PROPOSTA DE UM HARDWARE PARA O SISTEMA FUZZY-PI TAKAGI-SUGENO EM FPGA

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

*Departamento de Computação e Automação (DCA)
Universidade Federal do Rio Grande do Norte (UFRN)
Natal, RN, Brasil

Email: edumorais92@gmail.com, sergionatan@dca.ufrn.br, mfernandes@dca.ufrn.br

Abstract— This paper aims to present the development of dedicated hardware for a Fuzzy-PI control system in a Field Programmable Gate Array (FPGA). The proposed system uses a Takagi-Sugeno inference machine consisting of two inputs and one output. In order to validate the proposed hardware, the paper presents results through bit accuracy in simulation for a dynamic system, level control in a tank. In addition to the validation results are presented information about the occupation and throughput of the proposed hardware. The hardware was implemented in a Xilinx Virtex 6 FPGA xc6vlx240t-1ff1156.

Keywords— Fuzzy-PI, Takagi-Sugeno, FPGA, Hardware, Fuzzy Control.

Resumo— Este trabalho tem como objetivo apresentar o desenvolvimento de um hardware dedicado para um sistema de controle Fuzzy-PI em um *Field Programmable Gate Array* (FPGA). O sistema proposto utiliza uma máquina de inferência Takagi-Sugeno formada por duas entradas e um saída. Objetivando validar o hardware proposto, o trabalho apresenta resultados através de simulação em precisão de bit para um sistema dinâmico, de controle de nível em um tanque. Além dos resultados de validação são apresentadas informações sobre a ocupação e *throughput* do hardware proposto. O hardware foi implementado em um FPGA Xilinx Virtex 6 xc6vlx240t-1ff1156.

Palavras-chave— Fuzzy-PI, Takagi-Sugeno, FPGA, Hardware, Controle Fuzzy.

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 outros. 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) (Oviedo et al., 2004). 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 técnicas são aplicadas em diversas aplicações. Todavia, as propostas baseadas em Takagi-Sugeno têm focado em sua maioria em aplicações nas áreas de automação e controle (Takagi and Sugeno, 1985; Ding et al., 2010). Em outra via tem crescido o interesse da academia e da indústria 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 (Simsek et al., 2016) 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 quantidade de dados, como as áreas de *Big Data* e *Mining of Massive Datasets* (MMD)(Poli, 2015; Nasrollahzadeh et al., 2017; Yaqoob et al., 2016).

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 que esses sistemas em hardware possam trabalhar em baixa potência. Os trabalhos apesentados em (Sun et al., 2015; Zavala and Nieto, 2012; Chowdhury and Saha, 2008) propõem implementações de SF em FPGA nos quais mostram as possibilidades associadas a acelerar o processo de inferência fuzzy que possuam 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 controlador Fuzzy-PI 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 a paralelização do hardware proposto. Tomando como base todos os trabalhos descritos em (Zavala and Nieto, 2012) (em torno de 100), nenhum implementa um sistema Fuzzy-PI 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

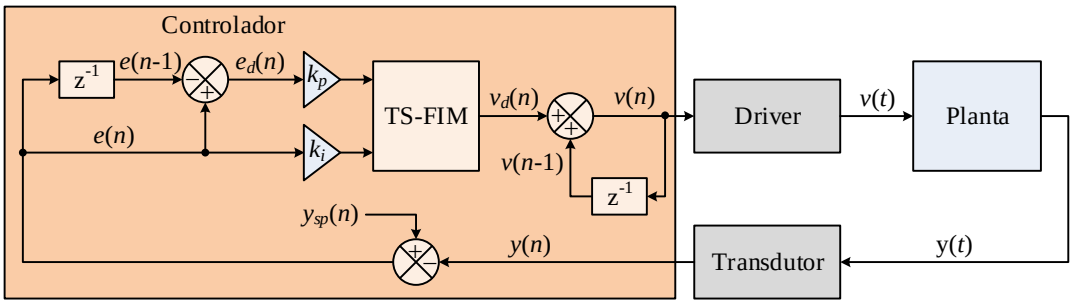

Figura 1: Arquitetura do sistema de controle Fuzzy PI Takagi-Sugeno.

comparados com uma implementação em ponto flutuante executada em um processador de uso geral. Além dos resultados de validação são apresentados resultados de síntese também para várias quantidades de bits, mostrando a performance do sistema em relação ao seu *throughput*.

