IMPLEMENTAÇÃO DE UMA REDE NEURAL ARTIFICIAL EM FPGA: APLICAÇÃO DA MLP COMO ARQUITETURA MODULAR

CARLOS ALBERTO DE A. SILVA, ADRIÃO D. DORIA NETO, JOSÉ ALBERTO N. DE OLIVEIRA, JORGE DANTAS DE MELO

DEE/CT - Universidade Federal do Rio Grande do Norte - UFRN, Campus Universitário Lagoa Nova, CEP 59072-970 Natal – RN – Brasil

E-mails: carlos77.albuquerque@gmail.com, adriao@dca.ufrn.br, nicolau@ufrnet.br, jdmelo@dca.ufrn.br

Abstract — This work present an implementation of artificial neural networks on hardware, using a FPGA programmable device. For such application, it was decided to use two development approaches, one in software, that uses MATLAB® for training and validation of the network, and another one on hardware able to deal with the synaptic weights and bias by the means of fixed point arithmetics using VHDL. The hardware architecture of the neural network developed for this work has two inputs, two hidden neurones and a neurone in the output, each of them showing as a result of the activation function a value refereed on a lookup table.

Keywords — FPGA, Hardware, VHDL, Neural Network, Arithmetic Fixed Point.

Resumo — O presente trabalho apresenta uma implementação de redes neurais artificiais em *hardware* por meio de um dispositivo programável do tipo FPGA. Para esta aplicação convencionou-se utilizar duas abordagens de desenvolvimento, uma em *software* que utiliza o MATLAB® para treinamento e validação da rede e outra em *hardware*, capaz de lidar com os pesos sinápticos e *bias* por meio da aritmética de ponto fixo utilizando VHDL. A arquitetura em *hardware* da rede neural desenvolvida para este trabalho possui duas entradas, dois neurônios ocultos e um neurônio na saída, com cada neurônio apresentando como resultado da função de ativação um valor referenciado em uma *lookup table*.

Palavras-chave — FPGA, *Hardware*, VHDL, Redes Neurais Artificiais, Aritmética Ponto Fixo.

1 Introdução

Por constituírem uma das ramificações da Inteligência Artificial (IA), cada vez mais as Redes Neurais Artificiais estão sendo utilizadas nos mais variados campos de pesquisa, que necessitam de um algoritmo para solucionar problemas de reconhecimento de padrões, classificação e aproximação de funções.

O desenvolvimento de uma rede neural pode ser realizado tanto em *software* quanto em *hardware*, havendo vantagens e desvantagens entre ambos. Para o desenvolvimento em *software*, as vantagens recaem sobre a facilidade e o tempo gasto para execução da rede, já as desvantagens estão relacionadas à lentidão dos dados, por serem processados sequencialmente. Para a implementação em *hardware*, as vantagens estão relacionadas ao paralelismo intrínseco das redes neurais, enquanto que as desvantagens ficam a cargo de se alcançar um equilíbrio razoável na precisão de *bits*. Dentre as características inerentes das redes neurais, duas chamam a atenção para o desenvolvimento em *hardware*, são elas: **não-linearidade,** por permitir a resolução de problemas que não sejam linearmente separáveis; **processamento paralelo,** que é a capacidade de receber múltiplas informações e testá-las ao mesmo tempo [1,6,10].

Atualmente, verificou-se um aumento no uso do *Field Programmable Gate Array* (FPGA) como plataforma para implementar as Redes Neurais Artificiais (RNA) em dispositivos dedicados, explorando o alto poder de processamento, o baixo custo e a facilidade de programação. Outra característica apresentada pela maioria dos FPGA é possuir blocos de memória *Random Access Memory* (RAM) internas do tipo *dual port*, com canais de leitura e escrita, nos quais é possível executar simultaneamente essas duas operações com uso de endereços distintos [2,9].

Diante do exposto, o objetivo deste artigo é colocar em prática o uso de redes neurais em *hardware*, com uma arquitetura capaz de acrescentar ou de retirar qualquer neurônio, viabilizando uma rede modular e com aritmética de ponto fixo utilizando FPGA.

