Proposta de um Sistema Fuzzy Takagi-Sugeno para FPGA

Antônio E. M. Silva, Sérgio S. Natan, and Marcelo A. C. Fernandes

Departamento de Computação e Automação (DCA)
Universidade Federal do Rio Grande do Norte (UFRN)
edumorais92@gmail.com, sergionatan@dca.ufrn.br,
mfernandes@dca.ufrn.br

Resumo Este trabalho tem como objetivo apresentar o desenvolvimento de um hardware dedicado de um sistema baseado em lógica Fuzzy para um *Field Programmable Gate Array* (FPGA). O sistema proposto utiliza uma máquina de inferência Takagi-Sugeno e sua implementação utilizou uma estratégia híbrida com partes em ponto-fixo e outras ponto flutuante. Objetivando validar o hardware proposto, o trabalho apresenta resultados através de simulação em precisão de bit para várias resoluções em tamanho de bit. Além dos resultados de validação são apresentadas informações sobre a ocupação e *throughput* do hardware proposto. O hardware utilizou como alvo um FPGA Xilinx Virtex 6 xc6vlx240t-1ff1156. *abstract*

Keywords: Sistema Fuzzy, Takagi-Sugeno, FPGA, Hardware dedicado.

1 Introdução

Sistemas baseados em Lógica Fuzzy, também conhecida como lógica nebulosa, têm sido utilizados em diversas aplicações industriais e comerciais como robótica, automação, controle, problemas de classificação e outras. Uma das grandes vantagens é a possibilidade de trabalhar com informações imprecisas, diferentemente de outros tipos de lógica. Sistemas inteligentes com base em regras de produção podem facilmente utilizar a lógica fuzzy no processo de inferência, sendo chamado na literatura de Sistemas Fuzzy (SF) [5]. Entre os processos de inferência existentes, os mais utilizados são o Mamdani e o Takagi-Sugeno, no quais, diferenciam apenas na etapa final do processo de inferência. Ambas as técnicas são utilizadas em diversas aplicações [10,2]. Em outra via têm crescido o interesse da academia e da industria no desenvolvimento de sistemas de hardware dedicados para implementar os sistemas fuzzy. O interesse aumentou devido algumas demandas como a possibilidade de sistemas de controle rápidos para aplicações emergentes como Internet Tátil e/ou aplicações voltadas para Internet das Coisas, *Internet-of-Things* (IoT), onde questões associadas a baixo consumo e miniaturização são fundamentais, e também para sistemas com elevada demanda de dados [6,4,12].

O desenvolvimento de sistemas de hardware dedicados podem tanto acelerar a técnica, levando considerações de paralelização, como também permitem realizar mais operações com menos pulsos de *clocks*, permitindo trabalhar em baixa potência. Os trabalhos apesentados em [9,13,1] propõem implementações de SF em FPGA nos quais mostram as possibilidade associadas a acelerar o processo de inferência fuzzy que possui um alto grau de paralelização. Todavia, a maioria dos trabalhos estudados na literatura possuem como alvo o método de inferência Mamdani. Assim, este trabalho tem como objetivo desenvolver uma nova proposta de hardware para um sistema Fuzzy do tipo Takagi-Sugeno. Diferentemente a vários trabalhos apresentados, este projeto apresenta uma plataforma híbrida utilizando representação em ponto fixo e ponto flutuante, objetivando maximizar o paralelização do hardware proposto. Tomando como base todos os trabalhos descritos em [13] (em torno de 100), nenhum implementa um sistema Fuzzy Takagi-Sugeno com 16 bits, 2 duas entradas, 7 funções de pertinência para cada entrada, 49 regras e um *throughput* em torno de 10 MHz o que quantifica a importância da estratégia proposta neste trabalho.

Resultados de validação do sistema para vários parâmetros em termos de quantidade de bits da parte fracionária e inteira são apresentados e comparados com uma implementação em ponto flutuante executada um processador de usos geral. Além dos resultados de validação são apresentados resultados de síntese para diferentes configurações no número de bits, enfatizando o desempenho do sistema em relação ao seu *throughput*.