2 Sistema de Controle Fuzzy-PI Takagi-Sugeno

A Figura 1 detalha o esquema de controle utilizado como referência, no qual a variável de saída da planta, $y(t)$, também chamada de variável de processo, pode assumir valores de nível, $n(t)$, velocidade angular, $\omega(t)$, e velocidade linear, $x(t)$, e outros dependendo da planta. A variável de processo, $y(t)$, passar por um transdutor que converter a medida física em um sinal elétrico proporcional que é depois discretizado, a uma taxa de amostragem, t_s, gerando o sinal, $y(n)$.

O controlador, chamado na literatura de Fuzzy-PI (Oviedo et al., 2004), atua no sinal de erro, $e(n)$, baseado na diferença entre a variável de processo, $y_(n)$, e uma variável de referência y_{sp} (também chamada de *set point*). Além do valor de $e(n)$ o controlador calcula o valor da diferença do erro que pode ser descrito como

$$e_d(n) = e(n) - e(n-1), \tag{1}$$

e envia estes sinais para a máquina de inferência fuzzy Takagi-Sugeno, chamada neste artigo de *Takagi-Sugeno - Fuzzy Inference Machine* (TS-FIM) (Oviedo et al., 2004). A TS-FIM possui como saída o valor referente a diferença do sinal de controle, $v_d(n)$, que é então integrado no tempo, gerando o sinal de controle $v(n)$ e este sinal é enviado para um driver que transforma o sinal discreto em um sinal contínuo de potência, $v(t)$, para ser aplicado a planta.

O TS-FIM é formado basicamente 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

$$f_{i,j} = \mu_{i,j}(x_i(n)) \text{ para } j = 0, \ldots, F_i, \tag{2}$$

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 (Oviedo et al., 2004). A etapa de fuzificação, para o caso do controlador proposto com 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. Cada g-ésima regra pode ser expressa como

$$r_g = \min(f_{0,l}, f_{1,k}) parag = 0, \ldots, F_0F_1 - 1, \tag{3}$$

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 função de saída (defuzificação) da TS-FIM pode ser expressa como

$$v_d(n) = \frac{\sum_{g=1}^{F_0F_1-1} r_g \left(A_g x_0(n) + B_g x_1(n) + C_g\right)}{\sum_{g=0}^{F_0F_1-1} r_g}, \tag{4}$$

onde A_g, B_g e C_g são parâmetros definidos durante o projeto (Oviedo et al., 2004).

3 Descrição da Implementação Proposta

A Figura 2 apresenta a estrutura geral do hardware proposto, baseado na estrutura apresentada na Figura 1, no qual é formado por cinco módulos principais chamados aqui de *Input Processing Module* (IM), *Membership Function Module* (MFM), *Operation Module* (OM), *Output Function Module* (OFM) e *Integration Module* (IM). Os módulos MFM, OM e OFM fazem parte da TS-FIM. O hardware foi desenvolvido em sua maior parte utilizando um formato em ponto fixo para as variáveis, no qual, para uma dada variável qualquer, $x(n)$, no n-ésimo instante, a notações x[uT.W](n) e x[sT.W](n) indicam que a variável é formada por T bits dos quais W bits são destinados a parte fracionária e os símbolos "s"e "u"indicam que a variável possui ou não sinal, respectivamente. Para

