ARQUITETURA DE HARDWARE BASEADA EM REDES NEURAIS PARA APROXIMAÇÃO DE UMA FUNÇÃO NÃO LINEAR

Maicon A. Sartin*, Alexandre C. R. da Silva†

*Departmento de Computação
Universidade do Estado de Mato Grosso - UNEMAT
Colider, MT, Brazil

†Departmento de Engenharia Elétrica
Universidade Estadual Paulista - UNESP
Ilha Solteira, SP, Brazil

Emails: mapsartin@unemat.br, acrsilva@dee.feis.unesp.br

Abstract— The bio-inspired models are a counterpart to tradicional methods mainly in the problems solution with nonlinear behavior. Artificial neural networks are methods widely investigated by robustness, besides of the capacity of generalization, pattern recognition and learning. There are several methods and techniques for prototyping of artificial neural network in reconfigurable devices (FPGA) by its parallelism level. The contributions of this work is in the implementation of a Multilayer artificial neural network with one hidden layer and hyperbolic tangent for activation function. A nonlinear function approximation serves as application for system validation. The method employed in the implementation of the activation function is hybrid between peace-wise linear and combinational. The system makes results comparison in low and high level of abstraction. The results analysis are performed by means of the performance, used area in FPGA, and errors (absolute and relative).

Keywords— Artificial Neural Networks, Hyperbolic Tangent, FPGA, Floating Point.

Resumo— Os modelos bioinspirados são uma contrapartida aos métodos tradicionais, principalmente na solução de problemas não lineares. As redes neurais artificiais são métodos amplamente investigados pela sua robustez, além da capacidade de generalização, reconhecimento de padrões e aprendizado. Existem diversos métodos e técnicas para a prototipação de redes neurais artificiais em dispositivos reconfiguráveis (FPGA) pelo seu nível de paralelismo. A contribuição deste trabalho está na implementação de uma RNA Multicamadas com 4 neurônios, uma camada escondida e função de ativação tangente hiperbólica. A aplicação para validação do sistema da RNA é a aproximação de uma função não linear. O método empregado na implementação da função de ativação é híbrido entre divisão em partes lineares e combinacional. O sistema faz a comparação dos resultados em baixo e alto nível de abstração. A análise dos resultados são realizadas através do desempenho, da área utilizada na FPGA e dos erros, absoluto e relativo.

Palavras-chave— Redes Neurais Artificiais, Tangente Hiperbólica, FPGA, Ponto Flutuante.

1 Introdução

Atualmente, a inteligência computacional consiste de modelos inspirados na natureza para a resolução de problemas reais. A taxonomia divide a inteligência computacional em quatro grandes grupos, lógica nebulosa, computação evolucionária, redes neurais artificiais e inteligência artificial (Goldschmidt, 2010). As redes neurais artificiais (RNA) são modelos bioinspirados no cérebro humano para solucionar problemas em que os modelos tradicionais são ineficientes. RNAs têm como as principais especialidades à capacidade a generalização, o aprendizado de máquina e o reconhecimento de padrões. Esse modelo adquire dados do ambiente real e adapta-se conforme seu método de aprendizado e sua organização com as camadas, os neurônios e as funções de ativação, tal aprendizado conclui-se com a inferência dos dados para gerar resultados conforme a necessidade da aplicação alvo.

As RNAs são amplamente investigadas e prototipadas em dispositivos reconfiguráveis (FPGA). A relação entre RNA e FPGA tem aumentado, principalmente por dispor de desempenho associado a um alto nível de paralelismo, flexibilidade, reuso de componentes de hardware de propriedade intelectual (IP) e barramentos padronizados.

Uma das principais características da FPGA é a facilidade na atualização através da reconfiguração do sistema tornando-se mais dinâmico, inclusive para o desenvolvimento de novos projetos com o mesmo dispositivo. Existem três características essenciais no desenvolvimento de RNA em dispositivo reconfigurável, a representação dos dados, o tipo de arquitetura e o método para função de ativação não linear.

A representação dos dados é dividida em ponto fixo (da Silva et al., 2010) e flutuante (Santos et al., 2011). O uso de ponto fixo pode prejudicar a precisão e dificultar o seu reuso. Já o ponto flutuante, determina de forma organizada o sinal, o expoente e a fração, facilitando a manutenção e expansão do sistema de forma padronizada.