2 Representação dos Dados em Ponto Fixo

Em se tratando de redes neurais artificiais e o seu desenvolvimento em hardware, deve-se dar uma atenção especial à notação numérica utilizada para os valores de entrada e saída dos neurônios, assim como também para os valores dos pesos sinápticos e *bias*.

Para este trabalho foi adotada a notação em ponto fixo por duas razões: a primeira está relacionada à falta de suporte aos valores não inteiros pelas ferramentas de síntese, e a segunda, ao aumento no número de *Configuration Logical Blocks* (CLB) empregados, caso utilizasse a notação em ponto flutuante [2,8].

O ponto fixo é utilizado como alternativa à representação de um valor numérico em ponto flutuante, podendo ser utilizado na representação de frações por meio da manipulação da vírgula fracionária (*radix point*), pela qual o programador pode escalar inteiros em valores fracionários.

Pode-se observar na Figura 1, a seguir apresentada, a forma como o programa interpreta uma sequência de 16 *bits* de acordo com a localização do ponto *radix*. Com isso, um programador pode aumentar a precisão da fração movendo o ponto de *radix* para a esquerda, consequentemente diminuindo a parte inteira do número [5,12].

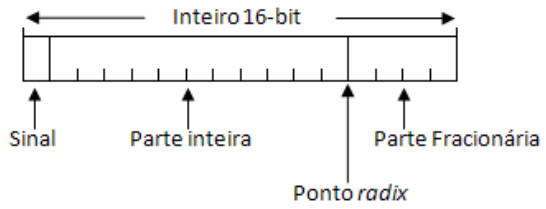

Figura 1 - Representação Numérica de 16 *bits* em Ponto Fixo.

Para a representação dos dados da rede neural foi utilizada a seguinte notação:

Número (ponto fixo) = <*mantissa*>**S**<*expoente*>

Como colocado em [5], utiliza-se como notação em ponto fixo a expressão <*mantissa*>**S**<*expoente*>, para diferenciar da notação de ponto flutuante, definida por <*mantissa*>**E**<*exponente*>, ou seja, considerando a **mantissa** como sendo sempre um número inteiro, o **expoente** como um fator de escala e o **S**, expressando que a mantissa deve ser dimensionada por $2^{\exp oente}$ para determinar o valor do número de ponto fixo. Como pode ser visualizado na Tabela 1:

Tabela 1 - Codificação empregada.

Valor Real	Valor Codificado
0.8125	**26**S-5

Para se trabalhar com os dados da rede neural em *hardware*, foi feita uma conversão em todos os valores de entrada e saída, assim como nos respectivos pesos e bias de cada neurônio, ou seja, de posse dos valores, os mesmos foram multiplicados por 10^{-1}, com o intuito de obter um padrão numérico com parte inteira 0 (zero).

Assim, com os números em um novo formato, torna-se possível fazer os devidos dimensionamentos, com a sua representação em um inteiro de 16 *bits*. Nas Equações 1 e 2 observa-se a obtenção da **mantissa** e do **expoente** (fator de escala) para qualquer valor de *x* entre -32767.0 até +32767.0.

$$fator = -\left(\frac{\log\left(\frac{32767}{|x|} \right)}{\log(2)} \right) \quad (1)$$

$$mantissa = 2^{-fator} \times x \quad (2)$$

Deve-se realizar um truncamento no valor atribuído ao **fator** (Equação 1), afim de se obter apenas a parte inteira do resultado. Para o número que se pretende representar em ponto fixo, deve-se atribuí-lo a *x*. Quando *x* for 0.0, tanto a **mantissa** quanto o **fator** vão ser sempre zero.

3 Estrutura do *Multilayer Perceptron* (MLP)

A estrutura conceitual das redes neurais artificiais está relacionada ao modelo biológico, o qual é associado com o conexionismo do cérebro humano. Ela é constituída por elementos interconectados, chamados de neurônios, reorganizados em camadas [4].

Ao se trabalhar com RNA para a resolução de um problema específico, se faz necessário uma escolha prévia de uma arquitetura, de um método de aprendizado e de um algoritmo de aprendizado. Desse modo, o trabalho deteve-se na utilização de uma rede do tipo *Multilayer Perceptron* (MLP) como arquitetura, a qual é caracterizada por ser uma rede alimentada adiante com os elementos ou os nós dispostos em uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída (Figura 2).

