REDES NEURAIS DE HOPFIELD PARA ROTEAMENTO DE REDES DE COMUNICAÇÃO EM FPGA

Trabalho de Conclusão de Curso

Engenharia de Computação

Aluno: Marcos Antonio da Cunha Oliveira Junior

Orientador: Prof. Dr. Carmelo José Albanez Bastos Filho

UNIVERSIDADE
DE PERNAMBUCO

Marcos Antonio da Cunha Oliveira Junior

Redes Neurais de Hopfield para Roteamento de Redes de Comunicação em FPGA

Monografia apresentada para obtenção do Grau de Bacharel em Engenharia de Computação pela Universidade de Pernambuco

Orientador:
Prof. Dr. Carmelo José Albanez Bastos Filho

GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO
ESCOLA POLITÉCNICA DE PERNAMBUCO
UNIVERSIDADE DE PERNAMBUCO

Recife - PE, Brasil

Maio de 2011

De acordo

Recife

/ /

Prof. Dr. Carmelo Bastos-Filho
Orientador

Resumo

As redes de comunicação têm seu desempenho drasticamente impactado pelos seus algoritmos de roteamento. Para que toda capacidade da rede seja usufruída, torna-se necessário o desenvolvimento de algoritmos de roteamento que otimizem o uso de seus recursos.

Inteligência Computacional é um termo utilizado para representar um conjunto de técnicas que apresentam habilidade de aprender e/ou lidar com novas situações. Entre essas técnicas, há as Redes Neurais Artificiais que são baseadas nas redes neurais do cérebro humano e que têm a capacidade de aprendizado e de tomar decisões.

As Redes Neurais de Hopfield são um tipo de rede neural recorrente que podem ser utilizadas para roteamento. Elas se mostram como uma ótima alternativa para o processo de roteamento, pois agregam a característica de aprendizado a ele. Contudo, apesar de sua eficácia, as redes neurais não são rápidas na resposta a uma requisição de roteamento como os algoritmos tradicionais. Por outro lado, graças ao comportamento paralelo presente nas redes neurais, elas são apropriadas para implementações em plataformas paralelas.

Matrizes de Blocos Lógicos Programáveis em Campo (FPGA, em inglês, *Field Programmable Gate Array*) são dispositivos lógicos programáveis organizados em uma matriz bi-dimensional de células lógicas e conexões programáveis. Devido a essa organização, FPGA é uma plataforma inerentemente paralela possibilitando a implementação de algoritmos essencialmente paralelos nela.

Dados o atributo de paralelismo presente nas redes neurais e a característica paralela do FPGA, a adaptação do processo de roteamento nesta plataforma pode levar a desempenhos comparáveis aos algoritmos tradicionais. Além disso, a vantagem da rede neural ser adaptativa pode ser colocada em prática em ambientes reais.

Neste trabalho, o modelo proposto para a rede neural de Hopfield em FPGA foi validado demostrando ser possível sua implementação em dispositivos FPGA. Particulamente, a simplificação necessária para a codificação da função de ativação utilizada pela rede neural em FPGA não traz erros significativos ao algoritmo de roteamento.

Ademais, o modelo proposto simulado é 78 vezes mais rápido do que a versão sequêncial da rede neural de Hopfield em um computador comum. Além disso, o *speed-up* alcançado com relação a uma versão paralela em GPU é de aproximadamente 15.

Abstract

The routing algorithms influence drastically on the computer networks perfomance. Therefore, good routing algorithm must be created in order to optimize their resources and thus fit to the network needs.

Computational Intelligence is a set of techniques with an ability to learn and to deal with new situations. Among these techniques, Neural Networks are inspired by the brain and by the biological neurons. They have the ability of learning and decision-making.

Hopfield Neural Networks are a kind of Neural Networks with feedback that may be used to route computer networks. Once they are adaptive, they are an interesting option to be used as routing algorithm in order to handle the dynamic behaviour presented in computer networks.

However, although they work well to solve the routing problem, they are still slower than the other approaches used nowadays. In the other hand, the neural networks are inherently parallel, once the neurons perform their operations individually. Therefore, they are suited to be implemented on parallel platforms in order to improve their performance.

Field Programmable Gate Arrays (FPGA) are programable logic devices organized in a bi-dimensional matrix of logic cells. The architecture presented in the FPGAs turns them inherently parallel, thus they may be used to implement parallel algorithms.

Therefore, thanks the parallel behaviour of the neural networks and the parallel architecture of the FPGA, the routing algorithm based on Hopfield Networks running on FPGA may have perfomance similar to the standard algorithms used nowadays. Furthermore, the benefit of the adaptiveness of the neural networks may be used in real computer networks.

In this project, it is proposed a Hopfield Neural Networks model for FPGA. It is shown that the model is valid and then it is possible to have the routing algorithm running on FPGAs. Moreover, it is also shown that aproximations of the activation function can be used by the model without lack of perfomance.

Furthermore, the proposed model is almost 78 times faster than a sequential version of the Hopfield Neural Networks. Moreover, it has a speed-up of 15 when it is compared with a parallel version of the HNN running on GPUs (Graphic Processing Units).

Agradecimentos

Não faz muito sentido ao final de todo o projeto, agradecer apenas àqueles que contribuíram diretamente. Ou ainda, citar aqueles que indiretamente contribuíram de alguma forma no momento de elaboração do projeto.

Um trabalho de conclusão começa antes dos cinco anos de engenharia. E, por isso, são muitas pessoas que deveriam aparecer aqui. Não por ajudas certeiras mas por, de alguma forma, influenciarem no modo de pensar ou no modo de agir que, agora, refletem ao final do curso.

Ainda assim, seriam alguns capítulos no trabalho explicando, ou tentando explicar, como pessoas do mundo real ajudaram (e ajudam) na conclusão de um curso de engenharia.

E, mesmo assim, muitas dessas pessoas provavelmente jamais entenderiam que influenciaram em algo tão distante. O que, na realidade, creio, ser um dos motivos dessa influência.

Infelizmente, *our normal approach is useless here* e creio que jamais conseguiria explicar tudo.

Entretanto, devo agradecer a algumas constantes dentro do meu curso de engenharia, agradeço aos três caras que suportaram o meu humor durante os cinco anos; agradeço ao meu orientador Carmelo que acreditou fielmente que esse projeto ficaria pronto com palavras de apoio (“Acelere") nos momentos de caos; e agradeço a minha família por encarar com naturalidade imitações esquizofrênicas do Silvio Santos quando o projeto funcionou.

E o trabalho continua.