2 Descrição Geral da Proposta

A *Takagi-Sugeno - Fuzzy Inference Machine* (TS-FIM) [5], é formado por três etapas chamadas de fuzificação, operação das regras e função de saída ou defuzificação. Na fuzificação cada i-ésimo sinal de entrada $x_i(n)$ é aplicado a um conjunto de F_i funções de pertinência, cuja saída pode ser expressa como

$$f_{i,j} = \mu_{i,j}(x_i(n)) \text{ para } j = 0, \ldots, F_i, \tag{1}$$

onde $\mu_{i,j}(\cdot)$ é a j-ésima função de pertinência da i-ésima entrada. Existem vários tipos de funções de pertinência utilizadas na literatura e neste trabalho foram utilizadas funções trapezoidais e triangulares [5]. A etapa de fuzificação, para o caso do sistema proposto que possui duas entradas, $x_0(n)$ e $x_1(n)$, gera um conjunto de $F_0 + F_1$ sinais fuzificados ($f_{0,j}$ e $f_{1,j}$) e estes sinais são processados por um conjunto de F_0F_1 regras na etapa de operação. O sinal gerado de cada g-ésima regra pode ser expresso como

$$o_g = \min(f_{0,l}, f_{1,k}) \text{ para } g = 0, \ldots, F_0F_1 - 1, \tag{2}$$

onde $g = F_0l + k$ para $(l, k) = (0, 0), (0, 1), \ldots, (F_0 - 1, F_1 - 2), (F_0 - 1, F_1 - 1)$. Finalmente a saída (defuzificação) da TS-FIM, chamada aqui de $v(n)$, pode ser expressa como

$$v(n) = \frac{\sum_{g=1}^{F_0F_1-1} o_g \times (A_g x_0(n) + B_g x_1(n) + C_g,)}{\sum_{g=0}^{F_0F_1-1} o_g}, \tag{3}$$

onde A_g, B_g e C_g são parâmetros definidos durante o projeto [5]. Assim pode-se dizer que a cada n-ésimo instante TS-FIM recebe como entrada $x_0(n)$ e $x_1(n)$ e gera como saída $v(n)$, ou seja,

$$v(n) = TSFIM\left(x_0(n), x_1(n)\right), \tag{4}$$

onde $TSFIM\left(\cdot\right)$ é uma função que representa a TS-FIM.

A Figura 1 apresenta a estrutura geral do hardware proposto no qual é formado por três módulos principais chamados aqui de *Membership Function Module* (MFM), *Operation Module* (OM) e *Output Function Module* (OFM). O hardware foi desenvolvido em sua maior parte utilizando um formato em ponto fixo para as variáveis, em que para uma dada variável qualquer, $x(n)$, no n-ésimo instante, a notações x[uV.N](n) e x[sV.N](n) indicam que a variável é formada por V bits dos quais N bits são destinados a parte fracionária e os símbolos "s"e "u"indicam que a variável possui ou não sinal, respectivamente. Para o caso das variáveis com sinal, tipo s, o número de bits destinados a parte inteira será de $\mathrm{V}-\mathrm{N}-1$ e para variáveis sem signal, tipo u, tem-se $\mathrm{V}-\mathrm{N}$ para a parte inteira.

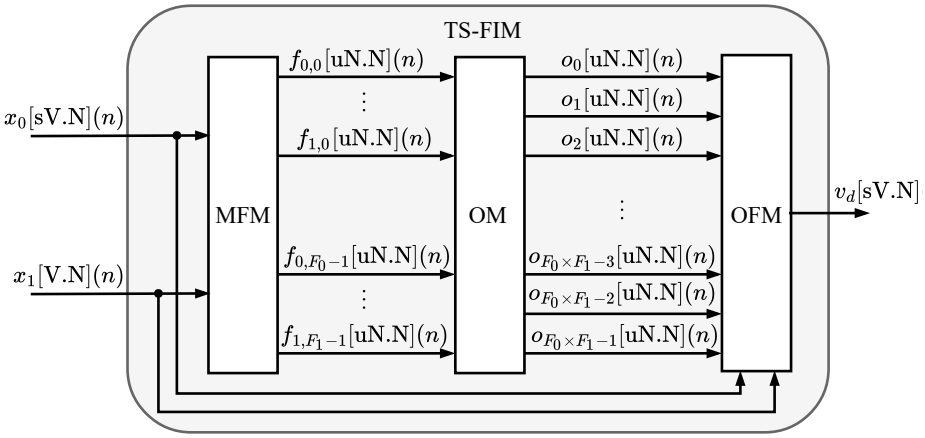

Figura 1. Visão Geral da Arquitetura Proposta.

