# ARQUITETURA PARA APLICAÇÕES GENÉRICAS DE REDES NEURAIS ARTIFICIAIS COM FÁCIL CONFIGURAÇÃO DE TOPOLOGIAS *MULTILAYER PERCEPTRON* EM FPGA.

**PRADO[1], R. N. A., *SILVA[2], C. A. e A., OLIVEIRA[3], J. A. N., DORIA NETO[4], A. D., MELO[5], J. D.**

1- Instituto Federal do Rio Grande do Norte IFRN, 2,3,4 e 5 – Universidade Federal do Rio Grande do Norte UFRN
1- rafael.prado@ifrn.edu.br, 2 - carlos77.albuquerque@gmail.com, 3 - nicolau@ufrnet.br, 4 - adriao@dca.ufrn.br, 5 - jdmelo@dca.ufrn.br

**Resumo –** Este trabalho propõe uma arquitetura em *hardware*, descrita em VHDL, desenvolvida para embarque de redes neurais artificiais do tipo *Multilayer Perceptron* (MLP), com possibilidade de reuso de processadores neuronais (neurônios artificiais – *Perceptron*). Nessa arquitetura é possível configurar redes MLP com topologias diferentes, utilizando sempre uma mesma área de silício. Idealiza-se que, nessa arquitetura, as aplicações com RNA tenham facilidade no procedimento de embarque de uma rede neural MLP em *hardware*, bem como permitam fácil configuração de vários tipos de redes MLP em campo, com diferentes topologias (quantidade de neurônios e camadas variáveis e com diferentes funções de ativação). Uma rede de comunicação de propósito específico foi desenvolvida para possibilitar o reuso de neurônios artificiais com eficiência. A arquitetura proposta vislumbra que o operador não necessita conhecer o dispositivo internamente, nem, tampouco, ter conhecimento sobre linguagem VHDL. O dispositivo reconfigurável FPGA Cyclone® II foi o escolhido para simulações e testes nas aplicações com RNA.

**Palavras Chave –** Sistemas Embarcados, Redes Neurais Artificiais, Reuso, FPGA e VHDL.

## 1 Introdução

Os dispositivos em *hardware* para trabalho com Redes Neurais Artificiais atuam com o conceito de embarque em campo nas indústrias e em equipamentos portáteis de consumo; no entanto, ainda são poucas as pesquisas voltadas para essa área. Novas ferramentas para desenvolvimento de sistemas digitais reconfiguráveis e que permitem prototipagem em circuitos estão possibilitando a criação de novas arquiteturas, com reduzido tempo de desenvolvimento [15] [13]. O *hardware* reconfigurável *Field Programmable Gate Array* (FPGA) possibilita prototipagem de projetos descritos em linguagens de descrição de *hardware*, a exemplo do *Very High Speed Integrated Circuits Hardware Description Language* (VHDL) e do Verilog HDL, o que aperfeiçoa o processo de desenvolvimento dos projetos, além de permitir prototipagem em circuito digital para testes [11].

Este trabalho propõe uma arquitetura em *hardware*, descrita em VHDL, desenvolvida para embarque de redes neurais artificiais do tipo *Multilayer Perceptron* (MLP), com possibilidade de reuso de processadores neuronais (neurônios artificiais – *Perceptron*). Nessa arquitetura é possível configurar redes MLP com topologias diferentes de até 8.192 pesos sinápticos com a utilização de uma mesma área de silício. Para possibilitar o reuso de processadores neurais, uma rede de comunicação de propósito específico foi desenvolvida. Essa rede controlará as conexões entre os neurônios de camadas diferentes e realizará realimentação para propiciar o reuso de neurônios em uma topologia de rede MLP.

Pretende-se, com esta pesquisa, contribuir para que as RNA sejam vistas, também, como uma boa opção para embarque em campo, ou seja, colaborar para que as redes MLP possam ser testadas em problemas práticos dos mais variados tipos, comprovando, ou não, sua eficácia em problemas com *hardware* em campo. Outra necessidade identificada diz respeito à economia de energia e de espaços nas operações em campo, as quais, geralmente, utilizam dispositivos, às vezes instalados em locais de difícil acesso para manutenção e substituições, a exemplo de implantes médicos, dispositivos em poços profundos de petróleo etc [1] [8] [10].

O restante deste artigo está estruturado da seguinte forma: na Seção 2 serão referenciados alguns trabalhos que tratam de RNA em FPGA; na Seção 3 será feita uma breve descrição das redes neurais *Multilayer Perceptron* e de alguns tipos de funções de ativação, que serão usadas na arquitetura proposta; na Seção 4 será descrita a metodologia usada no desenvolvimento do projeto e as soluções utilizadas para sua implementação; na Seção 5 serão apresentados alguns resultados obtidos com a RNA implementada em FPGA e; finalmente, na Seção 6, serão apresentadas as conclusões do presente trabalho.

## 2 O Dispositivo FPGA e as Redes Neurais Artificiais

Algumas propostas de redes neurais para embarque foram desenvolvidas em FPGA. Algumas delas trabalharam a construção do neurônio como unidade básica de processamento e sua posterior replicação no projeto para, então, formar uma rede MLP [13]. Outros autores propuseram o uso de uma quantidade fixa de neurônios em *hardware* e a subsequente multiplexação dos resultados, para a reutilização das mesmas unidades neuronais desenvolvidas fisicamente, poupando a quantidade de blocos lógicos no FPGA [5].

Pode-se definir o projeto processando dados numéricos apenas em aritmética de ponto fixo; determina-se, assim, a quantidade de *bits* para representatividade do número, sem perder de vista o acúmulo de erros a serem gerados através dos

* Autor 2 – Agradece ao Programa de Recursos Humanos - ANP 14 pelo apoio financeiro ao projeto.

cálculos em ponto fixo, ao desprezarem-se os dados à direita da vírgula [5] [13]. Alternativamente, pode-se optar por trabalhar uma combinação híbrida de aritmética em ponto fixo e em ponto flutuante. Nesse contexto, são separados *bits* para definir mantissa, expoente, sinal e um *offset* fixo, o que permite a demonstração do valor em ponto flutuante [15].