“Como um pássaro o tempo voa
A procura do exato momento
Onde o que você pode fazer fosse agora
Com as roupas sujas de lama
Porque o barro arrudeia o mundo
E a TV não tem olhos pra ver
Eu sou como aquele boneco
Que apareceu no dia na fogueira
E controla seu próprio satélite
Andando por cima da terra
Conquistando o seu próprio espaço
É onde você pode estar agora”

Chico Science & Nação Zumbi – Um Satélite na Cabeça

Sumário

Lista de Figuras p. xi

Lista de Tabelas p. xiii

Lista de Algoritmos p. xiv

Lista de Abreviaturas e Siglas p. xv

1 Introdução p. 1

2 Redes Neurais de Hopfield p. 3

2.1 Redes Neurais Artificiais . . . . . p. 3

2.1.1 Neurônio Biológico . . . . . p. 4

2.1.2 Neurônio Artificial . . . . . p. 4

2.2 Redes Neurais de Hopfield . . . . . p. 5

2.3 Redes Neurais de Hopfield para Roteamento de Redes de Comunicação . . . . . p. 7

3 Matriz de Blocos Lógicos Programáveis em Campo p. 11

3.1 Dispositivos de lógica programável . . . . . p. 11

3.1.1 Matriz de Blocos Lógicos Programáveis em Campo . . . . . p. 12

3.2 Linguagem de descrição de *hardware* . . . . . p. 13

3.2.1 VHDL . . . . . p. 14

3.2.2 Síntese . . . . . p. 15

4 Proposta e Arranjo Experimental p. 17

4.1 Paralelismo das Redes Neurais . . . . . p. 17

4.1.1 Modelo paralelo de uma Rede Neural de Hopfield para roteamento de uma rede de comunicação . . . . . p. 18

4.1.2 Particularidades no Código . . . . . p. 20

4.2 Precisão Numérica nas Redes Neurais . . . . . p. 21

4.2.1 Representação Numérica . . . . . p. 22

4.2.2 Função de Ativação . . . . . p. 22

4.3 Arranjo Experimental . . . . . p. 24

5 Resultados e Discussão p. 27

5.1 Validação . . . . . p. 28

5.2 Custo da Redução da Complexidade da Função de Atualização dos Neurônios . . . . . p. 29

5.3 Custo das Aproximações da Função de Ativação . . . . . p. 30

5.4 Tempo para Convergência . . . . . p. 31

6 Conclusões e Trabalhos Futuros p. 32

Referências Bibliográficas p. 34

Anexo A – Código VHDL do Modelo da Rede Neural de Hopfield p. 37

Lista de Figuras

2.1 A estrutura típica de um neurônio biológico. . . . . p. 4

2.2 A estrutura do neurônio artificial de McCulloch e Pitts. . . . . p. 4

2.3 A topologia básica de uma Rede Neural de Hopfield. . . . . p. 6

3.1 Arquitetura básica de um PLD. . . . . p. 12

3.2 Arquitetura básica de um FPGA. . . . . p. 13

3.3 Estrutura de um CLB. . . . . p. 13

3.4 Os vários níveis presente em VHDL. . . . . p. 15

3.5 Etapas da implementação de um circuito em FPGA. . . . . p. 15

4.1 Máquina de estados de um neurônio modelado em FPGA. . . . . p. 18

4.2 Máquina de estados da rede neural modelada em FPGA. . . . . p. 19

4.3 Bloco conceitual com as saídas e entradas de um neurônio de uma rede neural de Hopfield para roteamento de uma rede de 4 nós. . . . p. 20

4.4 Definição das saídas e entradas em VHDL de um neurônio de uma rede neural de Hopfield para roteamento de uma rede de 4 nós. . . . p. 20

4.5 Blocos de neurônios interconectados de uma rede neural com 12 nós. p. 20

4.6 Estrtutura da representação numerica utilizando ponto fixo. . . . . p. 22

4.7 A função sigmóide logística com $\lambda = 1$. . . . . p. 23

4.8 Aproximação da função sigmóide a partir de uma tabela de consulta. p. 23

4.9 Aproximação da função sigmóide a partir de uma interpolação. . . . p. 23

4.10 Rede de comunicação com quatro nós utilizada na modelagem da arquitetura de rede neural de Hopfield para roteamento. . . . . p. 25

4.11 Rede de comunicação com cinco nós utilizada na modelagem da arquitetura de rede neural de Hopfield para roteamento. . . . . p. 25

4.12 Rede de comunicação com seis nós utilizada na modelagem da arquitetura de rede neural de Hopfield para roteamento. . . . . p. 26

5.1 Resultado da simulação da rede neural de Hopfield para roteamento de uma rede de comunicação de quatro nós no simulador do Quartus II. . . . . p. 28

5.2 Caminho encontrado pela rede neural de Hopfield para a requisiçao Nó origem 0 e Nó destino 3 em uma rede de quatro nós. . . . . p. 28

5.3 Resultado da simulação da rede neural de Hopfield para roteamento de uma rede de comunicação de 5 nós no simulador do Quartus II. . p. 29

5.4 Caminho encontrado pela rede neural de Hopfield para a requisiçao Nó origem 0 e Nó destino 4 em uma rede de cinco nós. . . . . p. 29

5.5 Resultado da simulação da rede neural de Hopfield para roteamento de uma rede de comunicação de seis nós no simulador do Quartus II. p. 29

5.6 Caminho encontrado pela rede neural de Hopfield para a requisiçao Nó origem 0 e Nó destino 3 em uma rede de seis nós. . . . . p. 29

Lista de Tabelas

4.1 Resumo dos estados de um neurônio no modelo proposto da rede neural de Hopfield. . . . . p. 19

4.2 Tabela com os valores atribuídos aos 17 intervalos para a aproximação da função sigmóide logística. . . . . p. 24

4.3 Tabela com os valores atribuídos aos 17 intervalos para a aproximação da função sigmóide logística. . . . . p. 24

4.4 Valores dos parâmetros da rede neural de Hopfield para roteamento de redes de comunicação. . . . . p. 25

5.1 Recursos utilizados no FPGA por cada abordagem de atualização de neurônios. . . . . p. 30

5.2 Recursos utilizados no FPGA por cada abordagem de aproximação da função sigmóide. . . . . p. 30

5.3 Tempo médio necessário para a rede neural de Hopfield para roteamento de redes de comunicaçao convergir. . . . . p. 31

Lista de Algoritmos

1 Pseudocódigo do algoritmo de roteamento utilizando Redes Neurais de Hopfield. . . . . p. 9

Lista de Abreviaturas e Siglas

ASIC – *Application-Specific Integrated Circuit*
CI – *Computational Intelligence*
CLB – *Confgurable Logic Block*
CPLD – *Complex Programmable Logic Device*
EDA – *Electronic Design Automation*
EPROM – *Erasable Programmable Read-Only Memory*
FPGA – *Field Programmable Gate Array*
GPU – *Graphic Processing Unit*
HDL – *Hardware Description Language*
HNN – *Hopfield Neural Network*
IEEE – *Institute of Electrical and Electronics Engineers*
MCP – Neurônio de *McCulloch e Pitts*
PAL – *Programmable Array Logic*
PLD – *Programmable Logic Device*
PROM – *Programmable Read-Only Memory*
RNA – Redes Neurais Artificiais
VHDL – *Very high-speed integrated circuit Hardware Description Language*
VLSI – *Very-Large-Scale Integration*

1 Introdução

"*I think the problem, to be quite honest with you, is that*
you've never actually known what the question is.
So, once you know what the question actually is,
you'll know what the answer means. "
Deep Thought* para *Loonquawl
em "O Guia do Mochileiro das Galáxias".