3 *Membership Function Module* (MFM)

O MFM é o primeiro módulo associado a TS-FIM. O MFM corresponde ao processo de fuzificação apresentado em [5]. O sistema foi desenvolvido para trabalhar com duas variáveis de entrada, x_0[sV.N](n) e x_1[sV.N](n), e cada i-ésima variável é associada um módulo que agrupa F_i funções de pertinência, chamado aqui de *Membership Function Group* (MFG). As variáveis de entrada foram projetas para trabalhar normalizadas entre -1 e 1, ou seja, $\mathrm{V}=\mathrm{N}+1$. A Figura 3

apresenta o MFG-i associado a i-ésima entrada $x_i[\text{sV.N}](n)$ onde cada módulo MF-ij implementa a j-ésima função de pertinência associada a i-ésima entrada, $\mu_{i,j}(x_i(n))$, em cada n-ésimo instante. Em cada n-ésimo instante todas as F_0+F_1 funções de impertinência são executadas de forma paralela e na saída de cada MF-ij é gerado um sinal de N bits do tipo u e sem a parte inteira, chamado de $f_{i,j}[\text{uN.N}](n)$. Para ambas variáveis, $x_0[\text{sV.N}](n)$ e $x_1[\text{sV.N}](n)$, foram desenvolvidas sete funções de pertinência (tipo trapezoidal nos extremos e triangular no restante). A Figura 2 apresenta as funções de pertinência implementadas no MFM. Os termos linguísticos associados as funções de pertinência são Grande Negativo (GN), Médio Negativo (MN), Pequeno Negativo (PN), Zero (ZZ), Pequeno Positivo (PP), Médio Positivo (MP) e Grande Positivo (GP).

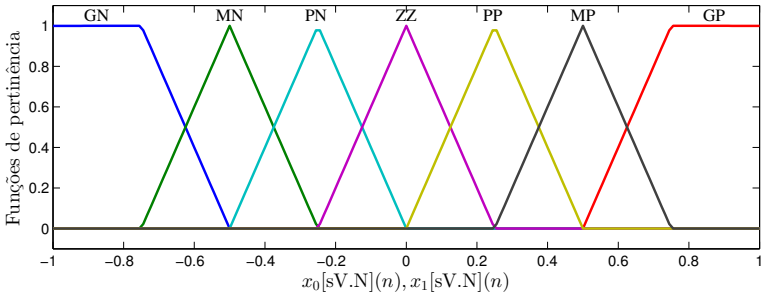

Figura 2. Funções de pertinência das variável de entrada $x_0[\text{sV.N}](n)$ e $x_1[\text{sV.N}](n)$.

Cada j-ésima função de pertinência associada a i-ésima entrada foi implementada diretamente em hardware baseadas nas seguintes expressões

$$\mu_{i,j}^{RT}(x_i[\text{sV.N}](n)) = \begin{cases} 0 & \text{se } x_i[\text{sV.N}](n) > d_{i,j}[\text{sW.T}] \\ G_{i,j}^{RT}(n) & \text{se } c_{i,j}[\text{sW.T}] \leq x_i[\text{sV.N}](n) \leq d_{i,j}[\text{sW.T}], \\ 1 & \text{se } x_i[\text{sV.N}](n) < c_{i,j}[\text{sW.T}] \end{cases} \quad (5)$$

onde $\mu_{i,j}^{RT}(\cdot)$ é a função trapezoidal da direita, $c_{i,j}[\text{sW.T}$ e $d_{i,j}[\text{sW.T}]$ são constantes ($c_{i,j}[\text{sW.T} < d_{i,j}[\text{sW.T}]$) e

$$G_{ij}^{RT}(n) = \frac{d_{i,j}[\text{W.T}] - x_i[\text{sV.N}](n)}{d_{i,j}[\text{W.T}] - c_{i,j}[\text{W.T}]}, \quad (6)$$

onde W e T são os números da bits da parte inteira e fracionária relativos as constantes da j-ésima função de ativação associada a i-ésima entrada. Para a trapezoidal da esquerda tem-se

$$\mu_{i,j}^{LT}(x_i[\text{sV.N}](n)) = \begin{cases} 0 & \text{se } x_i[\text{sV.N}](n) < e_{i,j}[\text{sW.T}] \\ G_{i,j}^{LT}(n) & \text{se } e_{i,j}[\text{sW.T}] \leq x_i[\text{sV.N}](n) \leq f_{i,j}[\text{sW.T}], \\ 1 & \text{se } x_i[\text{sV.N}](n) > f_{i,j}[\text{sW.T}] \end{cases} \quad (7)$$

onde $\mu_{i,j}^{LT}(\cdot)$ é a função trapezoidal da esquerda, $e_{i,j}$[sW.T e $f_{i,j}$[sW.T] são constantes ($e_{i,j}$[sW.T $< f_{i,j}$[sW.T]) e

$$G_{ij}^{LT}(n) = \frac{x_i[\text{sV.N}](n) - e_{i,j}[\text{W.T}]}{f_{i,j}[\text{W.T}] - e_{i,j}[\text{W.T}]}. \tag{8}$$

