15º Congresso de Inovação, Ciência e Tecnologia do IFSP - 2024

IMPLEMENTAÇÃO DE REDE NEURAL ARTIFICIAL EM FPGA PARA ANÁLISE VIBRACIONAL DE SINAIS DE ACELERÔMETROS

Davi de Lucas França[1], Miguel A. de A de Sousa[2], Ricardo Pires[3], Sara D. dos Santos[4]

[1]Graduando em Engenharia Eletrônica, Bolsista CNPq, IFSP, Câmpus São Paulo, davi.franca@aluno.ifsp.edu.br
[2]Doutor em Ciências, Instituto Federal de Educação, Ciência e Tecnologia de São Paulo - IFSP, angelo@ifsp.edu.br
[3]Docente no Instituto Federal de Educação, Ciência e Tecnologia de São Paulo - IFSP, ricardo_pires@ifsp.edu.br
[4]Doutora em Ciências, Instituto Federal de Educação, Ciência e Tecnologia de São Paulo - IFSP, sarad@ifsp.edu.br

Área de conhecimento (Tabela CNPq): 3.04.03.03-0 Circuitos Eletrônicos.

RESUMO: A implementação de redes neurais artificiais em *hardware* de pequeno porte (*TinyML*) está diretamente relacionada com os pilares da Indústria 4.0, visto que alguns dados coletados por sensores podem ser tratados localmente ao invés de serem enviados para a nuvem, garantindo maior segurança dos dados e maior velocidade de processamento. Portanto, o presente projeto consiste em implementar, em circuito integrado do tipo FPGA, uma rede neural previamente treinada com a arquitetura Perceptron Multicamadas com aprendizado supervisionado para classificar dados de acelerômetros com diferentes padrões de vibração. A topologia da rede utilizada foi 4-5-1, sendo 4 entradas, 5 neurônios na camada intermediária e 1 na saída. Foi utilizada a linguagem VHDL (*VHSIC Hardware Description Language*) para a descrição comportamental da rede. O ambiente de desenvolvimento foi o Quartus II, também utilizado como simulador, considerando o FPGA da família MAX 10. A rede neural implementada em FPGA apresentou acurácia de 75%, resultado semelhante ao obtido pela rede implementada em software, 79%, indicando que é possível o emprego de FPGAs em aplicações onde não se dispõe de acesso à rede e/ou computadores.

PALAVRAS-CHAVE: redes neurais artificiais; FPGA; Perceptron Multicamadas; análise vibracional; baixo custo.

IMPLEMENTATION OF ARTIFICIAL NEURAL NETWORK IN FPGA FOR VIBRATIONAL ANALYSIS OF ACCELEROMETER SIGNALS

ABSTRACT: Implementing artificial neural networks in small hardware (TinyML) is directly related to the pillars of Industry 4.0, as some data collected by sensors can be processed locally instead of being sent to the cloud, ensuring greater data security and faster processing speed. Therefore, this project aims to implement on FPGA type integrated circuit a pre-trained neural network with a Multi-Layer Perceptron architecture using supervised learning to classify accelerometer data with different vibration patterns. The network topology used was 4-5-1, with 4 inputs, 5 neurons in the intermediate layer, and 1 output. The VHDL (VHSIC Hardware Description Language) was used for the behavioral description of the network. The development environment was Quartus II, also used as a simulator, considering the MAX 10 family FPGA. The network implemented on FPGA showed an accuracy of 75%, a result similar to the network implemented in software, 79%, indicating that it is possible to use FPGAs in applications where network and/or computer access is unavailable.

KEYWORDS: artificial neural networks; FPGA; Multilayer Perceptron; vibrational analysis; low cost.