O crescimento da Internet e das aplicações multimídia que necessitam de altas taxas de transmissão de dados implicam a necessidade de redes de comunicação com banda de alta capacidade. Os algoritmos de roteamento utilizados pelas redes têm um grande impacto no desempenho destas redes. Desta forma, para que toda sua capacidade seja usufruída, torna-se necessário o desenvolvimento de algoritmos de roteamento que otimizem o uso dos recursos da rede.

Inteligência Computacional (CI, do inglês, *Computational Intelligence*) é um termo utilizado para representar um conjunto de técnicas que apresentam habilidade de aprender e/ou lidar com novas situações [1]. Como exemplo de técnicas, pode-se citar: Redes Neurais Artificiais, Computação Evolucionária, Algoritmos baseados em Inteligência de Enxame, Lógica Nebulosa, etc. As Redes Neurais Artificiais são inspiradas nas redes neurais do cérebro humano e têm a capacidade de aprendizado e de tomar decisões.

Redes Neurais de Hopfield (HNN, do inglês, *Hopfield Neural Networks*) são um tipo de redes neurais recorrentes que podem ser utilizadas para roteamento [2, 3]. O seu uso vai além do roteamento de redes eletrônicas, sendo possível também sua aplicação em redes ópticas [4, 5, 6, 7, 8].

As redes neurais se mostram como uma ótima alternativa para o processo de roteamento, pois além de terem resultados eficazes, elas agregam a característica de aprendizado ao algoritmo de roteamento. Este é um atributo bastante desejável

para o roteamento, uma vez que as redes de comunicação são ambientes sujeitos a constantes mudanças. Contudo, apesar de sua eficácia, as redes neurais não são rápidas para responder a uma requisição de roteamento como outros algoritmos de roteamento, como por exemplo o algoritmo de Dijkstra [9].

Por outro lado, graças ao comportamento paralelo presente nas redes neurais devido aos seus neurônios trabalharem individualmente, elas são apropriadas para implementações em plataformas paralelas. Esta otimização no processo de roteamento com as redes neurais de Hopfield já foi demonstrada ser alcançável em dispositivos massivamente paralelos, como em Unidades de Processamento Gráfico (GPUs, do inglês, *Graphic Processing Units*) [10].

Matrizes de Blocos Lógicos Programáveis em Campo (FPGA, em inglês, *Field Programmable Gate Array*) são dispositivos lógicos programáveis organizados em uma matriz bi-dimensional de células lógicas e conexões programáveis. Devido a essa organização, FPGA é uma plataforma inerentemente paralela possibilitando a implementação de algoritmos essencialmente paralelos nela.

Dados o atributo de paralelismo presente nas redes neurais e a característica paralela do FPGA, a adaptação do processo de roteamento nesta plataforma pode levar a desempenhos comparáveis aos algoritmos tradicionais. Além disso, a vantagem da rede neural ser adaptativa pode ser colocada em prática em ambientes reais.

Este projeto propõe um modelo paralelo para as Redes Neurais de Hopfield utilizadas no roteamento de redes de comunicação em FPGA. As dificuldades encontradas para implementar a rede neural na arquitetura, como a precisão numérica e a codificação da função de ativação, também são abordadas no trabalho.

No Capítulo 2 e no Capítulo 3, são apresentadas visões gerais sobre as Redes Neurais de Hopfield para Rotamento de Redes de Comunicaçao e sobre os dispositivos FPGA, respectivamente. A motivação e a proposta do modelo paralelo da rede neural em FPGA são descritas no Capítulo 4. Os resultados das simulações do modelo são resumidos no Capítulo 5. E, por fim, no Capítulo 6, são apresentadas as conclusões e considerações finais.

2 *Redes Neurais de Hopfield*

"If the human brain were so simple that we could understand it, we would be so simple that we couldn't. "

– Emerson M. Pugh.

As Redes Neurais Artificiais (RNAs) são inspiradas nas redes neurais do cérebro humano. Este modelo matemático tem a capacidade de aprendizado e de classificação. Um dos tipos de redes neurais recorrentes, denominado Redes Neurais de Hopfield, pode ser utilizado para solucionar o problema de roteamento em uma redes de comunicação.

Neste capítulo, as Redes Neurais Artificiais são apresentadas na Seção 2.1, as Redes Neurais de Hopfield são descritas na Seção 2.2 e o algoritmo de roteamento baseado nesse tipo de rede é exposto na Seçao 2.3.

2.1 Redes Neurais Artificiais

Uma rede neural é um processador maciçamente paralelamente distribuído constituído de processamento simples, que tem a propensão natural para armazenar conhecimento experimental e torná-lo disponível para o uso [11]. Elas são capazes de aprender e de tomar decisões baseadas em sua aprendizagem.

A utilização das redes neurais é uma forma de incorporar características presentes no cérebro humano – alto grau de paralelismo, tolerância à falhas, robustez, capacidade de adaptação, auto-organização, entre outras – na computação convencional.

2.1.1 Neurônio Biológico

Um neurônio é uma célula do sistema nervoso presente no cérebro cuja principal função é coletar, processar e disseminar sinais elétricos [12]. O cérebro humano contém entre 70 e 90 bilhões de neurônios e cada um pode ter em torno de 10.000 ou mais sinapses [13]. A Figura 2.1 mostra o diagrama esquemático de um neurônio típico.

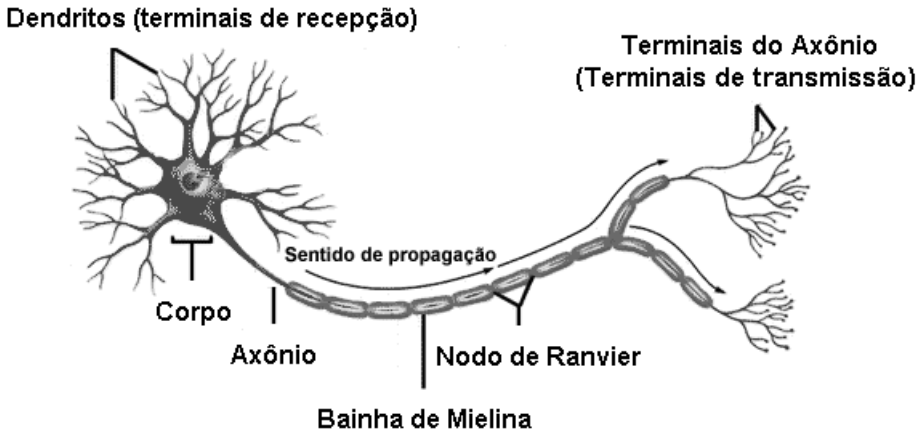

Figura 2.1: A estrutura típica de um neurônio biológico.