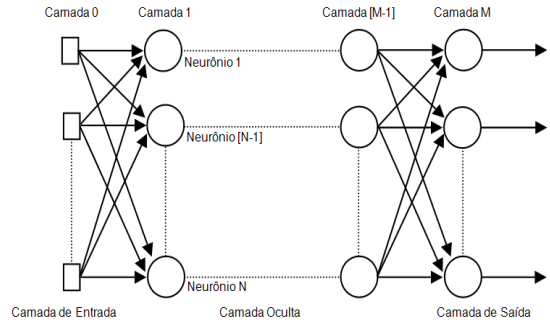

Figura 2 - Arquitetura da rede *Multilayer Perceptron* (MLP).

O método utilizado para o aprendizado supervisionado caracteriza-se por apresentar pares de exemplos de entrada e saída desejada, sendo este último comparado com a saída da rede de forma a ajustar os seus parâmetros, tais como: pesos sinápticos e *bias*, como também a minimização da diferença entre a saída apresentada pela rede e a saída desejada. Para o algoritmo de aprendizado, se optou pelo *backpropagation* com aprendizagem por correção de erros [6].

Todo processo de treinamento e aprendizado adotado pela rede neural foi realizado *off-line*, a partir do MATLAB®. Finalizado o aprendizado e de posse dos pesos sinápticos e bias corrigidos, passou-se para a descrição da rede neural em uma linguagem de descrição de *hardware* (HDL) [8].

4 RNA com FPGA

4.1 Prática da RNA em FPGA

Para a concepção de uma RNA em FPGA, foi adotada como método de entrada a descrição em *Very High Speed Integrated Circuit Hardware Description Linguagem* (VHDL), que é uma linguagem de descrição de *hardware* suportada pela maioria das ferramentas de síntese. O VHDL permite que circuitos complexos sejam projetados a partir de uma descrição estrutural, fluxo de dados e comportamental [7, 9, 11, 13].

Em uma arquitetura neural que envolva seu desenvolvimento em dispositivos reprogramáveis, os valores de entrada como os de saída devem ser transcritos para binário, com o propósito de se adequarem à arquitetura digital dos FPGAs. Entretanto, a arquitetura envolvida tem que ser cuidadosamente projetada e testada, para que possa atender os requisitos de velocidade, de processamento em tempo real e de como incorporar a função de transferência no FPGA [1,3].

4.2 Descrição estrutural do neurônio em FPGA

A abordagem utilizada para representar um neurônio em FPGA está relacionada à divisão do neurônio em duas estruturas de blocos: função de ativação e função de transferência, nomeadas, respectivamente, como **net** e **fnet**, com representação dos valores em binário.

Os blocos de multiplicação, de deslocamento e de soma foram desenvolvidos para compor o bloco **net** (Figura 3). Inicialmente, os *n* valores de entrada x_i de 16 *bits* são repassados pelo bloco **net** para os dois blocos de multiplicação, **mult1** e **mult2**, para serem multiplicados pelos correspondentes pesos w_{ij}, declarados como constantes.

De posse dos valores das multiplicações representados em 32 *bits* decorrentes da soma entre os expoentes, em que, de agora em diante, necessita-se da utilização de outros dois blocos, **shift1** e **shift2**, responsáveis pela mudança dos valores na base 32, resultante das operações de multiplicação, para a base 16 por meio do deslocamento de *bits*.

Tendo realizado o deslocamento e obtido os valores corrigidos para 16 *bits*, o bloco **soma** pode realizar a soma dos valores corrigidos com o valor de *bias*, uma vez que todos os valores estão na representação numérica de 16 *bits*, gerando assim, a saída do bloco **net**.

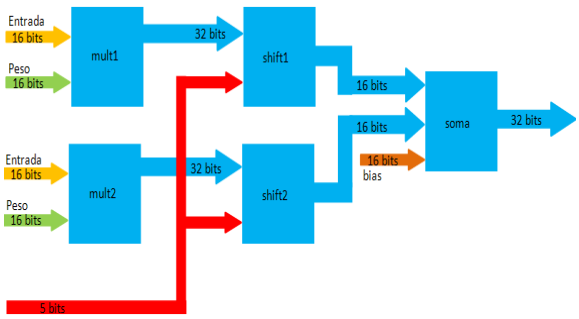

Figura 3 - Diagrama de blocos do bloco **net**.