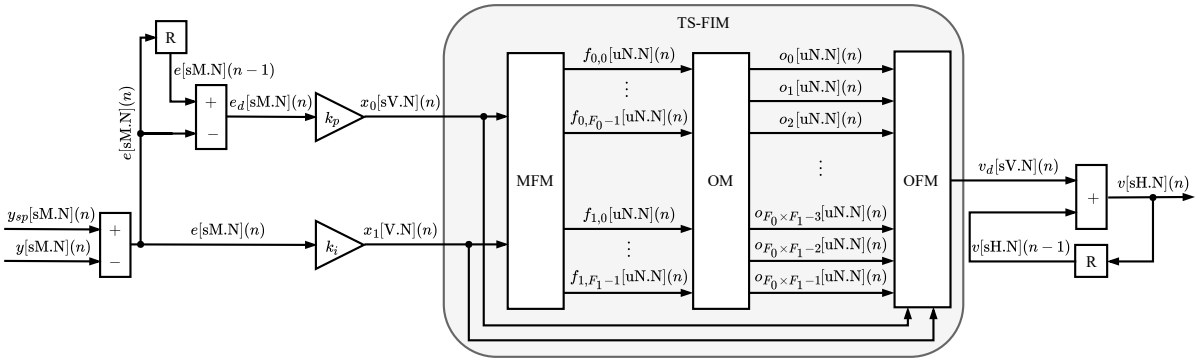

Figura 2: Visão Geral da Arquitetura Proposta.

o caso das variáveis com sinal, tipo s, o número de bits destinados a parte inteira será de $T-1$.

O módulo IM (ilustrado na Figura 2) é responsável pelo processamento do sinal de controle gerado pela planta para a entrada do Takagi-Sugeno Fuzzy PI. A maioria dos sinais associados a este módulo foram implementados com M bits onde um é reservado para o sinal e N para a parte fracionária onde, o valor de M pode ser expresso como

$$M = N + \log_2(\lceil y_{max} \rceil) + 1, \tag{5}$$

onde y_{max} representa o maior valor em módulo da variável de processo, $y(n)$. Os valores de k_p e k_i devem ser projetados a tentar manter os sinais de saída do módulo, x_0[V.N](n) e x_1[V.N](n), entre -1 e 1, respectivamente. Desta forma, pode-se configurar o V = N + 1, objetivando a redução do número de bits associado ao projeto. É importante observar que os dois módulos de ganho, k_p e k_i, também saturam o sinal em [V.N] bits após a multiplicação.

Já o módulo MFM é o primeiro módulo associado a TS-FIM e ele corresponde ao processo de fuzificação apresentado na Seção 2. 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).

A Figura 3 apresenta o MFG-i associado a i-ésima entrada x_i[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 pertinência 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 $f_{i,j}$[uN.N](n).

Já a Figura 4 apresenta as funções de pertinência implementadas no MFM. Para ambas variáveis, x_0[sV.N](n) e x_1[sV.N](n), foram criadas sete funções de pertinência (tipo trapezoidal nos extremos e triangular no restante). Os termos linguísticos associadas as funções de pertinência são

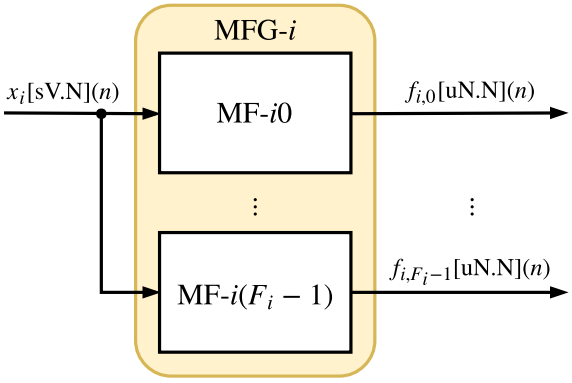

Figura 3: Arquitetura do módulo MFG-i associado a i-ésima entrada, x_i[sV.N](n).

Grande Negativo (GN), Médio Negativo (MN), Pequeno Negativo (PN), Zero (ZZ), Pequeno Positivo (PP), Médio Positivo (MP) e Grande Positivo (GP).

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 5 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).

Finalmente, o OFM, ilustrado na Figura 6, realiza a geração da variável de saída do TS-FIM durante a etapa chamada de defuzificação. Esta etapa corresponde essencialmente a implementação da Equação 4 apresentada na Seção 2. Os blocos chamados de NM e DM realizam as operações do numerador e denominador apresentadas na Equação 4, 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 no padrão de ponto-flutuante (IEEE754) de 32 bits pelo módulo *Fixed-point to Float* (FP2F) e após a divisão a saída do TS-