O neurônio consiste basicamente de corpo celular, axônio (e seus terminais) e dendritos. Os dendritos são os principais receptores de sinais neurais para comunicação entre neurônios. O axônio é o canal pelo qual há a condução de mensagem para os terminais pré-sinápticos, onde cada neurônio está em contato sináptico com outros neurônios. O corpo celular é o local de processamento das informações recebidas pelos dendritos, no qual um impulso nervoso no axônio é criado a partir de uma reação eletroquímica baseada em suas entradas [13].

2.1.2 Neurônio Artificial

A era moderna das redes neurais começou com o trabalho pioneiro de McCulloch e Pitts em 1943 [14]. Warren McCulloch e Walter Pitts – um neuroanatomista e um matemático, respectivamente – descreveram um modelo de cálculo lógico das redes neurais que unificava os estudos da neurofisiologia e da lógica matemática [11]. A partir de um de neurônio com seu disparo seguindo a lei de “tudo ou nada", criaram o modelo ilustrado na Figura 2.2.

Neste modelo, conhecido como modelo McCulloch-Pitts ou modelo MCP, os den-

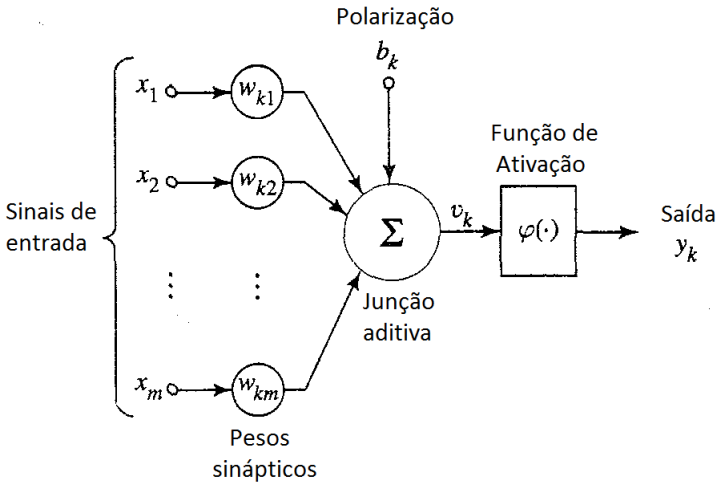

Figura 2.2: A estrutura do neurônio artificial de McCulloch e Pitts.

dritos do neurônio biológico *k* são representados pelas *m* entradas (x_i) e o axônio é representado pela saída (y_k). As sinapses nervosas são representadas por um peso (w_{ki}) associado a cada entrada, podendo este ser excitatório (valor positivo) ou inibitório (valor negativo). A saída do neurônio é o resultado da aplicação da função de ativação sobre a soma entre o somatório dos sinais de entradas ponderados pelos seus respectivos pesos sinápticos (w_k) e a polarização externa (*bias*) b_k que por sua vez tem o efeito de aumentar ou diminuir a polarização na função de ativação. O processo é descrito pelas seguintes equações:

$$u_k = \sum_{i=1}^{m} x_i w_{ki} \tag{2.1}$$

e

$$y_k = f(u_k + b_k), \tag{2.2}$$

onde $f(.)$ é a função de ativação do neurônio.

Os tipos de funções de ativação – responsável pelo limiar de disparo do neurônio – mais utilizados são: função Degrau, função Linear, função Logística e função Tangente Hiperbólica [15].

2.2 Redes Neurais de Hopfield

As Redes Neurais de Hopfield (HNN, do inglês *Hopfield Neural Networks*) são redes neurais artificiais que apresentam realimentação desenvolvidas por John Hop-

field [2]. O trabalho de Hopfield – publicado em 1982 – ajudou no ressurgimento do interesse pela área de redes neurais que passava por um período de crise após a publicação de Minsky e Papert na qual mostrava que o *Perceptron*[1] só resolvia problemas linearmente separáveis [17].

As redes de Hopfield podem ser utilizadas como memórias associativas, na qual a rede é capaz de armazenar informações baseada em alguns de seus estados [11]. Além disso, elas podem resolver problemas de otimização combinatória. Neste caso, a rede de Hopfield possui uma função de energia que provê uma medida de desempenho para o problema de otimização que possui um conjunto grande, mas finito, de possíveis soluções [15].

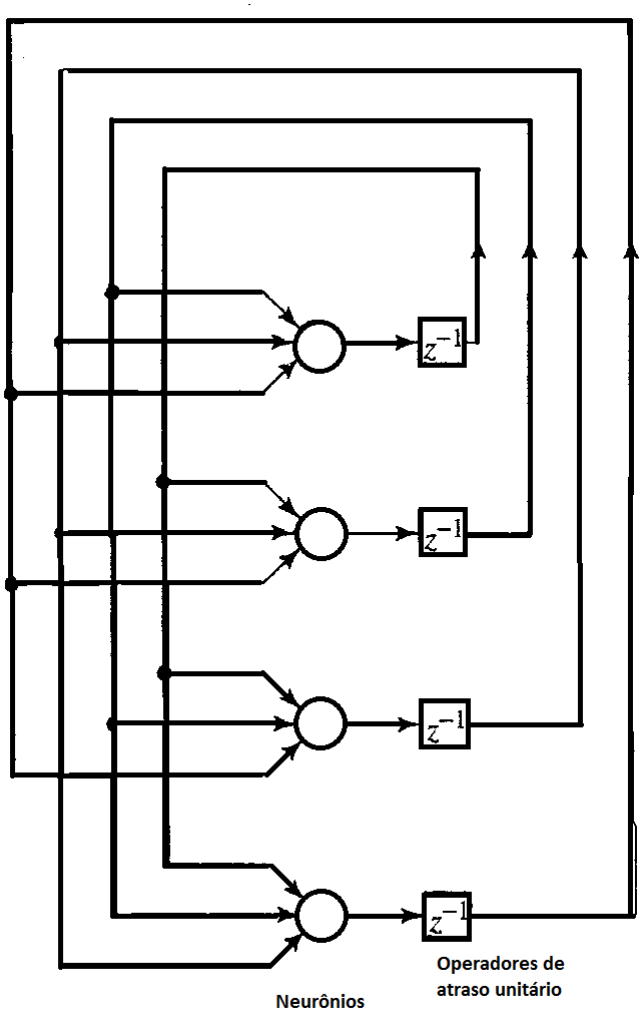

Figura 2.3: A topologia básica de uma Rede Neural de Hopfield.