Sistemas embarcados em um único *chip* possuem grande aceitação industrial. Nesse intuito, em [9] foi desenvolvido um conjunto de especialistas em uma única máquina de comitê, utilizando-se o processador Nios® II sintetizado em um FPGA. Tal proposta visa solucionar problemas de maior complexidade, para os quais são necessários mais de um sistema especialista avaliador.

DSPs e os FPGAs podem ser combinados no uso de sistemas de redes neurais com processamento paralelo e treinamento *online* da rede [6]. Outras combinações foram realizadas para alcançar o mesmo resultado, com o treinamento em *software* e a rede neural executada em *hardware* reconfigurável FPGA [13]. Projetos contemplaram redes do tipo *Radial Basis Function* (RBF) [7] e o uso de aritmética estocástica, na tentativa de simplificar a implementação em hardware e também de obter treinamento *online* (em *hardware*) [16].

## 3 Projeto de Redes Neurais Artificiais

Testes e simulações com as diferentes redes neurais existentes mostram suas diferentes funcionalidades diante do vasto campo de aplicação em que as RNA atuam. Redes do tipo MLP, por exemplo, possuem capacidade de generalização de alguns problemas a que se propõem, porém não garantem em todos. Essa rede é capaz de resolver determinados problemas de interpolação e extrapolação, além de possibilitar a classificação de padrões em vários problemas [3]. As deficiências das RNA estão geralmente associadas à demora no treino da rede e ao alto custo computacional e ao seus comportamentos como caixas pretas depois de treinadas, dentre outros aspectos [4]. Porém, a rede MLP é uma das RNA mais utilizadas, além de se revelar flexível na questão das múltiplas tarefas que podem ser por ela trabalhadas.

### 3.1 Redes *Perceptron* de Múltiplas Camadas (*Multilayer Perceptron* – MLP)

O *Perceptron* foi escolhido por ser o modelo de neurônio artificial mais conhecido, o modelo de McCulloch e Pitts (1943), de um neurônio [4]. Sua capacidade de aprendizado se dá através da solução de um problema de otimização. Nesse caso, ocorre a correção dos parâmetros livres que são os pesos sinápticos (W) e o *bias*. Esses parâmetros e o modelo matemático do *perceptron*, descrito em *DataFlow*, podem ser visualizados na Figura 1:

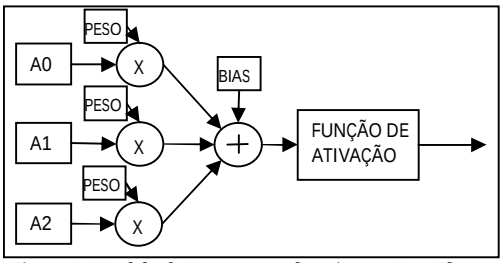


**Figura 1: Modelo do Perceptron descrito em DataFlow.**

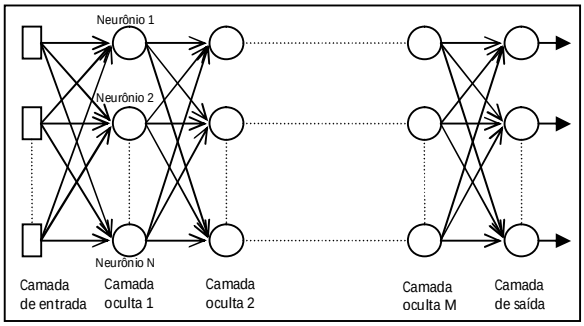


**Figura 2: Arquitetura Multilayer Perceptron descrito em DataFlow.**

Observa-se, na Figura 1, que o modelo matemático do *perceptron* é bem simples, constituído de uma função de transferência seguido de uma função de ativação. O produto escalar do vetor de entrada (A0, A1, A2) pelo vetor de parâmetros livres, os pesos, com o resultado desse produto somado ao último parâmetro, o *bias*, compõe a função de transferência. Por fim, o resultado da função de transferência é calculado em uma determinada função de ativação, como visualizado na Figura 1. Tal função pode ser do tipo linear (Equação 1), do tipo sigmoidal (Equação 2) ou do tipo tangente hiperbólica (Equação 3).

As redes MLP, alimentadas adiante (*feedfoward*), consistem em um conjunto de unidades sensoriais (sem processamento) que constituem a camada de entrada; um grupo de nós computacionais (*perceptrons*) compõe a camada oculta. A MLP pode conter um ou mais tipos de camada oculta e uma camada de saída também composta de nós computacionais. O sinal de entrada se propaga para a frente, através da rede, camada por camada. As redes MLP são constituídas de diversos *perceptrons* (nós computacionais) processando paralelamente, sob essa organização, em camadas, como mostradas na Figura 2.

Nessa arquitetura de rede, as variáveis livres (ajustáveis) são os pesos sinápticos que interligam os neurônios entre as camadas e os *biases* de cada *perceptron*. Ou seja, esses parâmetros são ajustáveis durante o treinamento, e, após, tornam-se fixos durante a execução das tarefas para as quais a rede foi previamente treinada.

## 4 Materiais e Métodos

Na proposta de desenvolver um dispositivo capaz de se reconfigurar em diversas arquiteturas de redes neurais do tipo MLP, a partir de instruções previamente conhecidas e ordenadas, inicialmente, determinou-se que instruções possibilitam descrever uma rede MLP qualquer. Tais instruções contem informações sobre a quantidade de camadas, quantidade de neurônio por camada e sobre como os neurônios serão interconectados. Para desenvolver um sistema baseado em instruções específicas, foi utilizada a metodologia de desenvolvimento *Design* RTL, que separa parte operativa (*Datapath*) e unidade controladora (controlador). O problema foi modelado com a utilização de ferramentas de especificação, como o *Data Flow* e o *State Chart*.