Em relação à implementação da função de transferência do tipo sigmóide em FPGA, a solução adotada foi a construção de *lookup tables* por intermédio do bloco **fnet** (Figura 4). Esse bloco incorpora outros dois blocos: um gerador de endereços e uma *Read Only Memory* (ROM).

O bloco **gera-ender** (gerador de endereços) receberá como entrada os valores de saída vindos do bloco **net**, redirecionando-os para os respectivos endereços da ROM, tendo como conteúdo a representação do valor da função sigmóide, representando assim a saída do neurônio.

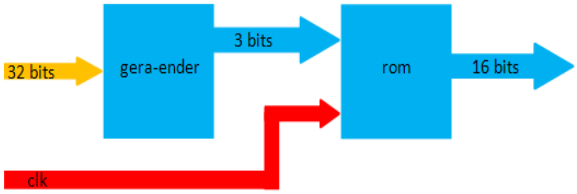

Figura 4 - Diagrama de blocos do bloco **fnet**.

4.3 Descrição estrutural da rede neural em FPGA

O recurso proposto para viabilizar o estudo das redes neurais artificiais em FPGA teve como base a resolução da função lógica XOR. A arquitetura neural usada para resolver o problema do XOR foi composta por três camadas, uma de entrada, uma oculta e uma de saída, com dois neurônios dispostos na camada oculta e um neurônio na camada de saída.

A rede neural partiu da descrição em VHDL do bloco neurônio, utilizando como ambiente para desenvolvimento e simulação lógica o programa Quartus® II da Altera®. Com o bloco neurônio, dá-se início à descrição da rede, a partir do DSP Builder®, que é uma ferramenta de desenvolvimento gráfica utilizada no ambiente SIMULINK/MATLAB®.

A descrição da rede neural utilizando recursos do DSP Builder® no SIMULINK® caracterizou-se pela importação e replicação do bloco neurônio pelo componente *HDLImport* do DSP Builder®. Com a replicação, os blocos neurônios foram dispostos em camadas, caracterizando a rede neural sugerida para o problema do XOR. Com isso, a compilação do projeto por meio do bloco *Signal Compile* permitiu a geração do *Register Transfer Level* (RTL), compilação, carregamento e teste do projeto em *hardware*.

5 Resultados Obtidos

A análise comparativa do tempo de execução se resume a confrontar os resultados finais obtidos em *software* com os alcançados pelo *hardware*, para os quais a execução em *software* reportou um tempo de 21ms, e com relação ao FPGA o tempo gasto para a execução foi de 162,23µs mais rápido do que o realizado em *software*.

Na execução em FPGA, o erro máximo calculado foi de 0.01562 frente ao esperado do modelo XOR implementado no MATLAB® que foi de 0.0150.

O processador foi programado no FPGA EP2C70F672C6 da Altera® por meio do *software* Quartus® II. A tabela a seguir apresenta os principais resultados da implementação:

Tabela 2 - Resultados da Implementação no FPGA EP2C70F672C6.

Dispositivo	EP2C70
Elementos Lógicos	90 (<1%)
Número total de registradores	86%
Número de bits de memória	0%
Multiplicadores Dedicados de 9 bits	1 (<1%)
Frequência máxima de clock	260,01 MHz

6 Perspectivas e Trabalhos Futuros

As redes neurais artificiais são amplamente empregadas em aplicações industriais. Com tais aplicações, explorando as características das RNAs de operarem como classificadores de padrões ou interpoladores universais.

Nesse sentido, propõe-se uma plataforma de software embarcado em FPGA, cujo *software* implementado será uma Rede Neural Artificial reconfigurável do tipo *Perceptron* de múltiplas camadas (MLP), constituída por neurônios artificiais que podem ser reorganizáveis e reprogramados dinamicamente. O *software* será capaz de realizar tanto operações em tempo real, quanto executar algoritmos e estratégias de controle distribuído voltadas a aplicações na indústria do Petróleo e Gás Natural.