O modelo de Hopfield consiste em um conjunto de neurônios e um conjunto corres-

[1] *Perceptron* é um modelo simples de rede neural no qual várias unidades de processamento estão ligadas a uma única saída [16].

pondente de atrasos unitários, formando um sistema realimentado de múltiplos laços. Como ilustrado na Figura 2.3, a saída de cada neurônio é realimentada, atráves de um elemento de atraso unitário, para cada um dos outros neurônios da rede (não há auto-realimentação) [11].

As redes neurais de Hopfield são formadas por neurônios de McCulloch-Pitts [14]. Desta forma, cada neurônio possui um somatório do conjunto de entradas ponderadas U_i, uma saída V_i que é o resultado da aplicação de uma função de ativação sobre a entrada U_i. A saída V_i após um atraso de tempo, é aplicada à entrada dos outros neurônios ponderada por um peso sináptico T_{ij} somado a uma polarização externa (*bias*) I_i. As saídas dos neurônios podem ser calculadas usando a Equação (2.3).

$$V_i = g_i(U_i) = \frac{1}{1+e^{-\lambda_i U_i}}. \tag{2.3}$$

A dinâmica da rede de Hopfield de atualização da entrada do neurônio i (U_i) é descrita pela equação:

$$\frac{dU_i}{dt} = -\frac{U_i}{\tau} + \sum_{j=1}^{n} T_{ij}V_j + I_i, \tag{2.4}$$

onde τ é a constante de tempo [18].

A dinâmica de qualquer sistema do tipo Hopfield com uma matriz de conexão simétrica é governada por uma função de energia definida por [18, 19]:

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} T_{ij}V_iV_j - \sum_{i=1}^{N} I_iV_i. \tag{2.5}$$

O segundo e o terceiro termo da Equação (2.4) representam a variação de energia da rede de Hopfield. Assim, a dinâmica do *i*-ésimo neurônio da rede de Hopfield pode ser descrita em termos da função de energia:

$$\frac{dU_i}{dt} = -\frac{U_i}{\tau} - \frac{\partial E}{\partial V_i}. \tag{2.6}$$

2.3 Redes Neurais de Hopfield para Roteamento de Redes de Comunicação

A utilização das Redes Neurais de Hopfield para solucionar o problema do menor caminho a partir de um determinado par origem-destino foi iniciado por Rauch e Winarske [3]. Essa primeira abordagem tem como um dos parâmetros o número de nós que formam o caminho a ser encontrado. Zang e Thomopoulos retiraram essa limitação, porém introduziram a necessidade de alteração na topologia da rede neural a cada requisição origem-destino [20]. Mais tarde, Ali e Kamoun propuseram um novo algoritmo adaptativo no qual a matriz de pesos apenas leva consigo informações de convergência e as informações sobre custo dos enlaces e topologia são informadas pelas entradas de polarização externa dos neurônios [21].

Para fazer uso das Redes Neurais de Hopfield, Ali e Kamoun apresentaram uma forma de modelar o menor caminho a partir do estado final da rede neural. Nesse modelo, o custo do nó x ao nó i é denotado por C_{xi}, o qual assume valores reais positivos e normalizados entre $(0,1]$. Além disso, uma matriz de conexão ρ_{xi} informa a existência dos enlaces, definida por:

$$\rho_{xi} = \begin{cases} 1, & \text{se o arco do nó } x \text{ ao nó } i \text{ existe;} \\ 0, & \text{caso contrário.} \end{cases} \tag{2.7}$$

Cada elemento na matriz C_{xi} é representado por um neurônio. Desta forma, o número de neurônios da rede de Hopfield para este problema é de $n(n-1)$.

No modelo de Ali e Kamoun para roteamento, a equação de energia da rede de Hopfield é definida como:

$$E = E_1 + E_2 + E_3 + E_4 + E_5, \tag{2.8}$$

onde:

$$E_1 = \frac{\mu_1}{2} \sum_{\substack{x=1 \\ (x,i)\neq(d,s)}}^{n} \sum_{\substack{i=1 \\ i\neq x}}^{n} C_{xi} V_{xi},$$

$$E_2 = \frac{\mu_2}{2} \sum_{\substack{x=1 \\ (x,i)\neq(d,s)}}^{n} \sum_{\substack{i=1 \\ i\neq x}}^{n} \rho_{xi} V_{xi},$$

$$E_3 = \frac{\mu_3}{2} \sum_{x=1}^{n} \left\{ \sum_{\substack{i=1 \\ i \neq x}}^{n} V_{xi} - \sum_{\substack{i=1 \\ i \neq x}}^{n} V_{ix} \right\}^2,$$

$$E_4 = \frac{\mu_4}{2} \sum_{x=1}^{n} \sum_{\substack{i=1 \\ i \neq x}}^{n} V_{xi}(1 - V_{xi}),$$

$$E_5 = \frac{\mu_5}{2}(1 - V_{ds}),$$

onde $\mu_1, \mu_2, \mu_3, \mu_4$ e μ_5 são constantes.

A Equação (2.8) tem o objetivo de definir o processo que induz a rede neural ao estado de mais baixa energia. Quando a rede de Hopfield alcança esse estado, o conjunto de saídas representa o menor caminho.

Cada neurônio pode ser externamente excitado por uma polarização (*bias*), a partir de uma entrada externa I_{xi}, como definido pela Equação (2.9):

$$I_{xi} = -\frac{\mu_1}{2} C_{xi}(1 - \delta_{xd}\delta_{is}) - \frac{\mu_2}{2} \rho_{xi}(1 - \delta_{xd}\delta_{is}) - \frac{\mu_4}{2} + \frac{\mu_5}{2} \delta_{xd}\delta_{is}, \tag{2.9}$$

$$\forall (x \neq i), \forall (y \neq i),$$

onde d é o nó destino, s é o nó origem e δ é a função delta de Kronecker que é definida por:

$$\delta_{ab} = \begin{cases} 1, & \text{se } a = b; \\ 0, & \text{caso contrário.} \end{cases} \tag{2.10}$$

Esta polarização ajuda a ajustar o nível de excitação da rede inteira e a repassar os dados sobre a topologia da rede e o par origem-destino.