As instruções visualizadas no Quadro 1 descrevem o tipo da camada (entrada, oculta ou saída), a quantidade de neurônios em cada camada e o tipo de função de ativação nos neurônios de cada camada (linear, sigmóide ou tangente hiperbólica).

| INSTRUÇÃO | OPERANDO 1 | OPERANDO 2 | EXEMPLO |
|---|---|---|---|
| ENTRADA | Quantidade de neurônios (nós) de entrada | Não possui. | ENTRADA 2 (dois nós de entrada) |
| OCULTA | Quantidade de neurônios na camada oculta. | Define a função de ativação para todos os neurônios da camada oculta (linear, sigmóide ou tangente hiperbólica). | OCULTA 5, 1 ('5' neurônios nessa camada oculta, '1' significa que a função de ativação será a tangente sigmóide) |
| SAIDA | Quantidade de neurônios de saída. | Define a função de ativação para todos os neurônios da camada de saída (linear, sigmóide ou tangente hiperbólica). | SAIDA 1, 0 ('1' neurônio na camada de saída, '0' significa que a função de ativação será a linear pura) |

**Quadro 1: Descrição das instruções executadas pela arquitetura.**

Nessas instruções os operandos indicam qual função de ativação será utilizada em cada camada e também a quantidade de neurônios existentes em cada camada. Caso o número de neurônios ultrapasse a quantidade suportada pelo sistema, a unidade controladora será capaz de reutilizá-los várias vezes, armazenando os resultados em memória e substituindo os pesos e *biases* dos neurônios, completando, assim, a quantidade desses elementos solicitada no operando, ou seja, na camada que necessitar de mais neurônios. A determinação de qual a configuração da rede que interconectará os neurônios atuais com os registradores que contêm os resultados dos processamentos da camada anterior é outra tarefa exercida pelo controlador.

Para a entrada dessas instruções e dos pesos e *bias* que serão carregados na memória da rede, algumas sugestões de Interface Homem Máquina (IHM) e de alocação das instruções e pesos nas memórias do dispositivo FPGA serão dadas ao final desta seção.

### 4.1 O *Perceptron* Descrito em VHDL

Tendo como referência o neurônio proposto por McCulloch e Pitts, representado na Figura 1, subseção 3.1, a arquitetura do neurônio proposta nesse trabalho seguiu o modelo de implementação feito em VHDL em [13] [4], mostrado na Figura 3. O NEURONIO foi dividido em dois blocos funcionais. O primeiro bloco é um combinador linear, responsável pelo somatório das entradas ponderadas pelos pesos sinápticos; e o segundo é um bloco responsável pelo cálculo da função de ativação, denominados, respectivamente de blocos NET e FNET.

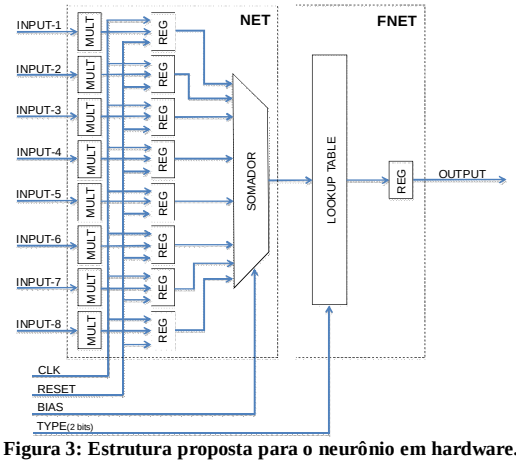


**Figura 3: Estrutura proposta para o neurônio em hardware.**

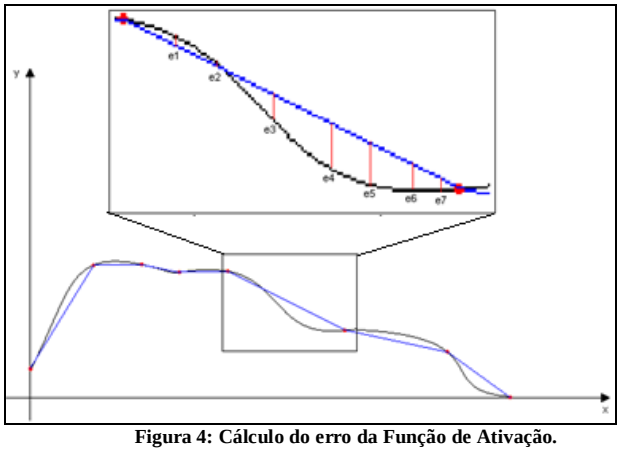


**Figura 4: Cálculo do erro da Função de Ativação.**

O bloco NET mostrado na Figura 3, apresenta um fluxo de dados (*data flow*) com uma unidade denominada MULTIPLICADOR e outra denominada SOMADOR. A construção desse bloco se baseou em uma abordagem de RTL design (projeto em nível de transferência entre registros), com a inclusão de registradores para a sincronia dos fluxos de dados. O bloco NET calcula o campo local induzido do *perceptron* com até 8 entradas de 16 *bits* que lhes são impostas.

Para manter o paralelismo, cada entrada do neurônio possui um multiplicador exclusivo para realizar o produto com o peso sináptico, ambos definidos em ponto fixo com dezesseis *bits* e com sinal. No MULTIPLICADOR é feito o deslocamento de dezesseis *bits* para manter a compatibilidade com a representação de ponto fixo dos dados do sistema. O SOMADOR é a unidade responsável pela soma dos resultados das multiplicações e do *bias*.

Para a conclusão do bloco NEURONIO, após o cálculo do campo local induzido, prossegue-se com o cálculo da função de ativação, no bloco FNET; pode-se optar pelo uso das funções de ativação sigmóide, linear pura e tangente sigmóide, calculadas como descrito a seguir.

## 4.2 Cálculo da Função de Ativação

A implementação da função de ativação do tipo sigmóide ou tangente sigmóide em FPGA é realizada com a utilização de uma *lookup table* (LUT), cuja estrutura é constituída de um bloco comparador e de duas ROMs paralelas de 16 x 21 *bits* de dados. O motivo de se escolher uma *lookup table* para simular a função tangente sigmóide está relacionado ao custo e à dificuldade de se implementá-la matematicamente em FPGA. A função de ativação aplicada se constitui em uma tabela de 21 pontos com valores previamente definidos [17].