Quanto a trabalhos futuros, pretende-se desenvolver e embarcar em um FPGA uma rede MLP, capaz de realizar medições exatas e confiáveis dos volumes de gás, assegurando, assim, níveis de erro e incerteza aceitáveis, ficando dentro dos limites regulamentados pela indústria do gás.

Outra proposta de grande valia seria a implementação de uma RNA, em FPGA, para inferir na qualidade do Gás de Petróleo Liquefeito (GLP).

7 Conclusão

A arquitetura neural desenvolvida em um dispositivo configurável utilizando a placa de desenvolvimento DE2 da Altera®, com FPGA EP2C70F672C6, demonstra eficiência na resolução do lógico XOR e precisão dos valores obtidos durante a execução da rede neural em FPGA.

As limitações encontradas durante a execução deste trabalho estão relacionadas com a aritmética em ponto flutuante, não suportada pelas ferramentas de síntese, como também, com implementação da função de transferência do tipo sigmóide. Para superar tais limitações, foram usados como artifícios, respec-

tivamente: a aritmética de ponto fixo e o *lookup table* para o armazenamento dos valores da função de transferência obtidos com a realização *off-line* do treinamento da rede neural por meio do MATLAB®.

Com a especificação do bloco neurônio constituído pelos blocos **net** e **fnet**, verificou-se a possibilidade de se criar uma arquitetura modular capaz de aumentar ou diminuir o número de neurônios da rede, permitindo, desse modo, a sua utilização em outras arquiteturas neurais, garantindo ainda a exatidão dos resultados.

A arquitetura da rede neural desenvolvida em FPGA, apesar de simples, qualificou os métodos e abordagens desenvolvidos, como aptos para o transporte da fase de simulação para sistemas reais, atendendo aos requisitos estabelecidos para o projeto.

Referências Bibliográficas

[1]A. A. Hassan, A. Elnakib, M. Abo-Elsoud. (2008) FPGA-Based Neuro-Architecture Intrusion Detection System, *Proceedings of the Internatinal Conference on Computer Engineering & Systems*, Cairo, pp. 268-273.

[2]Amore, Robert d'. VHDL: Descrição e Síntese de Circuitos Digitais. Rio de Janeiro: LTC, 2005.

[3]E.M. Ortigosa, A. Cañas, E. Ros, P.M. Ortigosa, S. Mota e J. Díaz. Hardware description of multi-layer perceptrons with different abstraction levels, *Microprocessors and Microsystems*, pp. 435–444, 2006.

[4]Haykin, Simon. Redes Neurais: Princípios e Pratica. Porto Alegre: Bookman, 2001.

[5]Labrosse, Jean J. Embedded Systems Building Blocks. San Francisco: CMP Books, 2002.

[6]Ludwig Jr., O. e Costa, Eduard Montgomery M. Redes Neurais: Fundamentos e Aplicações com Programas em C. Rio de Janeiro: Editora Ciência Moderna LTDA, 2007.

[7]M. Ali Çavuşlu, Cihan Karakuzu e Suhap Şahin. Neural Network Hardware Implementation Using FPGA. ISEECE, 2006.

[8]Rolf F. Molz, Paulo M. Engel e Fernando G. Moraes**.** (1999). Uso de um Ambiente Codesign para a Implementação de Redes Neurais, *Proceedings of the IV Brazilian Conference on Neural Networks*, São José dos Campos – SP, pp. 013-018.

[9]Vahid, Frank. Digital Design. Wiley, 2007.

[10]Valença, Mêuser. Aplicando Redes Neurais: Um Guia Completo. Olinda,PE: Ed.do Autor, 2005.

[11]Wang Qinruo, Yi Bo, Xie Yun e Liu Bingru. The Hardware Structure Design of Perceptron with FPGA Implementation. 2003.

[12]Xiaoguang LI, Medhat Moussa e Shawki Areibi (2005). Arithmetic formats for implementing Artificial Neural Networks on FPGAs. Canadian Journal on Electrical and Computer Engineering.

[13]Yamina Taright e Michel Hubin. FPGA Implementation of a Multilayer Perceptron Neural Network using VHDL, *Proceedings of ICSP*, pp. 1311-1314, 1998.