O tipo de arquitetura do sistema determina o fluxo de execução e os atrasos na computação dos dados do neurônio e consequentemente na RNA. Segundo Savich et al. (2007), existem diversos tipos de arquiteturas para computação dos neurô-

nios em FPGA.

Já os métodos, que frequentemente são utilizados na implementação da função de ativação não linear em FPGA são: Polinômios com diferentes ordens (Lee and Ko, 2006), mapeamento por LUT (Meher, 2010), divisão em partes lineares (PWL) (Amin et al., 1997) e Híbrido (Lin and Wang, 2008). Esses métodos tem a função de efetuar a aproximação da função de ativação e evitar a implementação direta de modelos matemáticos, devido ao alto custo na utilização de recursos computacionais e baixo desempenho.

A implementação em hardware da RNA MultiLayer Perceptron (MLP) é um dos principais desafios aos pesquisadores da área. Cada neurônio artificial realiza a computação dos seus dados de forma independente, tornando as RNAs estruturas massivamente paralelas e, tal componente, o mais importante de uma RNA. Em Ferreira and Barros (2010) os autores fazem a computação dos neurônios por meio do produto de matrizes. O processamento é feito em uma única entrada com todos os seus pesos (coluna) e depois por meio de deslocadores e somadores adquire a saída parcial do neurônio. O uso de recursos embarcados na FPGA são evidenciados por Zhang and Chen (2011) e (Al-Kazzaz and Khalil, 2008) para um melhor desempenho e redução de área da FPGA.

Neste trabalho apresenta-se uma arquitetura de hardware para uma RNA MLP e topologia 1-2-1 de quatro neurônios. A RNA é dividida em três partes principais a unidade de controle (UC), unidade de processamento (UP) e unidade de armazenamento (UA). A RNA faz uso da padronização IEEE Std 754 na computação do neurônio e realiza a aproximação de uma função não linear tangente hiperbólica com método próprio, na unidade de processamento. Na Seção 2 define-se a arquitetura do sistema detalhadamente, a metodologia empregada nas implementações e os principais componentes desenvolvidos. Os resultados obtidos e a comparação entre as implementações de alto e baixo nível são apresentadas na Seção 3. A Seção 4 finaliza o artigo com as conclusões da pesquisa.

2 Implementação em Hardware

A RNA foi desenvolvida em linguagem de programação de alto nível, com o Matlab sem toolbox, para comparação dos resultados obtidos em hardware. O desenvolvimento auxilia o projetista na escolha dos componentes a serem utilizados em seu sistema de baixo nível. Para o treinamento supervisionado da RNA adotou-se a regra delta em linguagem de alto nível e modo offline. No treinamento utilizou-se 256 amostras de entrada, com o devido valor desejado na saída da rede. O critério de parada usado foi o erro quadrático com limite de 10^{-12}, a convergência ocorreu com 56.987 épocas. A implementação em hardware (FPGA) adotou-se os mesmos pesos na RNA em alto nível e teve o uso da ferramenta CAD ISE na versão 14.2 da Xilinx.

O sistema implementa uma RNA MLP com três camadas, quatro neurônios e função de ativação tangente hiperbólica em todos. Para validação do sistema utilizou-se uma função não linear senóide para fazer a aproximação pela RNA, conforme a Equação (1), onde x é a entrada de dados da RNA.

