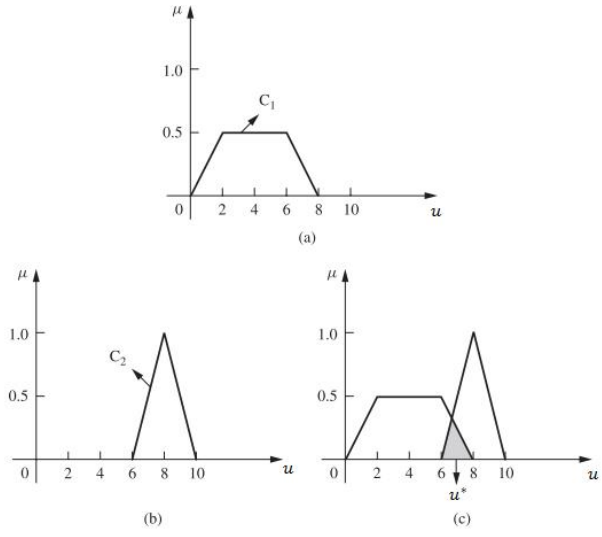


Figura 72: Centro da soma. (a) primeira função, (b)segunda função e (c) conjunto para de defuzicação. Fonte:[2]

Para aplicações em ambiente discreto a Equação (2.10) define o método em questão.

$$u^* = \frac{\sum_{i=1}^{l} u_i \sum_{k=1}^{n} u_{c(k)}(u_i)}{\sum_{i=1}^{l} \sum_{k=1}^{n} u_{c(k)}(u_i)} \qquad (2.10)$$

Para aplicações em ambiente contínuo a Equação (2.11) define o método em questão.

$$u^* = \frac{\int_u u_i . \sum_{k=1}^{n} u_{c(k)}(u) du}{\int_u \sum_{k=1}^{n} u_{c(k)}(u) du} \qquad (2.11)$$

Existem vários outros métodos para defuzificação que não são apresentados aqui. A escolha do melhor método depende do contexto da aplicação, compatibilidade das resposta com as variações da entrada, valores de saída sem ambiguidade, tempo de respostas e principalmente a complexidade computacional exigida. Dos métodos abordados, o da altura, média dos máximos e primeiro máximo são os mais rápidos, entretanto, são menos precisos. Já o método centro de área apresenta resposta mais precisa, porém, possui alta complexidade computacional, o que torna sua respostas lenta [2].