Finalmente, para a função de pertinência triangular tem-se

$$\mu_{i,j}^{T}(x_i[\text{sV.N}](n)) = \begin{cases} \mu_{i,j}^{LT}(x_i[\text{sV.N}](n)) & \text{se } x_i[\text{sV.N}](n) < m_{i,j}[\text{sW.T}] \\ \mu_{i,j}^{RT}(x_i[\text{sV.N}](n)) & \text{se } x_i[\text{sV.N}](n) \geq m_{i,j}[\text{sW.T}] \end{cases}, \tag{9}$$

onde $m_{i,j}$[sW.T] é o ponto central triângulo, ou seja, $m_{i,j}[\text{sW.T}] = c_{i,j}[\text{sW.T}] = f_{i,j}[\text{sW.T}]$. Os valores de W e T irão definir a resolução das funções de ativação. Na implementação proposta neste trabalho, o valor de W é sempre expresso como $\text{W} = 2 \times \text{T} + 1$. A utilização de funções de pertinência não lineares pode ser realizada com a aplicação de *Lookup Tables* (LUTs) na implementação.

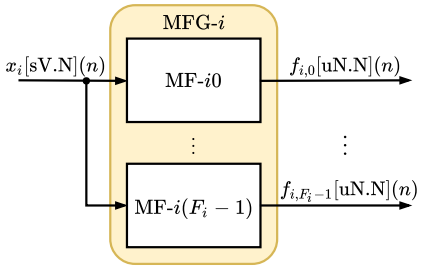

Figura 3. Arquitetura do módulo MFG-i associado a i-ésima entrada, x_i[sV.N](n).

Apesar desta implementação utilizar apenas duas entradas (x_0[sV.N](n) e x_1[sV.N](n)) e sete funções de pertinência para cada entrada, isto pode ser facilmente estendido para mais entradas e funções, dado que toda implementação é realizada de forma paralela.

4 *Operation Module* (OM)

As $F_0 + F_1$ saídas do módulo MFM são passadas para o módulo OM que executa em paralelo todas as operações relativas as F_0F_1 regras, como descrito na Seção 2. A Figura 4 detalha a estrutura em hardware de um dos F_0F_1 módulos de operação, chamado aqui de O-lk, que realiza a operação de mínimo (conector "E") entre o sinal fuzificado da l-ésima função de pertinência da entrada 0, $f_{0,l}$[uN.N](n), com a k-ésima função de pertinência da entrada 1, $f_{1,k}$[uN.N](n) (ver Equação 3).

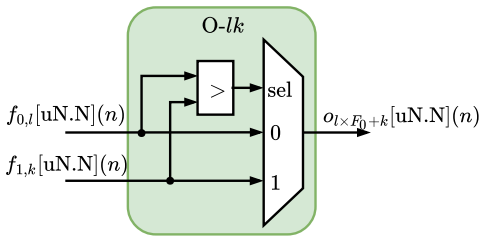

Figura 4. Arquitetura do módulo O-lk associado operação entre o sinal fuzificado da l-ésima função de pertinência da entrada 0, $f_{0,l}$[uN.N](n), com a k-ésima função de pertinência da entrada 1, $f_{1,k}$[uN.N](n) (ver Equação 3).

5 *Output Function Module* (OFM)

Finalmente, o OFM realiza a geração da variável de saída do TS-FIM durante a etapa chamada de defuzificação. Esta etapa corresponde essencialmente a implementação da Equação 3 apresentada na Seção 2. Os blocos chamados de NM e DM realizam as operações do numerador e denominador apresentadas na Equação 3, respectivamente. Para o cálculo da divisão, os sinais de saída, em ponto fixo, dos módulos NM e DM (a[sP.N](n) e b[sQ.N](n)) são transformados para padrão de ponto-flutuante (IEEE754) de 32 bits pelo módulo *Fixed-point to Float* (FP2F) ($\tilde{a}$[Float32](n) e $\tilde{b}$[Float32](n)) e após a divisão a saída do TS-FIM é convertida novamente em ponto-fixo pelo módulo *Float to Fixed-point* (F2FP). Dado que $-1 < A_g < 1$, $-1 < B_g < 1$ e $-1 < C_g < 1$ para $g = 0, \ldots, F_0F_1$ e os somatórios são realizados em árvore binária então