$$S(x_i) = \frac{x_i}{e^{0.03}} \times sin(0.03 \times x_i) \quad (1)$$

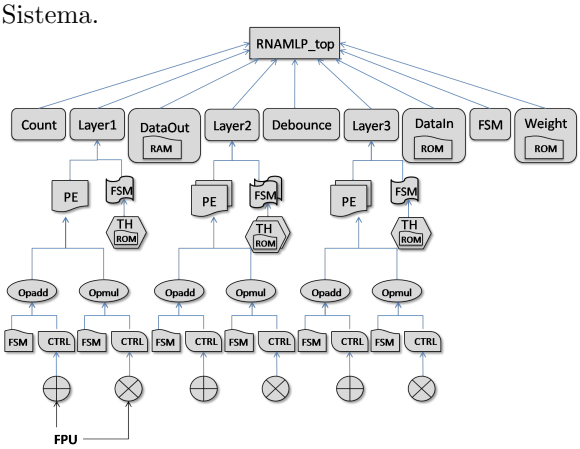
Para validação do sistema aplicou-se uma metodologia dividida em três etapas: preparação dos dados, execução do sistema e análise dos resultados. Na preparação dos dados definiu-se os pesos, dados de entrada do sistema e da função de ativação respeitando o padrão IEEE Std 754 (32 bits) para embarcar no hardware. A execução e a comparação do sistema realizaram-se em linguagem de alto e baixo nível de abstração. Os resultados obtidos no hardware (FPGA) pela simulação funcional foram decodificados para gerar gráficos e tabelas. A análise dos resultados é feita pela apresentação dos erros e área utilizada, os dados foram obtidos pelas distintas implementações da RNA, principalmente na alteração da função de ativação do sistema.

A relação entre os módulos e componentes utilizados na arquitetura do sistema apresenta-se na Figura 1, com as divisões das unidades de forma hierárquica. O sistema está organizado em três partes principais: a unidade de controle (UC), unidade de armazenamento (UA) e unidade de processamento (UP). A plataforma destinada a realização e validação do sistema foi a Spartan-3e (Xilinx). A plataforma de desenvolvimento é relativamente de baixo custo e contêm diversas interfaces com o usuário. Para a implementação em hardware (FPGA) usou-se VHDL, com módulos IP Core do próprio fabricante para a unidade de ponto flutuante e blocos de memória dedicados. Desta forma, para prototipar o sistema em outra plataforma basta integrar os módulos e respeitar suas características específicas, trazendo portabilidade e expansão ao sistema.

2.1 Unidade de Controle

A unidade de controle faz a administração do fluxo de execução de forma geral e em componentes específicos. O fluxo correto dos dados na RNA não admite problemas ocasionados na execução concorrente das operações, por exemplo, a informação de saída de uma camada não pode ser sobrescrita antes que a próxima camada tenha-a adquirido. A UC é responsável pela execução direcionada do

Figura 1: Árvore com a Hierarquia de Níveis do Sistema.



fluxo da RNA através de contadores e três máquinas de estado. Uma para gerenciar a entrada de dados e todo o processamento do sistema, duas no controle da função de ativação e elemento de processamento ou operações em ponto flutuante.

2.2 Unidade de Armazenamento

A unidade de armazenamento consiste de memórias do tipo BRAM embarcada e registradores baseados em área lógica da FPGA. A parametrização dos componentes embarcados segue a plataforma utilizada, geralmente é possível escolher dentre vários tipos e tecnologias distintas dependendo do fabricante e família da FPGA.

O sistema utilizou dois tipos distintos de memória BRAM, uma apenas de leitura (ROM) e outra para escrita dos dados (RAM). As duas tecnologias utilizam memória de porta simples para acessar os dados. A memória do tipo ROM está relacionada aos dados de entrada e a função de ativação. Na plataforma da Xilinx é carregado um arquivo inicial com extensão do tipo .coe. Os resultados da RNA são armazenados em memória RAM para permitir futuras transferências com barramentos. Em todas as memórias utiliza-se precisão simples (32 bits) nas palavras. Os dados de saída do sistema são armazenados em arquivo, por meio de simulação funcional, para a leitura e a comparação dos resultados no Matlab.

2.3 Unidade de Processamento

A unidade de processamento contém a soma ponderada entre as entradas e os pesos relacionados ao neurônio, conforme o padrão IEEE 754 de 32 bits. Para fazer esse processamento utiliza-se uma unidade de ponto flutuante baseado em um IP Core correspondente ao fabricante e com otimização de síntese para área. A unidade de processamento faz a computação do neurônio através de duas partes principais, função de ativação e elemento de processamento. O número desses componentes no sistema está diretamente relacionada a quantidade de entradas em cada neurônio, a quantidade de neurônios em determinada camada e a quantidade de camadas da RNA. Para cada entrada em um neurônio contém um elemento de processamento, assim se n é a quantidade de entradas a $Qtde_{PE} = 2^{n-1}$, e cada neurônio contém apenas uma função de ativação baseada na Tangente hiperbólica.