A matriz de sinapses $T_{xi,yj}$ é definida por:

$$T_{xi,yj} = \mu_4 \delta_{xy}\delta_{ij} - \mu_3 \delta_{xy} - \mu_3 \delta_{ij} + \mu_3 \delta_{jx} + \mu_3 \delta_{iy}. \tag{2.11}$$

As equações (2.3), (2.4) e (2.6) da dinâmica da rede neural de Hopfield levando em consideração a representação do neurônio aplicado ao problema de roteamento podem ser reescritas como:

$$V_{xi} = \frac{1}{1 + e^{-\lambda_{xi} U_{xi}}}, \tag{2.12}$$

$$\forall (x,i) \in \overline{N} \times \overline{N} / x \neq i,$$

$$\frac{dU_{xi}}{dt} = -\frac{U_{xi}}{\tau} + \sum_{y=1}^{n} \sum_{\substack{j=1 \\ j \neq y}}^{n} T_{xi,yj} V_{yj} + I_{xi}, \tag{2.13}$$

$$\frac{dU_{xi}}{dt} = -\frac{U_{xi}}{\tau} - \frac{\partial E}{\partial V_{xi}}. \tag{2.14}$$

Quando a rede atinge o estado de menor energia, uma matriz binária Y é construída a partir de:

$$Y_{xi} = \begin{cases} 1 \text{ se } V_{xi} \geq 0.5; \\ 0 \text{ se } V_{xi} < 0.5. \end{cases} \tag{2.15}$$

Se o valor de Y_{xi} é igual a 1, o enlace entre o nó x e i faz parte do menor caminho. Caso contrário, este enlace não faz parte do menor caminho.

O pseudocódigo do algoritmo de Ali e Kamoun é apresentado no Algoritmo (1):

Algorithm 1: Pseudocódigo do algoritmo de roteamento utilizando Redes Neurais de Hopfield.

```
Recebe parâmetros;
Determina T_{xi,yj};
Recebe C_{xi};
Calcula ρ_{xi};
Recebe fonte e destino;
Calcula I_{xi};
Insere ruído inicial em U_{xi};
Calcula V_{xi} (limiar de U_{xi});
repita
    Repassa histórico de U_{xi} e V_{xi};
    Atualiza os neurônios (U_{xi} e V_{xi});
até ΔV_{xi} < limiar;
Calcula Y_{xi} (binarização de V_{xi});
Obtem caminho a partir de Y_{xi};
```

Vários pesquisadores aperfeiçoaram a abordagem de Ali e Kamoun. Smeda e Hawary [22] simplificaram a equação de energia; Park e Choi [23] mostraram que o modelo de Ali e Kamoun não converge para redes com mais de 20 nós e propuseram uma nova abordagem adequada para redes com muitos nós; Ahn *et al.* [24] apresentaram um algoritmo com convergência mais rápida do que as abordagens anteriores.

Bastos-Filho *et al.* [25] propuseram uma simples equação de diferença finita e discreta em substituição à equação diferencial proposta por Ali e Kamoun:

$$U_{xi}[n+1] = U_{xi}[n] - AU_{xi}[n-1] - BU_{xi}[n-2] + C\left\{\sum_{y=1}^{n}\sum_{\substack{j=1 \\ j\neq y}}^{n} T_{xi,yj}V_{yj}[n] + I_{xi}\right\}, \quad (2.16)$$

onde $U_{xi}[n+1]$ é a próxima entrada do neurônio xi calculado com base nos valores da sua própria entrada em instantes passados $U_{xi}[n]$, $U_{xi}[n-1]$ e $U_{xi}[n-2]$, na saída de todos os neurônios da rede $V_{yj}[n]$, na matriz de pesos sinápticos $T_{xi,yj}$ e na matriz de polarização $I_{xi}[n]$.

Uma versão ainda mais simplificada da Equação (2.16), sem as entradas prévias $U_{xi}[n-2]$, foi proposta por Schuler *et al.* [26]:

$$U_{xi}[n+1] = U_{xi}[n] - AU_{xi}[n-1] + B\sum_{y=1}^{n}\sum_{\substack{j=1 \\ j\neq y}}^{n} T_{xi,yj}V_{yj}[n] + CI_{xi}. \quad (2.17)$$

As propostas de Schuler *et al.* [26] e de Bastos-Filho *et al.* [25] reduzem o custo computacional do algoritmo e permitem implementações em arquiteturas nas quais existem unidades com poucos recursos com alta capacidade de paralelização.

3 Matriz de Blocos Lógicos Programáveis em Campo

"*Engineering: it is like Math, but louder.*"
– David Malki em Wondermark.

A tecnologia VLSI (do inglês, *Very-Large-Scale Integration*) permitiu a criação de circuitos digitais complexos e poderosos graças a capacidade de colocar milhões de transistores em um único *chip* [27]. Contudo, cada *chip* deve ser projetado, fabricado e encapsulado individualmente. Para a produção em larga escala, o custo é razoável. Entretanto, essa abordagem é, em geral, cara e lenta [28].

O tempo consumido para desenvolvê-los e testá-los pode ser crucial em mercados nos quais os produtos devem ser lançados rapidamente. Além disso, esses *chips* são desenvolvidos para uma tarefa específica e, apesar de alcançarem um alto desempenho nessa tarefa, não há como carregar novas funcionalidades neles.

Os dispositivos programáveis permitem a configuração (*i.e.* a programação) de seu comportamento. Desta forma, combinam facilidade de programação em alto nível e rapidez na criação de novos dispositivos com o desempenho de *chips* de propósito único.

Neste capítulo, o conceito de dispositivos programáveis – em particular FPGAs, *Field Programmable Gate Arrays* – é definido. Além disso, uma linguagem de descrição de *hardware*, chamada VHDL, é brevemente apresentada.

3.1 Dispositivos de lógica programável

O primeiro tipo de dispositivo programável que alcançou sucesso foi baseado na tecnlogia PROM [27] (do inglês, *Programmable Read-Only Memory*). Em conjunto

com a EPROM (do inglês, *Erasable Programmable Read-Only Memory*), essas memórias podem ser modeladas para criar circuitos lógicos simples utilizando as linhas de endereço como entrada, de forma que a saída seja definida pelos *bits* armazenados.

Apesar de conseguirem sintetizar circuitos lógicos simples, PROMs são claramente mais adequadas para implementar memórias de computador. PLD (do inglês, *Programmable Logic Device*) é um tipo de dispositivo criado especificamente para implementação de circuitos lógicos [27]. O princípio básico do PLD é um conjunto de portas AND conectadas a um conjunto de portas OR, como ilustrado na Figura 3.1.

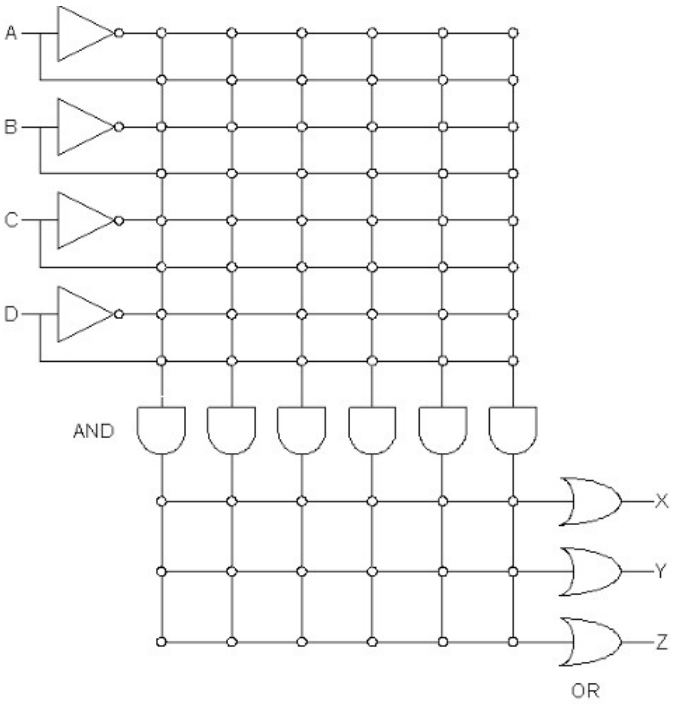

Figura 3.1: Arquitetura básica de um PLD.