$$P = N + \log_2(\lceil F_0F_1 \rceil) + 3 \tag{10}$$

e

$$Q = N + \log_2(\lceil F_0F_1 \rceil) + 1. \tag{11}$$

Como as entradas do TS-FIM e o valores de A_g, B_g e C_t estão entre -1 e 1, pode-se garantir, a partir da Equação 3, que a saída, v[sV.N](n), continue normalizada entre -1 e 1. Desse modo, pode-se utilizar a mesma resolução da entrada, ou seja, N para parte fracionária e $V = N + 1$ para parte inteira, como apresentado na Figura 5.

6 Resultados de Validação e Síntese

6.1 Síntese

A Tabela 1 apresenta os resultados de síntese relacionados a ocupação em hardware e o *throughput* ($R_s = 1/t_s$) máximo do sistema para vários valores de N e T. As colunas, NR, NLUT e NMULT representam o número de registradores, células lógicas utilizadas como LUTs e multiplicadores no hardware implementado

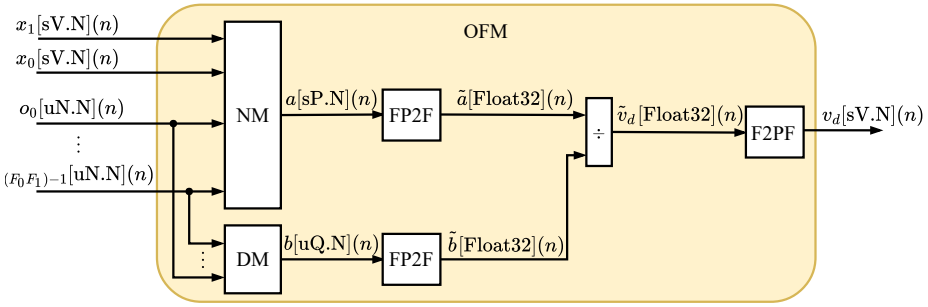

Figura 5. Arquitetura do módulo OFM.

no FPGA, respectivamente. Já as colunas PNR, PNLUT e NMULT representam a porcentagem em relação ao total de recursos do FPGA. Para o FPGA (Xilinx Virtex 6 xc6vlx240t-1ff1156) utilizado existem 301.440 registradores, 150.720 células lógicas para serem utilizadas como LUTs e 768 multiplicadores.

Os resultados de síntese, baseados na Tabela 1, mostram que a implementação proposta ocupa um espaço pequeno em hardware menos de 1%, PR, em registradores e menos de 7,5% em LUTs, PLUT, do FPGA. Estes dados viabilizam a utilização de várias TS-FIM implementadas em paralelo no FPGA, permitindo acelerar várias aplicações em ambientes de dados massivos. Por outro lado, o baixo consumo de hardware permite o uso do TS-FIM em FPGAs pequenas de baixo custo e consumo para aplicações de IoT e M2M. Outro ponto importante a ser analisado, ainda em relação a síntese, é o comportamento linear do consumo de hardware em relação ao número de bits, diferentemente do trabalho apresentado em [8,11], e isto é importante, pois viabiliza a utilização de sistemas com uma maior resolução.

Em relação aos valores de *throughput*, R_s, os resultados obtidos foram bastante relevantes, com valores entre 12.54 Msps (*samples per second*) para o caso (N = 8 e T = 4) e 11.28 Msps para o caso (N = 16 e T = 10) o que viabiliza sua aplicação em vários problemas com grande volume dados para processamento como apresentado em [1] ou em problemas com requisitos de controle rápidos como o caso das aplicações em internet tátil [7]. Observa-se também que o *throughput* possui um comportamento linear em função do número de bits.

As Figuras 6 e 7 mostram as superfícies do comportamento do número de LUTs (NLUT) e do *throughput* em função de N e T, respectivamente. Para ambos os casos foi realizado um ajuste, através de um técnica de regressão, para encontrar o plano que se adéqua mais aos pontos medidos. Para o caso do NLUT foi encontrado o plano, $f_{\mathrm{NLUT}}\,(\mathrm{N},\mathrm{T})$ expresso por

$$f_{\mathrm{NLUT}}\,(\mathrm{N},\mathrm{T}) \approx 1.682 + 532{,}2 \times \mathrm{N} + 6{,}493 \times 10^{-13} \times \mathrm{T} \qquad (12)$$

Tabela 1. Resultados relativos a ocupação, taxa de amostragem e *throughput*.