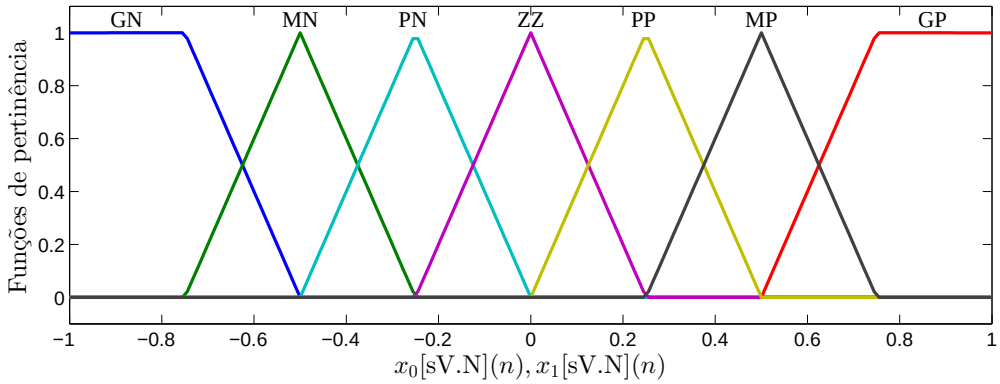

Figura 4: Funções de pertinência das variável de entrada x_0[sV.N](n) e x_1[sV.N](n).

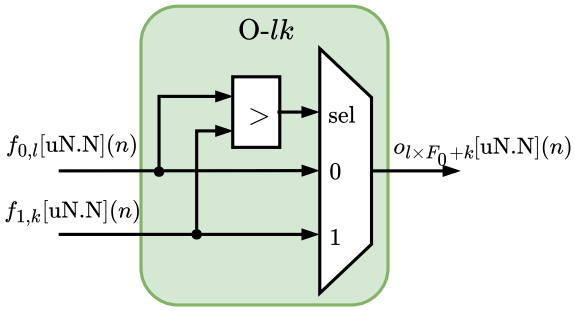

Figura 5: Arquitetura do módulo O-lk associado a operação entre o sinal fuzificado da l-ésima função de pertinência da entrada 0, $f_{0,l}$[uN.N](n), com a k-ésima função de pertinência da entrada 1, $f_{1,k}$[uN.N](n).

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

$$P = N + \log_2(\lceil F_0F_1 \rceil) + 3 \tag{6}$$

e

$$Q = N + \log_2(\lceil F_0F_1 \rceil) + 1. \tag{7}$$

4 Resultados de Validação e Síntese

4.1 Síntese do Hardware Fuzzy-PI

A Tabela 1 apresenta os resultados de síntese relacionados a ocupação em hardware e o *throughput*, R_s, máximo do sistema para vários valores de N. O *throughput* é expresso em milhões amostras por segundo (*mega samples per second* - Msps) e é calculado como

$$R_s = \frac{1}{t_s}. \tag{8}$$

As colunas, NR, NLUT e NMULT representam o número de registradores, células lógicas utilizadas como LUTs e multiplicadores no hardware implementado 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% em LUTs, PLUT, do FPGA. Além desses dados, é possível visualizar o números de multiplicadores embarcados, PNMULT, que se manteve abaixo dos 7%. Estes dados viabilizam a utilização de várias TS-FIM implementadas em paralelo no FPGA, permitindo acelerar várias aplicações em ambientes de dados massivos do tipo MMD por outro lado a baixa ocupação 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 (de Souza and Fernandes, 2014; Torquato and Fernandes, 2016), e isto é importante, pois viabiliza a utilização de sistemas com uma maior resolução.

Em relação aos dados de *throughput*, R_s, os resultados obtidos foram bastante relevantes, com valores entre $10,77$ Msps (*samples per second*) para o caso (N = 8 e T = 4) e $9,59$ Msps para o caso (N = 16 e T = 10) o que viabiliza sua aplicação em vários problemas com grande volume de dados para processamento como apresentado em (Chowdhury and Saha, 2008) ou em problemas com requisitos de controle rápidos como o caso das aplicações em internet tátil (Simsek et al., 2016). Observa-se também que o *throughput* possui um comportamento linear em função do número de bits.