2.3.1 Elemento de Processamento

O elemento de processamento restringe-se a um multiplicador, um somador em ponto flutuante e componentes de controle dos dados entre níveis e módulos, conforme Figura 1.

A inserção de novos elementos de processamento depende da quantidade de entradas associadas ao neurônio. Para ter um maior controle no elemento de processamento foi necessário criar uma estrutura que organiza a inserção das entradas e saídas nos módulos de operações em ponto flutuante.

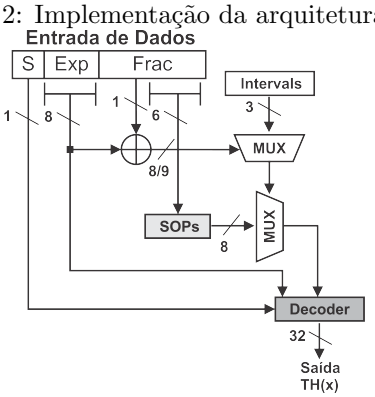
2.3.2 Função de Ativação

A função tangente hiperbólica corresponde a equação 2. A curva "S" da função é uma característica e nesse caso contém valores positivos e negativos, diferente da sigmoide (Logística) composta apenas de valores positivos. A variável x está relacionada a saída parcial do neurônio e λ a inclinação da tangente hiperbólica, assim a função de ativação realizará um filtro, conforme sua característica, inserindo a não linearidade na RNA.

$$Th(x) = \frac{1 - e^{-\lambda x}}{1 + e^{-\lambda x}} \tag{2}$$

Pelas dificuldades e grande variedade de técnicas na implementação da função de ativação com comportamento não linear, faz-se necessário a implementação de diversas arquiteturas para obter um limiar entre precisão, desempenho e área utilizada na FPGA. A arquitetura escolhida foi a híbrida de partes lineares com combinacional (HPC), conforme pesquisa realizada em Sartin and Silva (2013). A implementação utiliza a quantidade de elementos específicas em cada segmento e o mapeamento é feito por meio de lógica combinacional gerando a fração de resposta da função de ativação correspondente a saída da função. A quantidade de entradas e saídas do circuito combinacional está diretamente relacionado ao segmento. Assim, cada segmento contém uma soma de produtos (SOP) baseado em álgebra booleana e simplificado pelo algoritmo de Quine-Mccluskey. A arquitetura para a aproximação da função tangente hiperbólica é ilustrada pela Figura 2.

Figura 2: Implementação da arquitetura HPC.



3 Resultados

O sistema foi desenvolvido em linguagem de alto nível (Matlab), sem a toolbox, e em linguagem de descrição de hardware (VHDL), nas duas abordagens utilizando a mesma precisão de ponto flutuante simples (IEEE754). A função de ativação tangente hiperbólica foi baseada na "tansig" do Matlab é utilizada em todos neurônios da RNA nas três abordagens.

A análise dos resultados foi definida baseada em três principais partes do sistema a UC, UA e UP. Em cada parte será apresentada resultados específicos da implementação. A UC consiste principalmente pelas máquinas de estados para organizar a entrada e saída de dados e controle na execução do sistema. Uma importante observação é o tempo de execução de cada estado da máquina de estados finitos e o tempo total de execução da RNA. O tempo de execução em cada estado correspondente as camadas da RNA estão diretamente relacionados a UP e ao tempo total de execução.

A execução do sistema é realizada com 128 amostras, na Figura 6 pode-se observar o início da segunda amostra ou ciclo de execução. Em destaque, está a execução da primeira amostra que determina o tempo de execução em 3.12 us, e com 128 amostras o tempo de execução chega a aproximadamente 399 us.

Os dados coletados na RNA para análise de erro do sistema foram adquiridos pela simulação funcional. As entradas e os resultados finais da RNA são armazenados em memória dedicada, isso facilita posteriores implementações em tempo real na transferência de dados por um barramento de comunicação interligado ao sistema.

As diferenças encontradas na aproximação da função não linear foram semelhantes nos dois casos, no hardware e linguagem de alto nível. A implementação da RNA completa com todos os 128 pontos são apresentadas na Figura 3a, e nas Figuras 3b e 3c são reduzidas as quantidades de pontos para se observar a precisão do sistema de forma mais compreensível. Com a mesma linha de pensamento, os resultados na aproximação da função de ativação é apresentada pelas Figuras 4a, 4b e 4c.