N	T	NR	PR	NLUT	PLUT	NMULT	PNMULT	t_s (ns)	R_s (Msps)	MSE
8	4	217	$< 1\%$	6.339	$< 4{,}54\%$	49	6,38%	79,72	12,54	2.395×10^{-6}
	6			6.381				80,95	12,35	
	8			6.452				81,96	12,20	
	10			6.598				83,76	11,94	
10	4	259	$< 1\%$	6.772	$< 4{,}87\%$	49	6,38%	84,18	11,88	1.274×10^{-7}
	6			6.904				82,70	12,09	
	8			7.331				83,94	11,91	
	10			7.331				83,00	12,05	
12	4	324	$< 1\%$	7.280	$< 5{,}59\%$	49	6,38%	82,65	12,10	7.181×10^{-9}
	6			7.916				83,28	12,01	
	8			7.954				87,02	11,49	
	10			8.147				85,99	11,63	
14	4	384	$< 1\%$	8.761	$<6{,}29\%$	49	6,38%	84,12	11,89	4.829×10^{-10}
	6			8.915				85,08	11,75	
	8			8.999				86,39	11,58	
	10			9.163				86,75	11,53	
16	4	428	$< 1\%$	9.816	$< 7{,}05\%$	49	6,38%	86,42	11,54	2.644×10^{-11}
	6			9.990				84,80	11,79	
	8			10.072				88,31	11,32	
	10			10.252				88,65	11,28	

com um $R^2 = 0{,}9766$. Já para o *throughput* em Msps foi encontrado o plano, $f_{R_s}(\mathrm{N}, \mathrm{T})$, caracterizado como

$$f_{R_s}(\mathrm{N}, \mathrm{T}) \approx 13{,}24 - 0{,}1163 \times \mathrm{N} + 3{,}414 \times 10^{-16} \times \mathrm{T} \tag{13}$$

com um $R^2 = 0{,}7521$.

6.2 Validação

As Figuras 9 e 8 exibem o mapeamento entre entrada ($x_0(n)$ e $x_1(n)$) e saída $v(n)$ para hardware proposto e uma uma implementação de referência com *Fuzzy Matlab Toolbox* (*License number* 1080073) [3], respectivamente. A implementação do Matlab, apresentada na Figura 8, utiliza formato em ponto flutuante com 64 bits (formato *double*) enquanto que na Figura 9 é apresentada o mapeamento gerado pelo hardware proposto utilizando a menor resolução sintetizada (N = 8, V = 9 e T = 4). Estas figuras conseguem apresentar uma representação qualitativa da implementação proposta, no qual os resultados obtidos são bastante semelhantes do esperado.

A última coluna da Tabela 1 (MSE) mostra o erro quadrático médio (*Mean Square Error*) entre do *Fuzzy Matlab Toolbox* e a implementação proposta em hardware para vários casos N e T. Para o experimento, o cálculo do MSE é

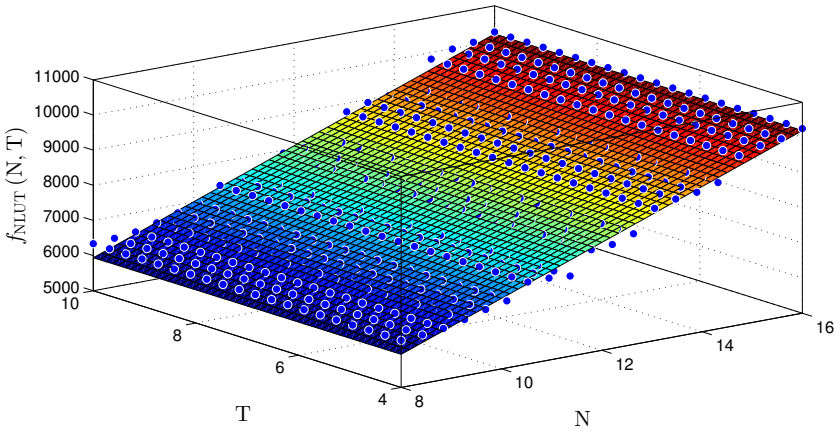

Figura 6. Plano, $f_{\mathrm{NLUT}}\,(\mathrm{N},\mathrm{T})$, encontrado para estimar o número de LUTs em função do número de bits N e T.