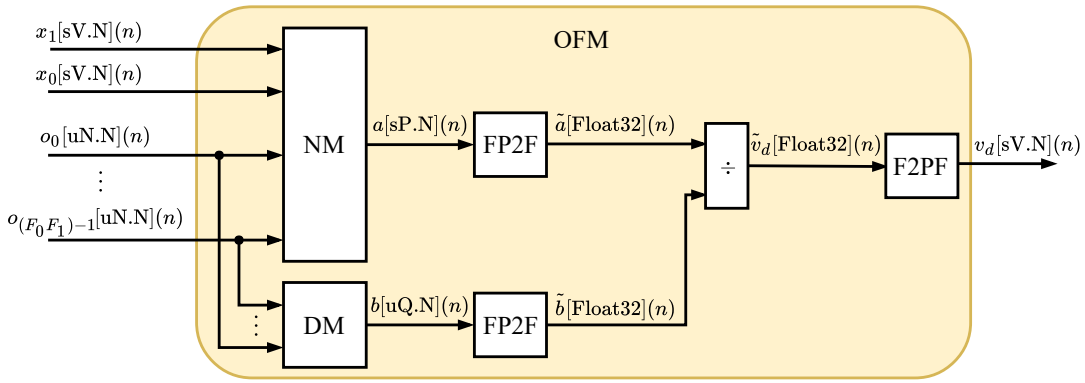

Figura 6: Arquitetura do módulo OFM.

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

N	NR	PR	NLUT	PLUT	NMULT	PNMULT	t_s (ns)	R_s (Msps)
8	261	$< 1\%$	6.834	4,53%	49	6,38%	92,87	10,77
10	307	$< 1\%$	7.331	4,86%	49	6,38%	98,44	10,16
12	375	$< 1\%$	8.409	5,58%	49	6,38%	98,68	10,13
14	438	$< 1\%$	9.460	6,28%	49	6,38%	99,98	10,00
16	488	$< 1\%$	10.595	7,03%	49	6,38%	104,31	9,59

4.2 Validação da TS-FIM

As Figuras 7 e 8 mostram o mapeamento entre entrada ($x_0(n)$ e $x_1(n)$) e saída $v(n)$ para o hardware proposto e uma implementação de referência com *Fuzzy Matlab Toolbox* (*License number* 1080073) (MATLAB, 2012), respectivamente. A implementação do Matlab, apresentada na Figura 7, utiliza formato em ponto flutuante com 64 bits (formato *double*) enquanto que na Figura 8 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 uma qualitativa da implementação proposta, no qual os resultados obtidos são bastante semelhantes do esperado.

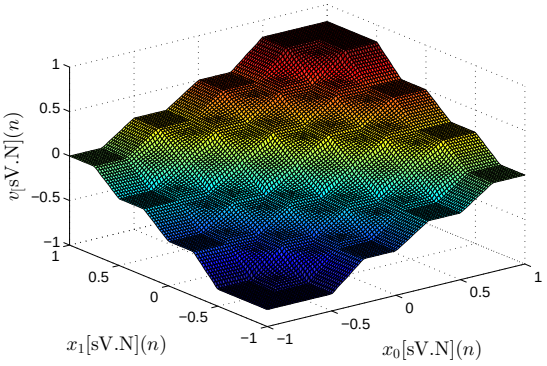

Figura 8: Mapeamento entrada e saída do TS-FIM gerado pelo hardware proposto utilizando o formato em ponto fixo com N = 8, V = 9 e T = 4.

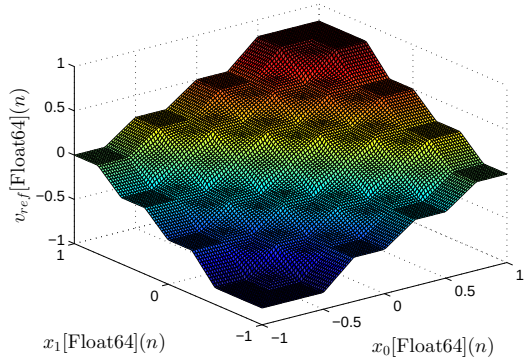

Figura 7: Mapeamento entrada e saída do TS-FIM gerado pelo *Matlab Fuzzy Logic Toolbox* utilizando o formato double.

A última coluna da Tabela 2 (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 expresso como

$$MSE = \frac{1}{Z}\sum_{n=0}^{Z-1}\left(v_{ref}[\text{Float64}](n) - v_d[\text{sV.N}](n)\right)^2 \quad (9)$$

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 7 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 fix de

8 bits ($MSE = 2.4 \times 10^{-6}$). Outro dado interessante foi relativo aos valores T que não influenciaram significativamente no valor MSE para as funções de pertinências utilizadas (ver Figura 4) 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.