INTRODUÇÃO

As redes neurais artificiais (RNAs) são modelos computacionais inspirados no funcionamento do cérebro humano e compostas por unidades de processamento chamadas neurônios, que estão interligados por conexões sinápticas. Elas podem ser usadas para resolver problemas de classificação, e de regressão. Uma das arquiteturas mais comuns é o *Multilayer Perceptron* (MLP) (Haykin, 2009). Ela é capaz de processar dados vindos do mundo externo, como sensores. Para este projeto, o sensor empregado é o acelerômetro, que detecta mudanças na velocidade de um objeto em relação ao tempo nos três eixos. Ele é usado em muitas aplicações, como dispositivos móveis e em veículos (Saraiva, 2013). Neste projeto, os dados coletados por este sensor vêm de um pequeno ventilador, cujas hélices estão nas condições de normalidade e de desequilíbrio, este último causado por desbalanceamento devido a distribuição assimétrica de massas. A rede neural tem por objetivo classificar os tipos de desequilíbrios a partir dos dados dos sensores. Ela será implementada em linguagem de descrição de hardware para ser embarcada em um *Field-Programmable Gate Array (FPGA)*, um tipo de circuito integrado com grande capacidade de paralelismo e completamente flexível no que diz respeito à reconfiguração (Adetiba, et al., 2014). Este projeto visa a avaliar a viabilidade dessa implementação para análise vibracional de acelerômetros e, com isso, pretende-se analisar aspectos desta implementação, como a representação dos pesos e dos dados de entrada e a sua viabilidade face ao número de blocos lógicos disponíveis no dispositivo.

FUNDAMENTAÇÃO TEÓRICA

O algoritmo Perceptron Multicamadas ou MLP (*Multilayer Perceptron*) é um tipo de rede neural composto por neurônios interconectados em que os sinais transmitidos entre eles possuem uma direção, percorrendo uma camada de entrada e passando por camadas intermediárias até chegar a saída (Popescu et al., 2009). Essa rede é composta por mais de um Perceptron simples, que é um neurônio dessa rede e que possui um comportamento capaz de lidar com as entradas para, posteriormente, gerar uma saída segundo a Equação 1:

$$u = \sum_{i=1}^{N} (x_i \times w_i) + b \qquad (1)$$

onde u é o potencial de ativação do neurônio; N é o número de entradas da rede; i é o índice com relação ao número de entradas; x é a entrada; w é o peso; b é o viés.

O resultado da soma ponderada representado por u é usado como entrada de uma função de ativação não linear, como a função degrau ou a função sigmoidal, por exemplo. A função degrau atribui à saída o valor “0” se u for negativo e “1” se u for maior ou igual a 0. Portanto, o Perceptron Multicamadas é uma rede de neurônios artificiais que propagam seus valores calculados a partir da soma ponderada e da função de ativação para o próximo neurônio da rede, a fim de que, no final, ocorra uma classificação dos dados de entrada.

Na Figura 1, é possível analisar um esquemático de uma rede neural artificial com 4 entradas, 5 neurônios na camada intermediária e 1 neurônio na camada de saída, sendo esse o modelo adotado neste projeto. Nas entradas, a título de ilustração, estão representados os dados dos acelerômetros (velocidade, x, y, z). Cada interconexão carrega um peso (w), assim, cada neurônio realiza a soma ponderada citada previamente e o resultado desta passa pela função de ativação; o valor obtido na função de ativação da primeira camada irá para o neurônio de saída, onde haverá uma nova soma ponderada com outros valores de pesos, classificando, enfim, os dados de entrada como ‘1’ ou ‘0’.

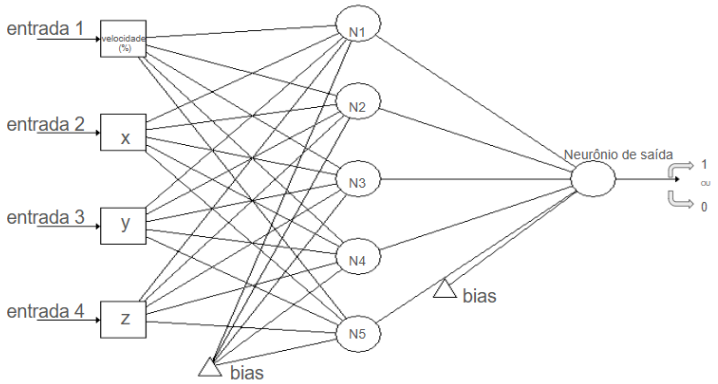

FIGURA 1. Representação visual de uma RNA com 4 entradas, 5 neurônios na camada intermediária e 1 neurônio na camada de saída.

O FPGA, por sua vez, é uma matriz de portas programáveis em campo, constituído de blocos lógicos interconectados eletricamente a partir de uma programação e pode proporcionar funções lógicas combinacionais ou sequenciais (Sulaiman et al., 2009). As características mais importantes do FPGA são a capacidade de paralelismo e de reconfiguração do circuito programado (Adetiba et al., 2014), sendo que a configuração pode ser realizada a partir das linguagens VHDL ou Verilog, que permitem a simulação e a verificação do comportamento do circuito descrito antes da implementação física de fato.