A análise de erros absoluto, relativo e quadrático foi realizada para as implementações em baixo e alto níveis de abstrações e apresentadas na Figura 3, conforme as Equações 3, 4 e 5. Pelo erro relativo fica perceptível os pontos mais ao centro com os maiores erros de aproximação, Figura 5c, devido a proximidade de zero no erro absoluto. Porém, no erro absoluto mostra que todos os erros permanecem abaixo de 0,002, ou seja, uma precisão adequada para soluções de problemas não lineares.

$$E_{abs} = |f(x_k) - \hat{f}(x_k)| \tag{3}$$

$$E_{rel} = \frac{f(x_k) - \hat{f}(x_k)}{f(x_k)} \tag{4}$$

$$E_{quad} = \frac{(f(x_k) - \hat{f}(x_k))^2}{2} \tag{5}$$

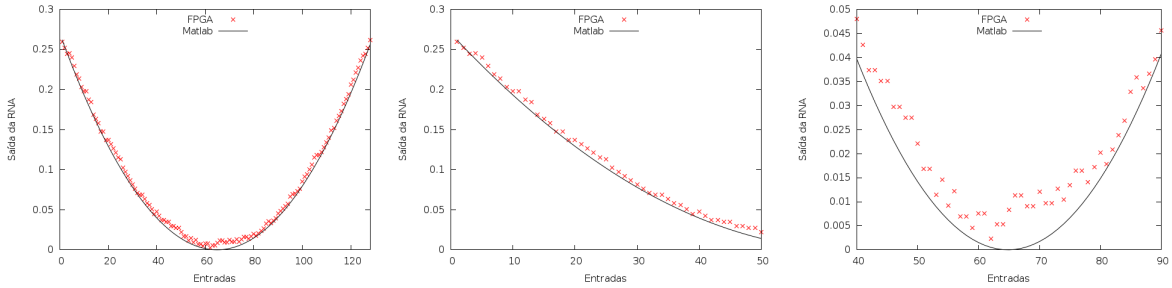
Nas Tabelas 1 e 2, apresentam-se os resultados finais da RNA, respectivamente, com os erros (médios e máximos) e os valores de área ocupada na FPGA. A média dos erros absoluto, relativo e médio quadrático é computada pela média aritmética simples e a quantidade de amostras analisadas na entrada da RNA (128) ou da função de ativação (500). O cálculo dos erros absoluto, relativo e médio quadrático foram truncados conforme a Equação (6), que corresponde ao erro máximo possível para determinada representação de dados (Tommiska, 2003). Assim, levou-se em consideração a fração (f), a quantidade de elementos nos segmentos, e as exceções *Overflow* e *Underflow*, conforme a Equação (6) de truncamento. A aproximação da função senoide obteve o erro máximo de $0,0051$ e o quadrático de $4,9x10^{-5}$ em Ferreira and Barros (2010).

$$E_{truncMax} = 2^{-(f+1)} \tag{6}$$

Na Tabela 2 são apresentados os valores de área utilizada na FPGA do Sistema da RNA completo e da prototipação apenas do módulo correspondente a função de ativação com o método Híbrido (HPC).

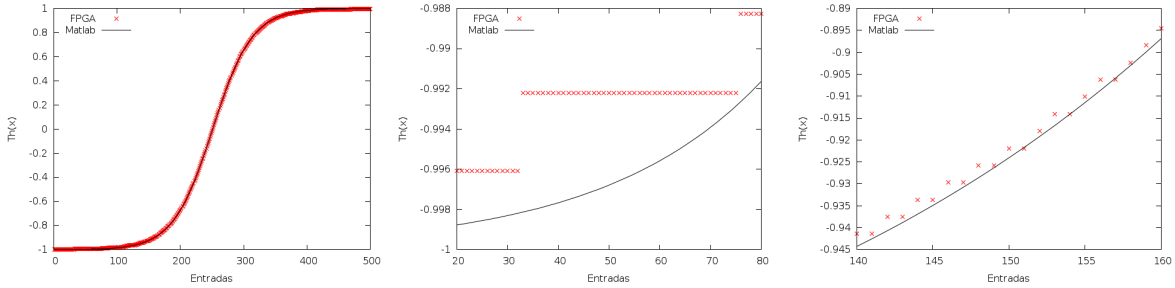
4 Conclusões

Neste trabalho apresentou-se a implementação de um sistema em ponto flutuante para aproximação de uma função não linear utilizando RNA. O sistema foi desenvolvido em baixo e alto nível de abstração para efeito de comparação, e com