Tabela 2: Resultados relativos em relação ao MSE.

N	T	MSE
8	4 6 8 10	2.4×10^{-6}
10	4 6 8 10	1.3×10^{-7}
12	4 6 8 10	7.2×10^{-9}
14	4 6 8 10	4.9×10^{-10}
16	4 6 8 10	2.7×10^{-11}

4.3 Validação Hardware Fuzzy-PI

Objetivando validar os resultados do controlador Fuzzy-PI, em hardware, foram realizados testes de simulação em precisão de bit para um sistema dinâmico não-linear caracterizado por tanque de água acoplado a uma bomba (ver Figura 9). A bomba com vazão constante, $q_m(t)$, é acoplada a uma válvula de entrada com controle contínuo, $V_e(t)$, no qual gera uma vazão de entrada no tanque de $q_e(t)$. A válvula de saída do tanque também possui um controle contínuo $V_o(t)$ e a vazão após esta válvula é dada por $q_s(t)$. A altura do nível de água do tanque é caracterizada pela variável $n(t) = y(t)$ (ver Figura 1). A planta possui duas entradas que são o controle da válvula de entrada, $V_e(t)$, e o controle da válvula de saída, $V_o(t)$, e uma saída que é o nível de água do tanque, $n(t)$.

A equação diferencial que modela o sistema pode ser expressas como

$$q_e(t) - V_o(t)q_s(t) = A\frac{dn(t)}{dt} \tag{10}$$

onde

$$q_e(t) = q_m(t)V_e(t) \tag{11}$$

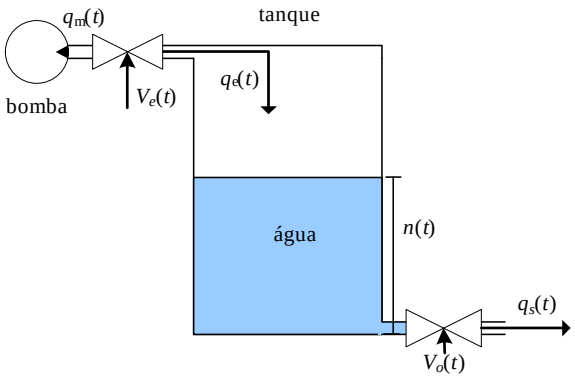

Figura 9: Planta associada ao sistema de controle de nível, $n(t)$.

e

$$q_s(t) = a\sqrt{2gn(t)}, \tag{12}$$

no qual, $q_m(t)$, A, g e a representam a vazão da bomba (m^3/s), área da seção transversal do tanque (m^2), a força gravitacional (m/s^2) e a área da seção transversal do tubo de saída do tanque (m^2), respectivamente. A Figura 10 apresenta os resultados de validação do hardware para várias resoluções em termos de número de bits da parte fracionária, $N = \{8, 10, 12, 14, 16\}$, no qual observa-se que o controlador seguiu a referência da planta em todos os casos.

Os resultados mostraram que a área de ocupação (número de registradores e células lógicas) aumenta em uma proporção alta com o número de bits utilizados na parte fracionária, N, conforme pode ser observado na Tabela 1. A cada acréscimo de dois bits em N o número de registradores aumenta em torno de 50 e o de células lógicas em torno de 1000. Por outro lado, observa-se que a redução do *throughput* em função do número de bits é pequena devido a paralelização associada a implementação em hardware do Fuzzy-PI. Outro aspecto interessante a ser observado é relativo ao funcionamento do controlador Fuzzy-PI Takagi-Sugeno que se manteve seguindo a referência mesmo para uma pequena quantidade de bits, ou seja, uma baixa resolução.