Um acelerômetro detecta variações de velocidade e direção de um objeto ao longo do tempo, medindo sua aceleração relativa à posição nas três dimensões do espaço (Saraiva, 2014). O acelerômetro utilizado para a obtenção dos dados utilizados neste projeto é capacitivo, já que o sensor é constituído por placas capacitivas que variam sua capacitância de acordo com a aceleração captada pelo sensor em virtude de alguma perturbação, considerando os eixos ortogonais (x, y e z) (Saraiva, 2014).

MATERIAIS E MÉTODOS

Dados de entrada provenientes de acelerômetros vieram da base de dados disponível em UCI (2021), elaborada por Scalabrini Sampaio et al. (2019).

Foram utilizados dois programas para o treinamento da rede neural proposta para este estudo, sendo o *software Multiple Back-Propagation* (1999 - 2016) e o ambiente Google Colab com a utilização da linguagem Python (FOUNDATION, 2024) e suas bibliotecas Numpy (NUMPY, 2024) e Tensor Flow (TENSOR FLOW, 2024). Além disso, foi utilizado o software Quartus (Quartus Prime 18.1) Lite Edition para desenvolver a rede neural em linguagem de descrição de hardware VHDL e realizar simulações acerca do funcionamento do algoritmo.

É importante mencionar que o principal objetivo deste projeto de pesquisa, é replicar em um FPGA de baixo custo uma rede neural previamente treinada em um computador de uso geral, analisando seu comportamento face ao desempenho da rede treinada em software, independentemente de esta apresentar uma acurácia elevada ou não.

A base de dados apresenta diferentes registros de vibração, coletados a partir de ventiladores (Akasa AK-FN059 com 1900 rpm de velocidade máxima de rotação) com pequenas cargas em suas hélices, a fim de causar diferentes tipos de desbalanceamento. Foi utilizado o acelerômetro MM8452Q para registrar as três maneiras de desequilíbrio. Foram configuradas 17 velocidades de rotação para as três condições de desequilíbrio, que variam de 20% a 100% da velocidade máxima do ventilador, com intervalos de 5%. As medições das coordenadas (x, y, z), coletadas com frequência de 20ms/min,

totalizam 3000 registros por velocidade, o que resulta em 51.000 medições por configuração e totalizam 153.000 registros de vibração.

Com a intenção de delimitar o problema e facilitar a implementação da rede em VHDL, os dados de entrada foram multiplicados por 100, evitando trabalhar com valores muito próximos a zero. Além disso, apenas duas das três classes disponíveis no banco de dados foram usadas, sendo elas: a classe com pesos em pás opostas (180°) e a classe com pesos em pás vizinhas. O conjunto de dados foi editado para atender a essa nova delimitação, apresentando 2746 dados para treino (sendo 1366 correspondentes a classe de pesos em pás opostas e 1360 para a classe de pesos em pás vizinhas) e 687 dados para teste da rede (sendo 350 para a classe de pesos em pás opostas e 337 para a classe de pesos em pás vizinhas).

O treinamento foi realizado em Python, no ambiente Google Colab, com a biblioteca Tensor Flow e a interface Keras. Foi implementada uma rede MLP com arquitetura 4-5-1, sendo 4 entradas, 5 neurônios de camada intermediária e um neurônio de saída, com função de ativação sigmoide. Esta arquitetura obteve a maior acurácia (79,0%) quando comparada às demais testadas, dentre elas: 4-3-1, 4-4-1, 4-5-1, 4-6-1 e 4-5-3-1.

Na implementação no FPGA, foi realizada uma descrição comportamental em VHDL, sendo construído um bloco de código simples chamado “Perceptron” para que sua funcionalidade pudesse ser replicada para outro bloco chamado “neural network” que se trata do bloco que comporta os cinco perceptrons e realiza as devidas operações das entradas com os pesos e, posteriormente, envia o seu resultado para o bloco denominado “last_neuron”, que classifica o conjunto de dados de entrada.

Foi adotada uma resolução de 10 bits para os dados de entrada e para os pesos, em que os dados são representados em complemento de 2, destacando que os pesos foram multiplicados por 10 para que pudessem ser representados com maior facilidade em VHDL. Uma vez que 10 bits permitem representar 1024 valores, considerando que os conjuntos de entradas e pesos possuem valores negativos, é possível representar valores de -512 até +511. Após uma análise do conjunto de entrada, constatou-se que poucos valores ultrapassavam esse intervalo e foram removidos do conjunto, para evitar um ajuste incorreto dos pesos e conservar a representação em 10 bits. Além disso, definiu-se também a representação dos pesos com 10 bits. Uma vez que os pesos e vieses obtidos com a linguagem Python possuíam valores entre -8 e +15, estes foram multiplicados por 10 para que pudessem ser representados de maneira decimal na linguagem de descrição de hardware. Por sua simplicidade, a função de ativação degrau foi usada. O diagrama de blocos do sistema está representado na Figura 2.