Para a definição desses valores, a solução adotada foi representar a função por meio de um conjunto de pontos interpolados linearmente, de tal modo que a diferença entre a curva da função e a curva dos pontos seja mínima. Para isso, utilizou-se uma técnica de inteligência computacional conhecida como algoritmo genético, que tem a finalidade de minimizar o erro de cada indivíduo com base em uma função-objetivo.

Após a execução do algoritmo genético, obtêm-se os 21 pontos da tabela, correspondentes aos valores de estrada x da função tangente sigmóide e suas respectivas saídas y. Assim, são obtidos o coeficiente angular e o coeficiente linear, armazenados nas ROM da unidade LUT.

Nesses termos, o cálculo do valor de saída do bloco FNET é executado da seguinte maneira: a partir do valor de entrada, proveniente do bloco NET, é definido um endereço da LUT comum às duas ROM, nas quais estão presentes os correspondentes coeficientes angular e linear do segmento de reta, a serem usados pelo INTERPOLADOR, para a geração do sinal de saída.

## 4.3 Rede de Comunicação com o Propósito Específico do Reuso de Processadores Neurais (Interconexão de Neurônios)

Os autores [13] optaram por não trabalhar com nenhum tipo de rede de interconexão, e sim com conexões fixas, enquanto que outros estudos, com características de camadas fixas, optaram por utilizar alguma das redes conhecidas e comumente usadas [2]. Na proposição feita neste trabalho, como a quantidade de neurônios não será fixa, e sim variável, de acordo com as mudanças de camada e em conformidade com as definições de redes previamente embarcadas, optou-se por desenvolver uma rede de interconexão própria.

Nesta nova proposição foram usados componentes digitais para permitir a comunicação ordenada entre os neurônios. Essa rede de comunicação realiza a interconexão da saída de cada neurônio à entrada do próximo(s) neurônio(s) da próxima camada, lembrando que os neurônios da próxima camada serão representados pelos próprios neurônios da camada atual. A rede de comunicação promoverá uma realimentação das informações processadas pelos neurônios de uma camada para a outra sucessora, ressalvando-se que, a cada camada, substituem-se os valores dos pesos e *biases* nos neurônios que serão reutilizados, para que representem uma nova camada de uma rede neural MLP. A solução proposta pode ser claramente visualizada na Figura 5.

Na Rede de Comunicação com realimentação proposta, o processo é bem simples. Todas as conexões necessárias para conectar as saídas de todos os neurônios de uma camada "n" às entradas de todos os neurônios de uma camada "n+1" são realizadas simultaneamente, através de um conjunto registrador-drive (Drive) de três estados, que irão determinar uma conexão de dados previamente armazenados ou uma ausência de dados, numa conexão em alta impedância. A decisão de que conexões serão ativadas (conectadas as entradas dos neurônios ao registrador que os antecede) ou desativadas (colocadas em alta impedância) será feita pela unidade de controle e repassada à rede de comunicação, através de uma unidade decodificadora mostrada na Figura 5, existem conexões entre a unidade decodificadora e os drives, porém não está sendo mostrado na figura. Enfim, o Quadro 2 relaciona a entrada da unidade decodificadora à sua saída. As saídas da unidade decodificadora estão conectadas aos registradores-drives, ativando ou desativando cada drive individualmente.

| Entrada | Drive 1 | Drive 2 | Drive 3 | Drive 4 | Drive 5 | Drive 6 | Drive 7 | Drive 8 | Drive 9 |
|---|---|---|---|---|---|---|---|---|---|
| 0101 | ativo | inativo | inativo | Inativo | inativo | inativo | Inativo | inativo | inativo |
| 0110 | ativo | ativo | inativo | Inativo | inativo | inativo | Inativo | inativo | inativo |
| 0111 | ativo | ativo | ativo | inativo | inativo | inativo | Inativo | inativo | inativo |
| 1001 | ativo | inativo | inativo | ativo | inativo | inativo | Inativo | inativo | inativo |
| 1010 | ativo | ativo | inativo | ativo | ativo | inativo | Inativo | inativo | inativo |
| 1011 | ativo | ativo | ativo | ativo | ativo | ativo | Inativo | inativo | inativo |
| 1101 | ativo | inativo | inativo | ativo | inativo | inativo | Ativo | inativo | inativo |
| 1110 | ativo | ativo | inativo | ativo | ativo | inativo | Ativo | Ativo | inativo |
| 1111 | ativo | Ativo | ativo | ativo | ativo | ativo | Ativo | Ativo | ativo |

**Quadro 2 – Funcionamento da unidade decodificadora com o resultado das saídas, a partir das entradas possíveis.**

Quando, hipoteticamente, se deseja conectar uma camada de dois neurônios com outros dois neurônios de uma próxima camada, a entrada adequada para a unidade decodificadora configurar os *drives* seria o código "1010" (coluna um, linha seis do Quadro 2), onde a unidade ativará os *drives* um, dois, quatro e cinco, conectando o registrador (que armazena temporariamente o resultado do processamento da camada anterior) aos neurônios um e dois, como pode ser visualizado na Figura 5. Para qualquer valor diferente dos mencionados no Quadro 2, a rede tornará todos os drives inativos.

| Rede | Crossbar | Multiest. | Proposta | Bus |
|---|---|---|---|---|
| **Clocks** | 2 | 2 | 2 | Muitos |
| **Drives** | 18x16 | 36x16 | 9x16 | 2x16 |
| **Mux** | 18 | 9 | 9 | 0 |
| **Regs** | 0 | 0 | 3 | 0 |

**Quadro 3 – Comparativo das redes de interconexão.**

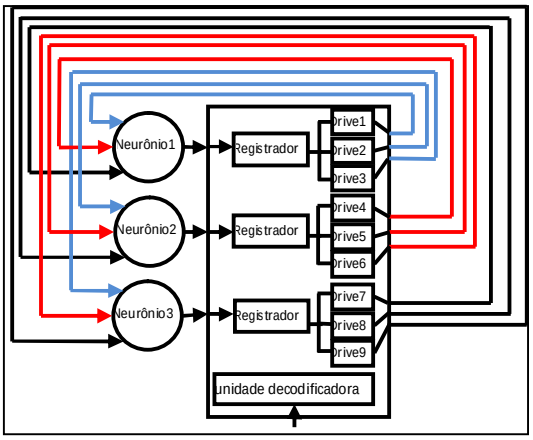


**Figura 5: Rede de comunicação proposta.**

A partir da análise do Quadro 3, nota-se que a rede de comunicação aqui proposta tem excelentes características quando comparada às redes já existentes, principalmente nos quesitos de redução de *drives* e de multiplexadores, no entanto, utiliza-se de registradores para armazenar o resultado da camada de neurônios executada anteriormente.