Matriz lógica programável (PAL, do inglês *Programmable Array Logic*) é o tipo mais básico de PLD e foi o primeiro dispositivo especificamente a ser programável. Ele consiste em um conjunto de portas lógicas que podem ser conectadas utilizando um outro conjunto de conexões as quais utilizam a tecnologia EPROM. Estes dispositivos podem produzir um pequeno número de flip-flops e são capazes de implementar pequenas máquinas de estado [28].

Dispositivo Lógico Complexo Programável (CPLD, do inglês *Complex Programmable Logic Device)*) foi desenvolvido para superar a pouca capacidade dos PAL's. Esses dispositivos seguem os mesmos princípios deles entretanto eles têm uma série de macroblocos (cada um equivalente a um PAL) conectados utilizando blocos de rotea-

mento.

3.1.1 Matriz de Blocos Lógicos Programáveis em Campo

Matriz de Blocos Lógicos Programáveis em Campo [1] (FPGA, do inglês *Field Programmable Gate Array*) é um dispositivo lógico que contém uma matriz bi-dimensional de células lógicas e conexões programáveis criado em 1985 pela Xilinx Company [27]. A estrutura conceitual de um FPGA é apresentada na Figura 3.2.

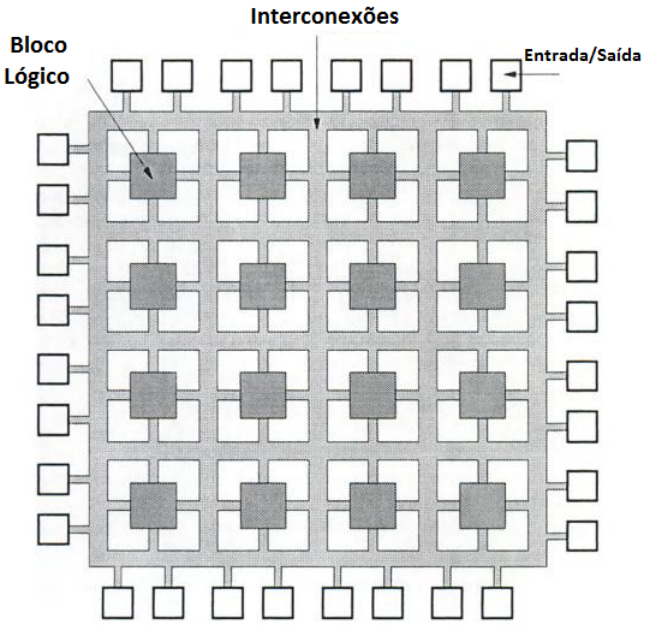

Figura 3.2: Arquitetura básica de um FPGA.

Em vez de de um conjunto de portas como o presente em CPLDs, o FPGA utiliza o conceito de Bloco Lógico Configurável (CLB, do inglês *Configurable Logic Block*). Um CLB típico é ilustrado na Figura 3.3. Cada CLB tem uma *Look-Up Table* (LUT) que pode ser configurada para criar um tipo específico de função lógica. Há ainda um Flip-Flop tipo D que permite que o CLB seja combinatorial ou síncrono.

Cada bloco lógico pode ser configurado (*i.e.* programado) para efetuar uma simples função e as conexões podem ser definidas para criar interconexões entre eles [29]. Desta maneira, circuitos lógicos são implementados em FPGA quebrando a lógica em blocos e então interconectando-os através dessas conexões.

Um FPGA padrão tem centenas de milhares de CLBs de diferentes tipos em um

[1] Uma vez que o processo de programação do dispositivo é feito fora da indústria, isto é, em campo, o dispositivo é conhecido como programável em campo – em inglês, "*field programmable*".

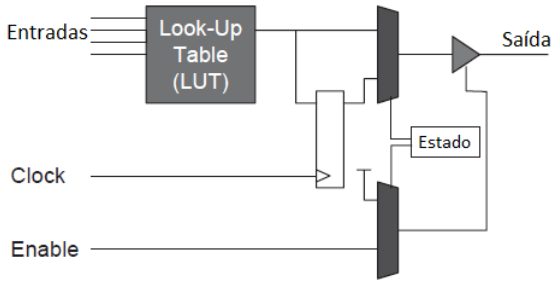

Figura 3.3: Estrutura de um CLB.

único disposito permitindo dispositivos complexos serem implementados em um único chip e configurado facilmente. FPGAs modernos têm capacidade de sintetizar processadores de 32-bits em um único dispositivo [28].

3.2 Linguagem de descrição de *hardware*

HDL (do inglês, *Hardware Description Language*) é um tipo linguagem de programação que facilita a documentação, o projeto e a criação de sistemas digitais [30].

Uma HDL utilizada no projeto de circuitos digitais permite que CPLDs e FPGAs sejam programados rapidamente com milhões de portas lógicas [31]. Ela descreve como o sistema digital é quebrado em blocos e como eles são conectados.

Além disso, com o uso de uma HDL, o projeto de um sistema pode ser simulado antes de ser criado [32]. Dessa forma, os projetistas podem comparar várias alternativas e verificar a sua corretude sem custos nem atrasos de prototipação [30].

VHDL (do inglês, *Very high-speed integrated circuit Hardware Description Language*) e Verilog são HDLs largamente adotadas em projetos. A capacidade das duas são bastante similares, apesar de diferenças sintáticas. A escolha entre elas em um projeto dentro de uma empresa é menos relacionada a uma decisão de projeto e mais a disponibilidade de *software*.

Na realidade, uma briga entre as comunidades de VHDL e Verilog existe há mais de 10 anos para decidir qual seria a melhor linguagem de descrição de *hardware* [28]. Seguindo as diferenças na sintaxe, o maior contraste entre as duas é o contexto na qual estão inseridas. De um lado Verilog segue a abordagem “*bottom-up*" usada na indústria de circuitos integrados, enquanto VHDL segue uma perspectiva mais próxima do conceito “*top-down*".

Neste projeto, a VHDL é a linguagem escolhida em todos os códigos.