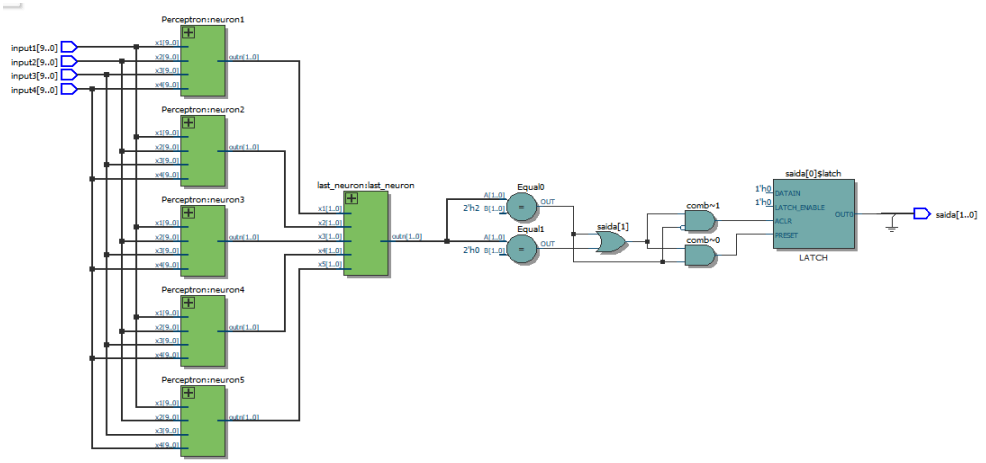

FIGURA 2. Esquemático RTL (*Register Transfer Level*) gerado pelo software Quartus II a partir do código descrito em VHDL.

RESULTADOS E DISCUSSÃO

Os testes no ambiente de simulação do software Quartus II foram realizados pela ferramenta University Program VWF (*Vector Waveform Program*), que permite visualizar as formas de onda na saída a partir das entradas e permite a depuração do projeto implementado.

Cabe ressaltar algumas diferenças entre o código em Python e a implementação em FPGA. No primeiro caso, foi utilizada a função de ativação sigmoide, enquanto no VHDL, por simplicidade e tempo, foi implementada a função degrau. Além disso, o treinamento em Python possui valores mais precisos, já que a implementação em VHDL foi feita com o arredondamento para valores inteiros tanto do conjunto de dados de entrada quanto do conjunto de pesos, já que não foi utilizada a técnica de ponto flutuante.

A Tabela 1 mostra que a rede neural artificial simulada para implementação em FPGA apresentou uma acurácia total de 75%, ou seja, um desempenho bastante próximo ao resultado obtido na rede implementada em Python (79%). É possível observar uma maior quantidade de acertos da rede para classificar os dados com saída 1 (88%), que representam a classe em que os pesos estão distribuídos em pás vizinhas, do que para classificar os dados com saída 0, que representam a classe em que os pesos estão em pás opostas, com acurácia de 62%.

TABELA 1. Relação entre o número de dados classificados corretamente e incorretamente pela rede neural descrita em linguagem de descrição de hardware.

	TOTAL TESTADO	Nº DE ACERTOS PELA REDE DESCRITA EM VHDL	Nº DE ERROS PELA REDE DESCRITA EM VHDL
DADOS COM SAÍDA 1	50	44	6
DADOS COM SAÍDA 0	50	31	19

Com relação à estrutura, uma vez que o FPGA da família MAX10 utilizado possui 40.000 blocos lógicos disponíveis, é importante mencionar que a rede neural artificial em questão utilizou apenas 336 blocos, o que resulta em aproximadamente 0,84% de ocupação do FPGA em questão.