## 4.4 *Datapath* – Parte Operativa

Foram definidos quatro unidades NEURONIO, com oito entradas cada uma, para compor o *datapath*; eles serão reutilizados para executar várias camadas e também para executar uma camada com mais de quatro neurônios. No tocante à execução de uma camada com mais neurônios do que os quatro existentes na arquitetura, a reutilização dos neurônios do sistema é comandada pelo controlador do sistema, de forma a executar as quatro unidades NEURONIO quantas vezes necessário. Faz-se a troca dos pesos, das entradas e dos *biases*, para a execução de cada quatro neurônios diferentes, até toda a quantidade de neurônios exigida em uma mesma camada ser comtemplada, quando esta contiver mais que quatro NEURONIOS. Foi necessário inserir uma memória RAM de 16 *bits* e 64 endereços para armazenar os resultados temporários de cada neurônio a ser utilizado como entrada na próxima camada.

Na Figura 6 visualiza-se a arquitetura do *datapath* da arquitetura proposta e a organização do fluxo de dados entre os componentes, o sinal do controlador está indicado pelo "Sinal MdE".

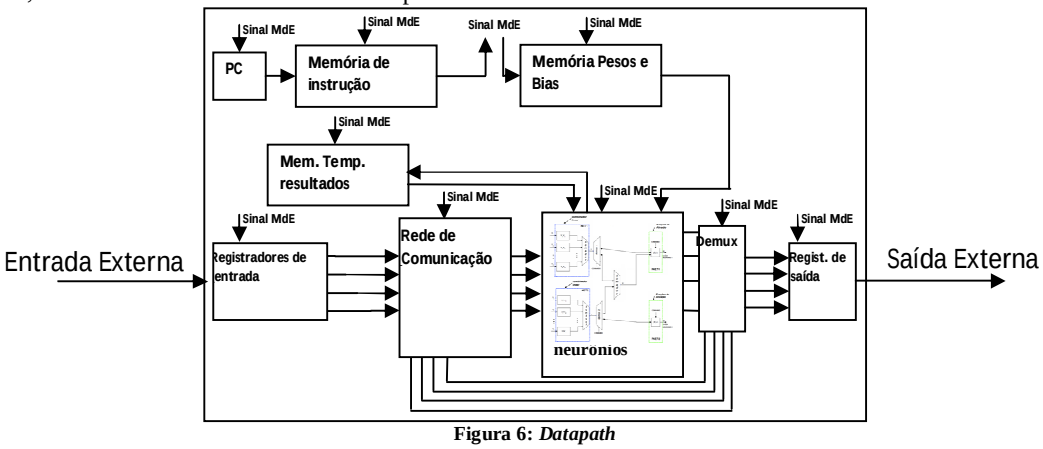


**Figura 6: *Datapath***

O *datapath* exibe a forma como os componentes do sistema estão interconectados. Observa-se que todos eles são controlados por um sinal vindo do controlador. Tanto a ação que será executada por cada componente interno como o seu momento de execução serão determinados pela Máquina de Estados Finita, MdE, a qual mantém o sincronismo dos componentes para que os dados trafeguem corretamente entre eles. O sinal de *clock* está conectado a todos os componentes, com exceção do demultiplexador (Demux), o que não está sendo mostrado na Figura 6.

Ainda na Figura 6, pode-se destacar que a saída do contador de programa (PC) está conectada à entrada de endereços da memória de instruções. A saída da memória de instrução conecta-se ao controlador, para que ele determine quais ações executar. E há registradores de entrada e saída para controlar o fluxo externo de dados, de entrada e saída.

## 4.5 Controlador

Na MdE foi trabalhado o conceito de *State-Chart*, para possibilitar hierarquia e facilitar a construção do controlador. A visualização gráfica está na Figura 9.

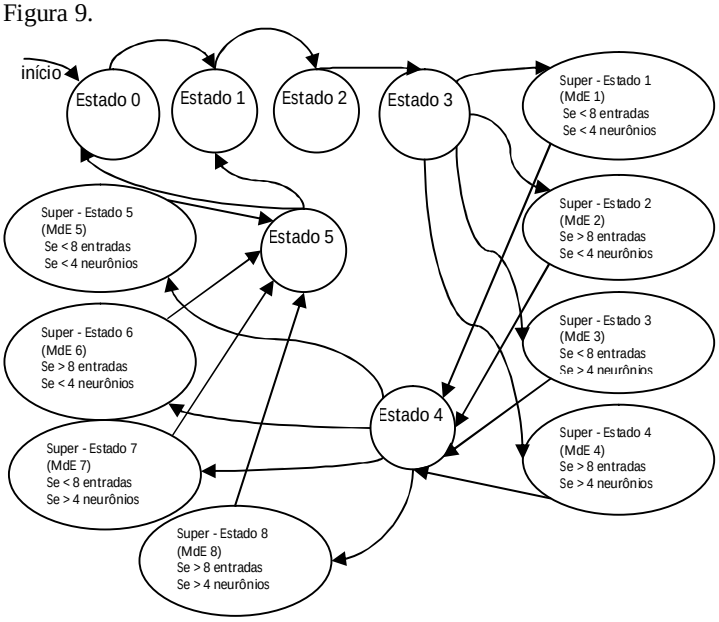


**Figura 9: Controlador do Sistema.**

Na MdE observam-se os super-estados (do um ao oito), os quais representam outras MdE (identificadas como MdE1 a MdE8). Os estados 0, 1, 2 e 3 inicializam os componentes sistema, leem instrução, inicializam entradas e definem condicionais para os próximos super-estados. Os super-estados 1, 2, 3 e 4 servem para executar camadas ocultas em quatro condições diferentes, com mais que quatro neurônios ou, menos que quatro neurônios, e com mais que oito entradas em cada neurônio ou menos que isso. O mesmo acontece com os super-estados 4 a 8, só que cada um executará um tipo diferente de camada de saída, dependendo do número de neurônios e de entradas.