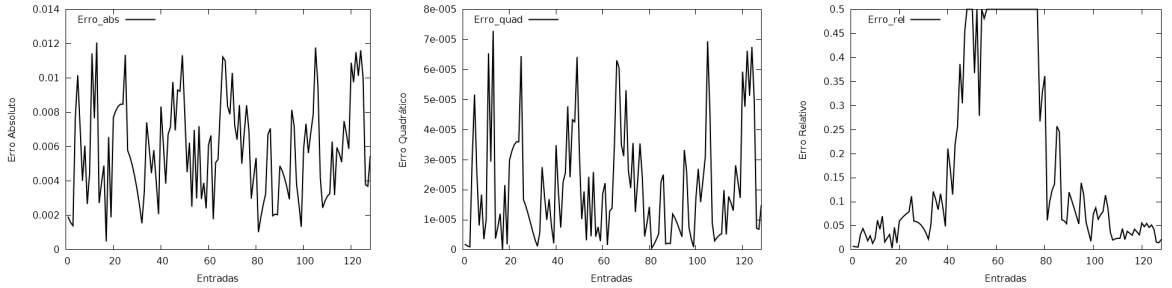
(a) 0 a 128 pontos. (b) 0 a 50 pontos (c) 40 a 90 pontos

Figura 3: Resultado das implementações em baixo e alto níveis de abstrações na aproximação da senóide



(a) 0 a 500 pontos. (b) 20 a 80 pontos (c) 140 a 160 pontos

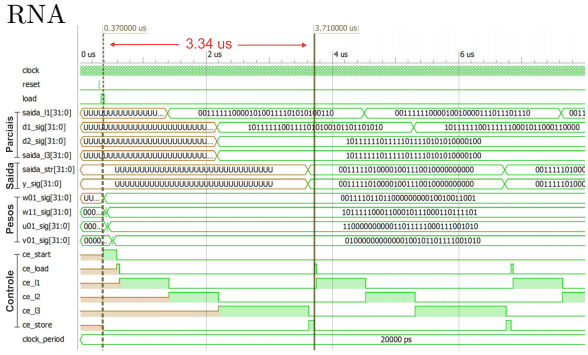
Figura 4: Resultado das implementações em baixo e alto níveis de abstrações na função de ativação



(a) Erro Absoluto. (b) Erro Quadrático (c) Erro Relativo

Figura 5: Resultado dos erros na aproximação da senóide

Figura 6: Simulação da Máquina de Estados da RNA



grande semelhança nos resultados na RNA e função ativação. A rede composta de 4 neurônios e função tangente hiperbólica em todos os neurônios

Tabela 1: Erros nas Implementações do Hardware

Implem.	Max_Abs	Med_Abs	Max_Rel	Med_Rel
Função de Ativação	$5,94 \times 10^{-3}$	$2,91 \times 10^{-3}$	$1,66 \times 10^{-2}$	$3,17 \times 10^{-3}$
Sistema da RNA	$6,25 \times 10^{-2}$	$7,79 \times 10^{-3}$	$4,81 \times 10^{-2}$	$1,02 \times 10^{-1}$

na FPGA foi adequada em prover alta precisão no sistema.

O emprego de FPGA para aplicações com RNA é adequado pelo nível de paralelismo previsto nas abordagens, o tempo de execução alcançado pela aplicação foi de aproximadamente 399 microssegundos (us) em 128 entradas. Porém, cálculos mais complexos junto ao uso de ponto flutuante é um desafio ao projetista no desenvolvimento de um sistema digital ou SoC (System On

Tabela 2: Área utilizada na FPGA.

Protipação	Slices	LUT-4in	BRAM	Multip
Função de Ativação	133(3%)	243(3%)	0	-
Sistema da RNA	4.567(98%)	5.340(57%)	1(5%)	20(100%)

Chip), pelas inúmeras possibilidades na construção de hardware. O projetista deve analisar as possibilidades de inserção ou a melhor escolha de componentes para determinada situação, além da melhor forma de desenvolvimento para unir duas características essenciais ao projeto do sistema, poupar área consumida da FPGA e não perder desempenho com isso.