expresso como

$$MSE = \frac{1}{Z}\sum_{n=0}^{Z-1}\left(v_{ref}[\mathrm{Float64}](n) - v[\mathrm{sV.N}](n)\right)^2, \tag{14}$$

onde Z representa o número de pontos testados que corresponderam a 10.000 pontos espalhados de forma uniforme dentro dos limites dos valores de entrada (-1 e 1). As Figuras 9 e 8 foram geradas com estes pontos. Os resultados obtidos em relação ao MSE foram também bastante significativos, mostrando que o TS-FIM possui uma resposta bastante similar a implementação com 64 bits mesmo para uma resolução em ponto fixo de 8 bits ($MSE = 2.395 \times 10^{-6}$). Outro dado interessante foi relativo aos valores de T que não influenciaram significativamente no valor MSE para as funções de pertinências utilizadas (ver 2) no projeto. É importante salientar que a implementação do TS-FIM com poucos bits acarreta em hardwares menores, com baixo consumo ou com valores de *throughput* altos.

7 Conclusões

Este trabalho teve como objetivo o desenvolvimento de um hardware para uma máquina de inferência fuzzy do tipo Takagi-Sugeno em um FPGA. O hardware desenvolvido utilizou uma implementação paralela utilizando um esquema híbrido com representação em ponto fixo e ponto flutuante em partes distintas do esquema proposto. Todos os detalhes da implementação foram apresentados bem como resultados relativos a síntese e a simulações em precisão de bit. Os resultados de síntese foram realizados para várias resoluções de tamanho de bit

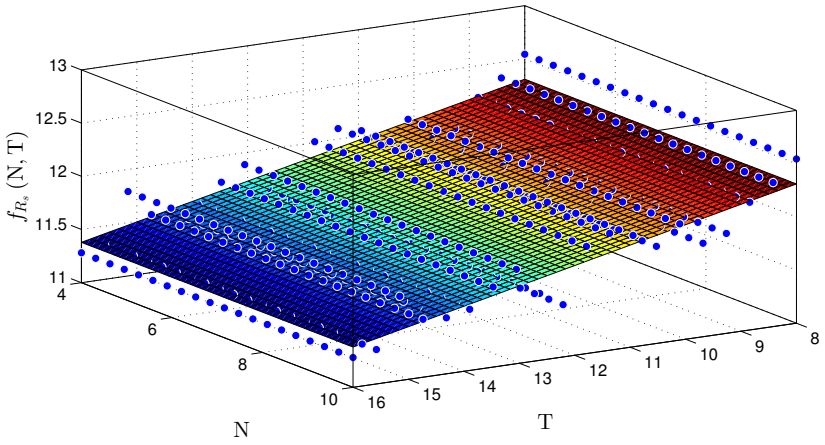

Figura 7. Plano, f_{R_s} (N, T), encontrado para estimar o *throughput*, R_s, em função do número de bits N e T.

e mostraram que o hardware proposto e viável, podendo ser utilizado em aplicações com requisitos criticos de tempo de processamento. Através dos dados de síntese, foram geradas curvas para predição do consumo de hardware e do *throughput* para valores de bits não testados, objetivando caracterizar o hardware proposto. Além, dos dados de síntese foram também realizados simulações com precisão de bit para vários valores de entrada e os resultados foram comparados uma implementação de referência, validando o hardware proposto.

Referências

1. Chowdhury, S.R., Saha, H.: A high-performance fpga-based fuzzy processor architecture for medical diagnosis. IEEE Micro 28(5), 38–52 (Sept 2008)
2. Ding, B., Luo, X., Wei, S.: A survey on stability research of discrete-time takagi-sugeno fuzzy control systems. In: IEEE ICCA 2010. pp. 411–416 (June 2010)
3. MATLAB: Matlab Fuzzy Logic Toolbox User's Guide - R2016a. The MathWorks Inc., Natick, Massachusetts (2012)
4. Nasrollahzadeh, A., Karimian, G., Mehrafsa, A.: Implementation of neuro-fuzzy system with modified high performance genetic algorithm on embedded systems. Applied Soft Computing (2017), `http://www.sciencedirect.com/science/article/pii/S156849461730409X`
5. Oviedo, J., Vandewalle, J., Wertz, V.: Fuzzy Logic, Identification and Predictive Control. Advances in Industrial Control, Springer London (2004)
6. Poli, V.S.R.: Fuzzy data mining and web intelligence. In: Fuzzy Theory and Its Applications (iFUZZY), 2015 International Conference on. pp. 74–79. IEEE (2015)
7. Simsek, M., Aijaz, A., Dohler, M., Sachs, J., Fettweis, G.: The 5g-enabled tactile internet: Applications, requirements, and architecture. In: 2016 IEEE Wireless Communications and Networking Conference. pp. 1–6 (April 2016)