O estado 5 finaliza a execução da rede MLP definida nas instruções e para o primeiro conjunto de entradas. Caso o sistema seja resetado, irá para o estado 0 (zero); caso o sistema execute a mesma topologia MLP, ele irá para o estado 1. O estado 4 representa um estado com avaliações condicionais, determinando qual será o próximo estado, dentre os super-estados cinco ao oito.

## 4.6 Exemplo de Funcionamento da Arquitetura

Com essa estrutura, é possível expandir a quantidade de entradas por neurônio (acima de oito), sem que seja necessário o uso dos somadores, de tal forma que se utilizou a rede de comunicação para direcionar os resultados dos blocos NET, procedendo-se à realimentação, para aproveitar os somadores já existentes nos blocos NET. As Figuras 7 e 8 abaixo irão explicitar este funcionamento. No exemplo à seguir, os blocos com linhas tracejadas definem que estão em operação e possuem o fluxo de dados.

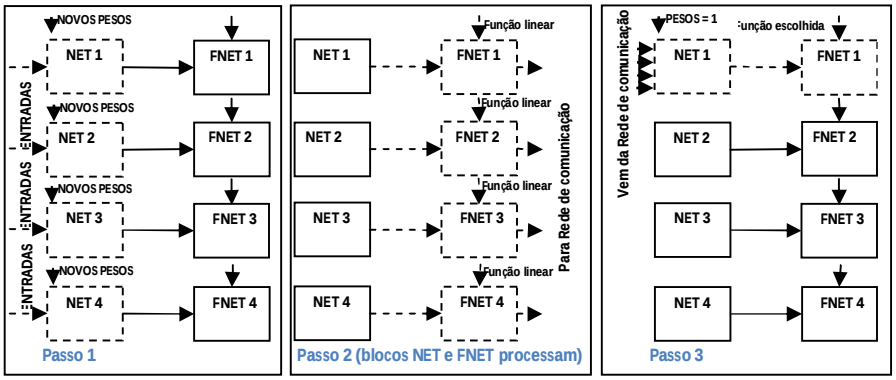


**Figura 7: Passos 1, 2 e 3 do exemplo de funcionamento da arquitetura.**

Na tentativa de elucidar mais a finalidade da otimização realizada na unidade NEURONIO, exemplifica-se a execução de um neurônio com 32 entradas, ressaltando-se que as unidades NET possuem apenas 8 entradas cada. A Figura 7, Passo 1, visualiza-se este Passo, no qual as 32 entradas estão sendo inseridas nas quatro unidades NET (8 em cada unidade). Cada unidade NET terá o valor dos pesos relativos ao de um mesmo neurônio, com 32 pesos (oito em cada bloco NET); dessa forma, um neurônio de 32 entradas será executado por quatro blocos NET conjuntamente. O valor dos *biases* de cada NET será preenchido pelo valor 0 (elemento neutro da adição). A Figura 7b exibe o processamento do bloco FNET, com a função de ativação sempre do tipo linear, para que, na saída de cada bloco FNET, haja quatro somatórios ponderados vindos das 32 entradas; o resultado segue para a rede de comunicação.

Na Figura 7, o Passo 3 descreve o preenchimento dos pesos com valor 1 (elemento neutro da multiplicação), para que seja feita apenas a soma das quatro entradas vindas da rede (resultado do processamento dos passos 1 e 2). O valor do bias será o valor constante na memória, destinado ao neurônio de 32 entradas em questão. A entrada de dados vinda da rede de comunicação (os resultados do processamento dos passos 1 e 2), e, por fim, o processamento dos blocos NET e FNET nessas condições, esse último configurado para executar a função de ativação definida na instrução do neurônio com 32 entradas.

## 5 Resultados e Discussões

Todos os resultados foram extraídos a partir de simulações e testes com a utilização o *software* Quartus® II da Altera® e das informações de síntese do circuito em um FPGA de referência EP2C35F672C6, Cyclone® II.

## 5.1 Resultados da Síntese da Arquitetura com Quatros Neurônios de Oito Entradas

O Quadro 5 demonstra o comparativo entre as taxas de ocupação de três arquiteturas no FPGA. A arquitetura proposta por [13] (coluna 3) é composta de 5 neurônios de 2 entradas na camada oculta e 1 neurônio de 5 entradas na camada de saída. Arquitetura proposta por [5] possui uma camada oculta com 5 neurônios de 8 entradas e uma de saída com 3 neurônios de 5 entradas (coluna 4). A arquitetura aqui proposta (coluna 2) é composta de 4 neurônios com 8 entradas e tem possibilidade de execução de redes MLP com até 8.192 pesos e com, no máximo, 32 neurônios de 32 entradas em uma mesma camada, através do sistema de realimentação e reuso. O número de camadas da arquitetura proposta é limitado pelo número máximo de pesos sinápticos

| **Arquiteturas Comparadas** | Arquitetura proposta neste trabalho | Arquitetura [13] | Arquitetura [5] |
|---|---|---|---|
| Dispositivo | **EP2C70F896C6** | **EP2C35F672C6** | **XCV400hq 240** |
| Elementos lógicos | **7975/68.416 (12%)** | **4936 / 33.216 (15%)** | **8018 / 19.200 (42%)** |
| Total de registradores | **936 /68.416 (< 2%)** | **768/33.216 (< 4%)** | ------- |
| Total de pinos | **583 / 622 (94 %)** | **876 / 475 (184 %)** | ------- |
| Número de *bits* de memória | **0 / 1.152.000 (0%)** | **0 / 483.840 (0%)** | **2/10 (20%)(BRAM)** |
| Multiplicadores dedicados de 9 *bits* | **72 / 300 (24%)** | **42 / 70 (60%)** | ------ |
| Frequência do *clock* | **69,92 MHz** | **54,16 MHz** | **73 MHz** |

**Quadro 5: Resultados da síntese da arquitetura inicial, comparada a outros trabalhos.**

O Quadro 5 exibe um comparativo das características da síntese realizada em FPGA da arquitetura aqui proposta, e das demais arquiteturas problematizadas nesta pesquisa. A frequência máxima de trabalho que o dispositivo desenvolvido pode suportar é da ordem de 69,92 MHz, um bom índice quando comparado ao das arquiteturas desenvolvidas por outros autores e exibidas no Quadro 5. As informações sobre freqüência máxima de trabalho foi extraída do software Quartus ® II da Altera®.