5 Conclusões

Este trabalho teve como objetivo o desenvolvimento de um hardware para um sistema de controle do tipo Fuzzy-PI baseado em uma máquina de inferência fuzzy do tipo Takagi-Sugeno em um FPGA. O hardware desenvolvido utilizou uma implementação paralela e 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

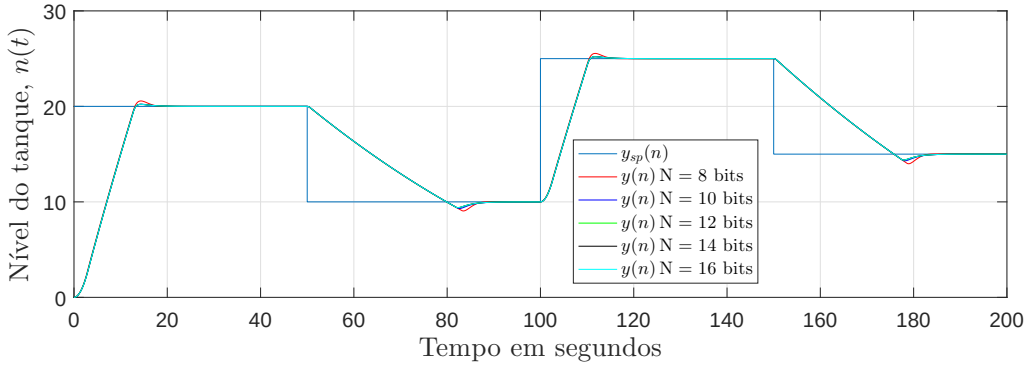

Figura 10: Resultados de validação da proposta em hardware do controlador Fuzzy-PI Takagi-Sugeno na planta de nível.

tamanho de bit e mostraram que o hardware proposto é viável podendo ser utilizado em aplicações com requisitos críticos de tempo de processamento. Além, dos dados de síntese foram também realizadas simulações com precisão de bit para vários valores de entrada e os resultados foram comparados a uma implementação de referência, validando o hardware proposto.

Referências

Chowdhury, S. R. and Saha, H. (2008). A high-performance fpga-based fuzzy processor architecture for medical diagnosis, *IEEE Micro* **28**(5): 38–52.

de Souza, A. C. and Fernandes, M. A. (2014). Parallel fixed point implementation of a radial basis function network in an fpga, *Sensors* **14**(10): 18223–18243.

Ding, B., Luo, X. and Wei, S. (2010). A survey on stability research of discrete-time takagi-sugeno fuzzy control systems, *IEEE ICCA 2010*, pp. 411–416.

MATLAB (2012). *Matlab Fuzzy Logic Toolbox User's Guide - R2016a*, The MathWorks Inc., Natick, Massachusetts.

Nasrollahzadeh, A., Karimian, G. and Mehrafsa", A. (2017). Implementation of neuro-fuzzy system with modified high performance genetic algorithm on embedded systems, *Applied Soft Computing* .

Oviedo, J., Vandewalle, J. and Wertz, V. (2004). *Fuzzy Logic, Identification and Predictive Control*, Advances in Industrial Control, Springer London.

Poli, V. S. R. (2015). Fuzzy data mining and web intelligence, *Fuzzy Theory and Its Applications (iFUZZY), 2015 International Conference on*, IEEE, pp. 74–79.

Simsek, M., Aijaz, A., Dohler, M., Sachs, J. and Fettweis, G. (2016). The 5g-enabled tactile internet: Applications, requirements, and architecture, *2016 IEEE Wireless Communications and Networking Conference*, pp. 1–6.

Sun, Y., Tang, S., Meng, Z., Zhao, Y. and Yang, Y. (2015). A scalable accuracy fuzzy logic controller on {FPGA}, *Expert Systems with Applications* **42**(19): 6658 – 6673.

Takagi, T. and Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control, *IEEE Transactions on Systems, Man, and Cybernetics* **SMC-15**(1): 116–132.

Torquato, M. F. and Fernandes, M. A. C. (2016). Proposta de implementação paralela de algoritmo genético em fpga, *XXI Congresso Brasileiro de Automática*.

Yaqoob, I., Hashem, I. A. T., Gani, A., Mokhtar, S., Ahmed, E., Anuar, N. B. and Vasilakos, A. V. (2016). Big data: From beginning to future, *International Journal of Information Management* **36**(6): 1231 – 1247.

Zavala, A. H. and Nieto, O. C. (2012). Fuzzy hardware: A retrospective and analysis, *IEEE Transactions on Fuzzy Systems* **20**(4): 623–635.