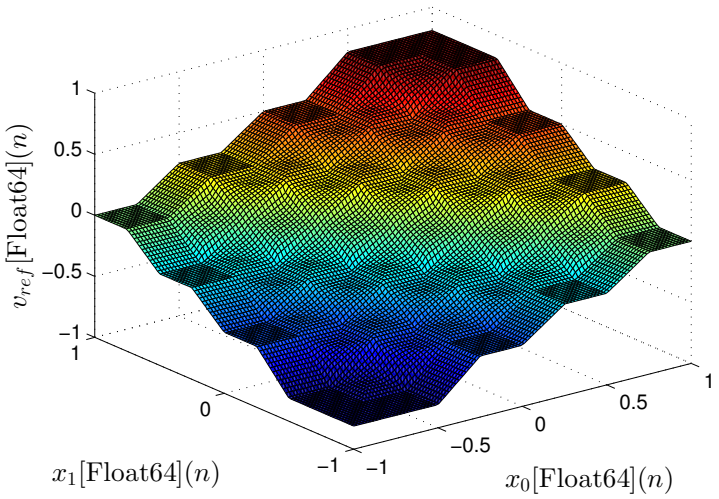

Figura 8. Mapeamento entrada e saída do TS-FIM gerado pelo *Matlab Fuzzy Logic Toolbox* utilizando o formato double.

8. de Souza, A.C., Fernandes, M.A.: Parallel fixed point implementation of a radial basis function network in an fpga. Sensors 14(10), 18223–18243 (2014)
9. Sun, Y., Tang, S., Meng, Z., Zhao, Y., Yang, Y.: A scalable accuracy fuzzy logic controller on {FPGA}. Expert Systems with Applications 42(19), 6658 – 6673 (2015)
10. Takagi, T., Sugeno, M.: Fuzzy identification of systems and its applications to modeling and control. IEEE Transactions on Systems, Man, and Cybernetics SMC-15(1), 116–132 (Jan 1985)
11. Torquato, M.F., Fernandes, M.A.C.: Proposta de implementação paralela de algoritmo genético em fpga. In: XXI Congresso Brasileiro de Automática (2016)
12. Yaqoob, I., Hashem, I.A.T., Gani, A., Mokhtar, S., Ahmed, E., Anuar, N.B., Vasilakos, A.V.: Big data: From beginning to future. International Journal of Information Management 36(6), 1231 – 1247 (2016), http://www.sciencedirect.com/science/article/pii/S0268401216304753
13. Zavala, A.H., Nieto, O.C.: Fuzzy hardware: A retrospective and analysis. IEEE Transactions on Fuzzy Systems 20(4), 623–635 (Aug 2012)

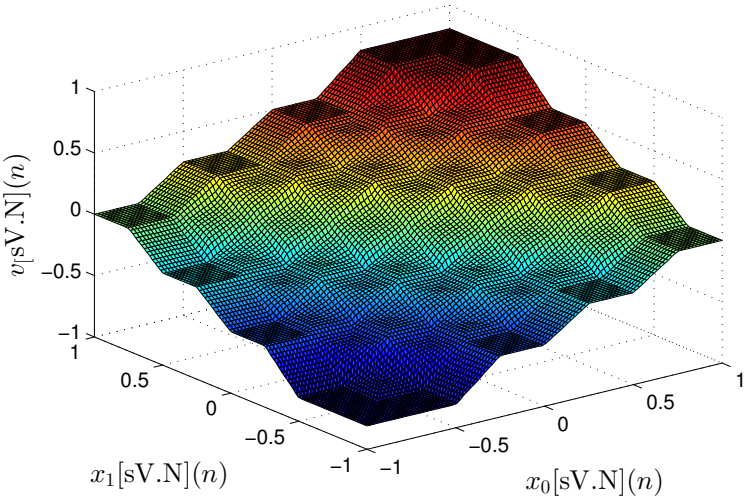

Figura 9. Mapeamento entrada e saída do TS-FIM gerado pelo hardware proposto utilizando o formato em ponto fixo N = 8, V = 9 e T = 4.