CONCLUSÕES

Uma rede neural do tipo MLP foi implementada em um FPGA de baixo custo (família MAX10) a fim de se comparar seu desempenho face à mesma rede implementada em Python. Para isso, dados de sensores do tipo acelerômetro foram utilizados, sendo que dois padrões de vibração serviram como classes a serem identificadas. Os resultados obtidos demonstraram que o comportamento das duas redes foi semelhante, sendo a acurácia total da rede em software igual a 79% enquanto a rede implementada em FPGA atingiu acurácia de 75%. Ou seja, é possível utilizar um FPGA de baixo custo, próximo à base da linha de produtos Altera, para aplicações de redes neurais artificiais para classificação de dados tais como a realizada neste projeto.

Para trabalhos futuros, propõe-se a implementação da função de ativação sigmoide ou de alguma outra que possa melhor se adequar ao hardware; a utilização da técnica de ponto flutuante para representação do conjunto de dados de entrada e do conjunto de pesos de maneira mais precisa; e também a utilização de um maior conjunto de dados para as etapas de treinamento e de teste da rede neural. Além disso, o aprofundamento nas características específicas do FPGA devem ser investigados, tais como tempo de processamento, paralelismo e gasto de energia.

AGRADECIMENTOS

Os autores agradecem ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pela bolsa fornecida durante o projeto de pesquisa, a qual foi fundamental para a realização deste trabalho.

REFERÊNCIAS

ADETIBA, Emmanuel. IBIKUNLE, F.A. DARAMOLA, S.A. OLAJIDE, A.T. **Implementation of Efficient Multilayer Perceptron ANN Neurons on Field Programmable Gate Array Chip.** IJET-IJENS, 2014.

BRANCO, Sérgio; FERREIRA, André G.; CABRAL, Jorge. **Machine Learning in resource-scare embedded systems, FPGAs, and end-devices: A survey.** Electronics, v. 8, n. 11, p 1289, 2019.

FLECK, Leandro. TAVARES, Maria. EYING, Eduardo. HELMANN, Cristina. ANDRADE, Minéia. **Redes Neurais Artificiais: Princípios Básicos.** Revista Eletrônica Científica Inovação e Tecnologia. UTFPR, v. 1, n. 12, p. 47-57, 2018. Disponível em:4330-15577-1-PB-libre.pdf (d1wqtxts1xzle7.cloudfront.net). Acesso em: 14 jul. 2024.

FPGA Intel® MAX® 10. 2014. https://www.intel.com.br/content/www/br/pt/products/details/fpga/max/10.html. Acesso em 28/05/2023.

HAYKIN, Simon. **Neural Networks and Learning Machines** (3rd ed.). Prentice Hall. 2009.

INTEL. **Quartus Prime: Software de Design FPGA**. Versão 21.1. Santa Clara: Intel Corporation, 2021. Disponível em: https://www.intel.com/content/www/us/en/software/programmable/quartus-prime/overview.html. Acesso em: 9 jan. 2024.

POPESCU, Marius-Constantin; BALAS, Valentina E.; PERESCU-POPESCU, Liliana; MASTORAKIS, Nikos. **Multilayer Perceptron and Neural Networks. WSEAS Transactions on Circuits and Systems**, 2009. Disponível em: https://www.academia.edu/52398369/Multilayer_perceptron_and_neural_networks. Acesso em: 20 ago. 2024.

SARAIVA, Fábio.**Propriedades de um acelerômetro eletrônico e possibilidades de uso no ensino de mecânica.** Latin American Journal of Physics Education, v. 7, n. 1, p. 1-10, 2013. Disponível em: http://lajpe.org/march13/6_LAJPE_739_Fabio_Saraiva_preprint_corr_f.pdf. Acesso em: 1 set. 2024.

SCALABRINI SAMPAIO, Gustavo et al. **Prediction of motor failure time using an artificial neural network.** Sensors, v. 19, n. 19, p. 4342, 2019.

SULAIMAN, Nasri; OBAID, Zeyad Assi; MARHABAN, M. H.; HAMIDON, Mohd Nizar. **Design and Implementation of FPGA-Based Systems - A Review.** Australian Journal of Basic and Applied Sciences, v. 3, n. 4, p. 3575-3596, 2009. Disponível em: https://www.researchgate.net/publication/268424617_Design_and_Implementation_of_FPGA-Based_Systems_-A_Review. Acesso em: 2 mai. 2024.

TARI, Zahir. **Security and privacy in cloud computing.** IEEE Cloud Computing, v. 1, n. 01, p. 54-57, 2014.

UCI - **Accelerometer Data Set. 2021.** Disponível em: https://archive.ics.uci.edu/ml/datasets/Accelerometer. Acesso em 28/05/2023.