A validação dos resultados com a análise de vários tipos de erros mostra a precisão do sistema da RNA completo e da função de ativação. A prototipação do sistema em uma plataforma FPGA com recursos limitados também acrescenta ao sistema uma adequada relação entre área e desempenho.

O sistema foi desenvolvido para proporcionar a portabilidade há outras plataformas de hardware, e facilitar a adaptação de novas variações relacionadas a quantidade de neurônios e de camadas na RNA. A comprovação da portabilidade foi realizada na implementação em outra plataforma FPGA.

Agradecimentos

Os autores gostariam de agradecer o suporte da UNEMAT (Universidade do Estado de Mato Grosso), PPGEE (Programa de Pós-Graduação em Engenharia Elétrica da UNESP), LPSSD (Laboratório de Processamento de Sinais e Sistemas Digitais), FAPEMAT (Fundação de Amparo à Pesquisa do Estado de Mato Grosso), CAPES e CNPQ processo (309023/2012-2).

Referências

Al-Kazzaz, S. and Khalil, R. (2008). FPGA implementation of artificial neurons, *Proceedings...*, International Conference on Information and Communication Technologies: From Theory to Applications - ICTTA, IEEE, Damascus, pp. 1–6.

Amin, H., Curtis, K. M. and Hayes-Gill, B. R. (1997). Piecewise linear approximation applied to nonlinear function of a neural network, *IEEE Proceedings Circuits, Devices and Systems* **144**(6): 313–317.

da Silva, R. M., Nedjah, N. and Mourelle, L. (2010). Arquitetura de hardware compacta e eficiente para redes neurais artificiais do tipo multiplas camadas, *Proceedings...*, Congresso Brasileiro de Automática - CBA, SBA, Bonito, pp. 2838–2845.

Ferreira, A. P. A. and Barros, E. N. S. (2010). A high performance full pipelined arquitecture of MLP neural networks in FPGA, *Proceedings...*, International Conference on Electronics, Circuits, and Systems - ICECS, IEEE, Athens, pp. 742–745.

Goldschmidt, R. R. (2010). *Uma Introdução à Inteligência Computacional*, IST-Rio, Rio de Janeiro.

Lee, Y. and Ko, S.-B. (2006). FPGA implementation of a face detector using neural networks, *Proceedings...*, Canadian Conference on Electrical and Computer Engineering - CCECE, IEEE, Ottawa, pp. 1914–1917.

Lin, C.-W. and Wang, J.-S. (2008). A digital circuit design of hyperbolic tangent sigmoid function for neural networks, *Proceedings...*, International Symposium on Circuits and Systems - ISCAS, IEEE, Beijing, pp. 856–859.

Meher, P. (2010). An optimized lookup-table for the evaluation of sigmoid function for artificial neural networks, *Proceedings...*, VLSI System on Chip Conference - VLSI-SoC, IEEE/IFIP, Madrid, pp. 91–95.

Santos, P., Ouellet-Poulin, D., Shapiro, D. and Bolic, M. (2011). Artificial neural network acceleration on FPGA using custom instruction, *Proceedings...*, Canadian Conference on Electrical and Computer Engineering - CCECE, IEEE Computer Society, Ontario, pp. 450–455.

Sartin, M. A. and Silva, A. C. R. (2013). Approximation of hyperbolic tangent activation function using hybrid methods, *Proceedings...*, International Workshop on Reconfigurable and Communication-Centric Systems-on-Chip - ReCoSoC, IEEE, Darmstadt, pp. 1–6.

Savich, A., Moussa, M. and Areibi, S. (2007). The impact of arithmetic representation on implementing MLP-BP on FPGAs, *IEEE Transactions on Neural Networks* **18**(1): 240–252.

Tommiska, M. (2003). Efficient digital implementation of the sigmoid function for reprogrammable logic, *IEE Proceedings - Computers and Digital Techniques* **150**(6): 403–411.

Zhang, J. and Chen, L. (2011). Hardware implementation of neural networks based on DSP48 slice of Virtex-4 FPGA, *Energy Procedia* **13**: 9548–9555.