Outro ponto a ser observado é a análise energética do dispositivo desenvolvido, orientado sobre os custos que a implementação de uma RNA do tipo MLP em *hardware* terá. O total de energia térmica dissipada pelo dispositivo é de 242.34 mW. Essa energia é dissipada em parte, pelos *drives* de entrada e saída do dispositivo em 87,23mW, e a outra parte acontece através de uma dissipação de energia térmica estática no núcleo (*core*). O máximo de corrente drenada pela arquitetura será de 155,03 mA, internamente, e de 17,06 mA nos pinos de entrada e saída. Esses dados foram todos registrados a partir de análise realizada pelo *software* Quartus® II da Altera®.

## 5.2 Resultados de Simulações com Aplicações em RNA

Serão mostrados os resultados do processamento de redes MLP para resolver os problemas da interpolação da função *Sinc*, interpolação da função exponencial e de um problema de classificação de cinco padrões. Todas essas redes MLP foram previamente treinadas e testadas em *software*.

Na Figura 10, visualiza-se o comparativo da função *Sinc* com as implementações realizadas em *software* e na arquitetura proposta. A rede MLP utilizada possui um neurônio na camada oculta e um na de saída, ambos com função de ativação tangente hiperbólica. Os neurônios da camada oculta possuem três entradas, pois o treinamento foi realizado utilizando-se três conjuntos de entradas e um de saída: a primeira entrada com os dados da função *Sinc*; a segunda entrada com os dados na função *Sinc* com atraso e a última entrada com a realimentação da saída com atraso da rede.

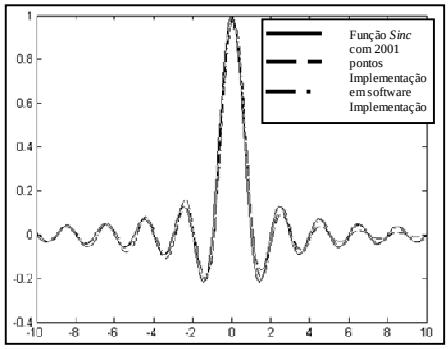


**Figura 10: Comparativo função *Sinc*.**

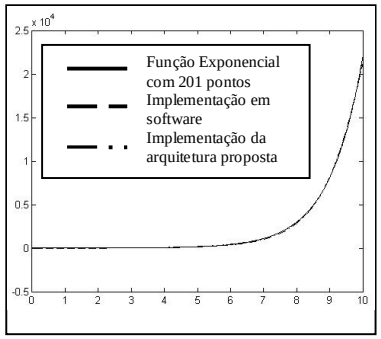


**Figura 11: Comparativo função exponencial.**

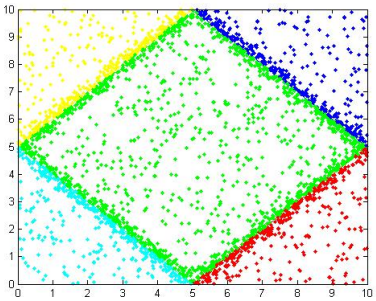

**Figura 12: Plano de com 5 regiões.**

Na Figura 11, observa-se o comparativo da função exponencial (novamente) com as implementações em software e na arquitetura proposta. A rede MLP utilizada possui um neurônio na camada oculta e um na de saída, ambas com função de ativação tangente hiperbólica. Os neurônios da camada oculta possuem uma entrada apenas.

O problema seguinte aplicado foi a classificação de cinco regiões em um plano cartesiano, no qual as regiões formam um losango dentro de um quadrado. Neste problema, as entradas são os dados cartesianos "x" e "y", que definirão um ponto no plano dividido, nas cinco regiões; a rede neural tem de ser capaz de identificar em qual região das cinco regiões do plano está localizado o ponto de entrada. A topologia da rede neural utilizada para solucionar tal problema apresenta dois nós de entrada, uma camada oculta com doze neurônios e cinco neurônios na camada de saída. Cada neurônio na camada de saída sinaliza uma determinada classe que identifica a região na qual o ponto de entrada se localiza.

A Figura 12 representa as cinco regiões nesse plano, separadas por cinco diferentes cores. Os pontos visualizados são que foram usados para o treinamento da rede em software.

Os resultados de classificação desses cinco padrões pela rede neural MLP, em software, obteve 99,1% de acerto. Foi utilizado um conjunto de pontos de entrada que são considerados mais difíceis ou críticos de identificar (pontos situados em regiões de fronteira entre classes). O sistema aqui proposto, que trabalha com representação numérica em ponto fixo e com aproximação por partes da função de ativação, obteve classificação igual à obtida em *software*, com 99,1% de acerto dos pontos testados.

## 6 Conclusão

Um dos objetivos deste projeto foi o de criar uma máquina, em *hardware*, capaz de ser reprogramada em campo, por meio de instruções bem simples, as quais possibilitassem que o operador não necessitasse conhecer o dispositivo internamente, nem tampouco ter conhecimento sobre linguagem VHDL, ou sobre a estrutura em *hardware* contido no sistema. Tal objetivo foi alcançado com sucesso, obtendo-se um dispositivo robusto no quesito flexibilidade de reconfiguração (variedade de topologias de redes MLP) e que consegue manter paralelismo no processo. Este dispositivo configura-se a partir de apenas três instruções, no mínimo (uma para configurar os nós de entrada, outra para a camada de saída e outra para a camada oculta, a necessidade de mais instruções seriam para configurar mais camadas ocultas), e, no máximo, 32, devido à capacidade da memória utilizada, possuindo boa relação custo-benefício nos quesitos economia de área de silício e velocidade.

Com uma memória ROM de 16 kBytes, é possível armazenar 8.192 pesos e *biases* de uma mesma rede. Assim, o sistema é capaz de configurar uma rede com 256 neurônios de 32 entradas paralelas ou uma rede de 1.024 neurônios de 8 entradas paralelas. Porém, a velocidade diminuirá quando utilizados muitos neurônios, o mesmo ocorrendo em *software*.

Uma rede de comunicação de propósito específico para RNA foi desenvolvida, proporcionou reuso dos componentes, mantendo-se uma redução na área de silício consumida e o sincronismo da RNA. É importante notar que a arquitetura é fixa e utiliza sempre a mesma quantidade de elementos lógicos em qualquer aplicação em que for usada. Sua modularidade está no fato de se reconfigurar a partir de instruções, sempre com o mesmo *hardware*.

## 7 Referências

[1] CHEN, P. SHIE, M. C. ZHENG, Z. Y. ZHENG, Z. F. Chu, C. Y. **A Fully Digital Time-Domain Smart Temperature Sensor Realized With 140 FPGA Logic Elements**. IEEE Transactions On Circuits And Systems—I: Regular Papers, Vol. 54, No. 12, p 2661-2668, December 2007.

[2] DIAS F. M., ANTUNES A., MOTA A. M. **Artificial Neural Networks: a Review of Commercial Hardware.** 2004. Journal Engineering Applications of Artificial Intelligence. Vol. 17, Issue 8, December, 2004.

[3] GONZÁLEZ, M. A. OLVERA. TANG, Y. **Black-Box Identification of a Class of Nonlinear Systems by a Recurrent Neurofuzzy Network.** IEEE Transactions On Neural Networks, Vol. 21, No. 4, April 2010.

[4] HAYKIN, S. **Redes neurais: princípios e práticas**. 2. ed. Porto Alegre: Bookman, 900 p, 2001.

[5] HIMAVATHI, S. ANITHA, D. MUTHURAMALINGAM, A. **Feedforward Neural Network Implementation in FPGA Using Layer Multiplexing for Effective Resource Utilization.** IEEE Transactions On Neural Networks, Vol. 18, No. 3, May 2007.

[6] KIM, S. S. JUG, S. **Hardware Implementation of a Real Time Neural Network Controller with a DSP and an FPGA**. Proceedings of the 2004 IEEE International Conference on Robotics 8 Automation New Orleans, LA Apt 41, 2004.

[7] KUNG, Y. S. QUANG, N. K. ANH, L. T. V. **FPGA-Based Neural Fuzzy Controller Design for PMLSM Drive.** International Journal of Power Electronics, Vol. 3, No.3 pp. 320 - 333 2011.

[8] LEINER, B. J. LORENA, V. Q. CESAR, T. M. LORENZO, M. V. **Hardware Architecture for FPGA Implementation of a Neural Network and its Application in Images Processing**. Proceeding CERMA '08 - Proceedings of the 2008 Electronics, Robotics and Automotive Mechanics Conference, ISBN: 978-0-7695-3320-9, 2008.

[9] LOPES, D. C. MELO, J. D. **Implementação e Avaliação de Máquinas de Comitê em um Ambiente com Múltiplos Processadores Embarcados em um Único Chip.** 2009. Disponível em:<http://ppgeec.ufrn.br>. Acesso em: 15 dez. 2010.

[10] OCUSSE, T. A. D. PEREIRA, A. S. MARRANGHELL, N. **Microcalcification Border Characterization - Using Wavelets on Digital Mammograms**. IEEE Engineering In Medicine And Biology Magazine 0739-5175 IEEE. September/October, pág. 41-43, 2009.

[11] PASERO, E. PERRI, M. **Hw-Sw Codesign of a Flexible Neural Controller Through a FPGA-based Neural Network Programmed in VHDL.** Proceedings. 2004 IEEE International Joint Conference on Neural Networks, Vol. 4, D.O.I. 10.1109/IJCNN, pág. 3161 - 3165, 2004.

[12] PEARSON, M. J. PIPE, A. G. MITCHINSON, B. GURNEY, K. MELHUISH, C. GILHESPY, I. NIBOUCHE, M. **Implementing Spiking Neural Networks for Real-Time Signal-Processing and Control Applications: A Model-Validated FPGA Approach.** IEEE Transactions On Neural Networks, Vol. 18, No. 5, September 2007.

[13] SILVA C. A. e A., DORIA NETO A. D., OLIVEIRA J. A. N., MELO J. D. (2009). **Implementação de uma Rede Neural Artificial em FPGA:Aplicação da MLP como Arquitetura Modular.** Anais do IX Congresso Brasileiro de Redes Neurais Inteligência Computacional (IX CBRN). Ouro Preto 25-28 de Outubro de 2009.

[14] TAFNER, M. A. **As Redes Neurais Artificiais: aprendizado e plasticidade.** Revista "Cérebro & Mente" 2(5), Março/Maio 1998.

[15] WIIST, H. KASPER, K. REININGER, H. **Hybrid Number Representation for the FPGA-Realization of a Versatile Neuro-Processor.** Euromicro Conference, 1998. Proceedings 24th. Vol. 2, D.O.I 10.1109/EURMIC, pág 694 - 701, 1998.

[16] ZHANG, D. SIMON, H. L. FOO, Y. **A Simplified FPGA Implementation of Neural Network Algorithms Integrated with Stochastic Theory for Power Electronics Applications. Industrial Electronics Society**. Industrial Electronics Society, 2005 - IECON 2005. 31st Annual Conference of IEEE, pág 6, Issue 6-6 Nov, 2005.

[17] SILVA, D. R. C. **Redes Neurais Artificiais no ambiente de redes industriais foundation fieldbus usando blocos funcionais padrões.** 2005. 67 f. Dissertação (Mestrado) - Departamento de Engenharia da Computação, Universidade Federal do Rio Grande do Norte - UFRN, Natal, 2005.