3.2.1 VHDL

VHDL foi originalmente patrocinada pelo Departamento de Defesa dos Estados Unidos e posteriormente transferida para o IEEE (*Institute of Electrical and Electronics Engineers*). Ela foi a primeira linguagem padronizada pelo IEEE[2], mais precisamente pelo padrão IEEE 1076 [29]. O padrão foi retificado em 1987 (conhecido como VHDL-87) e foi revisado diversas vezes (VHDL-93, VHDL-2002, etc).

Graças a padronização, uma vez que o código VHDL está escrito, ele pode ser utilizado para implementar um circuito em um dispositivo programável de vários tipos e marcas. Além disso, pode ser enviado a uma fábrica para a criação de um ASIC (do inglês, *Application-Specific Integrated Circuit*) [33].

VHDL tem um comportamento diferente de outras linguagens de programação. Enquanto as declarações nas outras são normalmente sequênciais, em VHDL, por outro lado, elas são paralelas. Esse é um dos motivos que geralmente ela é citada como código; ao invés de programa [33].

Uma grande vantagem do VHDL[3] é a capacidade de utilizar vários níveis de modelos para diferentes arquiteturas [28], como ilustrado na Figura 3.4.

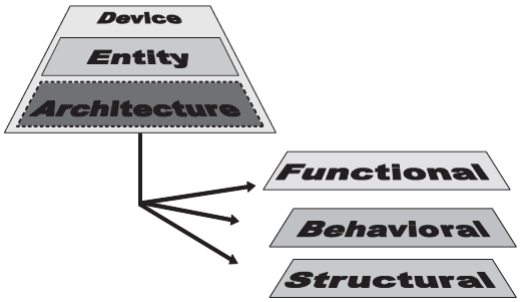

Figura 3.4: Os vários níveis presente em VHDL.

Essa divisão do modelo em sua interface (“entity" em VHDL) e em seu comportamento (“behavior" em VHDL) facilita a modelagem de vários comportamentos para uma única interface. Ademais, facilita alterações posteriores no modelo sem muito esforço.

[2] Verilog também segue padrão IEEE [29].

[3] Verilog também tem o conceito de vários comportamentos para um único módulo (“*module*"), entretanto isso é explícito em VHDL.

3.2.2 Síntese

A síntese de um circuito codificado em VHDL em um dispositivo programável ou em um ASIC segue os passos ilustrados na Figura 3.5.

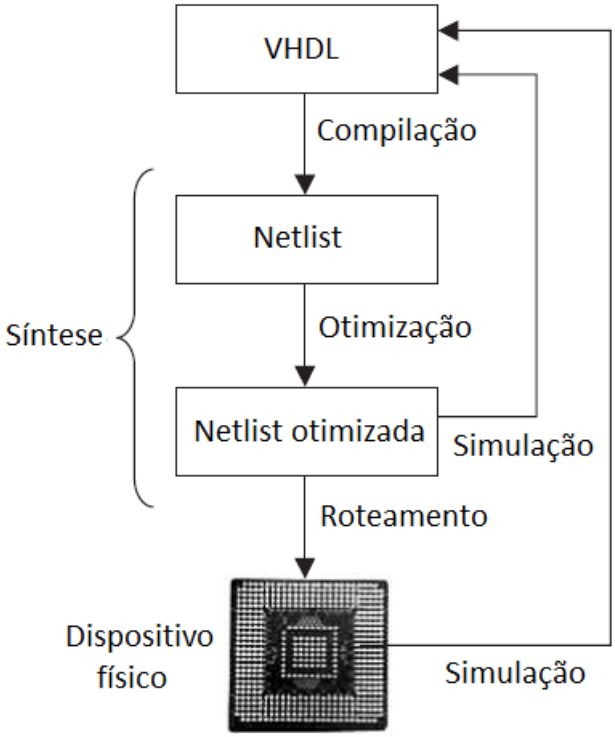

Figura 3.5: Etapas da implementação de um circuito em FPGA.

A primeira etapa no processo de síntese é a compilação. Nessa etapa, o código em VHDL é convertido para uma *netlist* em nível de portas lógicas. O segundo passo é a otimização, nela, a *netlist* criada é otimizada com respeito a área utilizada ou com respeito a velocidade. Por fim, o roteamento gera o *layout* físico para o *chip* PLD/FPGA ou cria as máscaras para o ASIC.

Todo esse processo é claramente impossível de ser feito sem a utilização de *softwares* específicos. Há várias ferramentas EDA (do inglês, *Electronic Design Automation*) para a síntese de circuitos, implementação e simulação utilizando VHDL.

Algumas dessas ferramentas são fornecidas como parte do kit de desenvolvimento de PLDs/FPGAs. Por exemplo, o *software* Altera's Quartus II permite a síntese em *chips* da Altera e o Xilinx's ISE permite em *chips* da Xilinx.

Outras ferramentas também são fornecidas por empresas especializadas em EDA. Por exemplo, o sintetizador Leonardo Spectrum e o Synplify são desenvolvidos respec-

tivamente pelas empresas Mentor Graphics e Synplicity, enquanto o simulador ModelSim é mantido pela companhia Mentor Graphics company.

Apesar dos códigos feitos em VHDL serem completamente simuláveis, nem todos são sintetizáveis. Na realidade, não e difícil criar modelos em VHDL que não são sintetizáveis caso o PLD/FPGA alvo não seja analisado completamente [28]. Esses dispositivos têm um número limitado de blocos lógicos e de recursos de roteamento, qualquer modelagem deve levar em conta essas características.

4 *Proposta e Arranjo Experimental*

"*Wanna go faster?*
Hands up if you want to go faster!"
***Slogan* da Montanha Russa *Alton Towers*.**

Há certas questões na arquitetura do FPGA que devem ser analisadas antes da implementação da rede neural de Hopfield nesta plataforma. Deve-se levar em conta a precisão numérica necessária ao problema, a função de ativação sigmóide e o nível de paralelismo da rede neural.

Nesse capítulo, as principais questões da implementação das Redes Neurais de Hopfield em FPGA são abordadas. Além disso, um modelo da Rede Neural de Hopfield para roteamento de redes de comunicação em VHDL é proposto.

4.1 Paralelismo das Redes Neurais

Redes Neurais apresentam uma variedade de tipos de paralelismo [34]. Antes de criar um modelo para a implementação em um *hardware*, deve-se examinar cuidadosamente tais niveis e analisar se o paralelismo é possível de ser alcançado.

Os tipos de paralelismo que podem estar presente em uma rede neural são [34]:

- **Paralelismo dos Neurônios:** Uma vez que as entradas são inseridas nos neurônios, o processamento de cada um pode ser feito em paralelo.
- **Paralelismo na Saída:** O produto presente no somatório da saída de um neurônio pode ser feito em paralelo e a soma deles pode ser feita com alto paralelismo.
- **Paralelismo de Camada:** Em redes multi-camadas, diferentes camadas podem ser processadas em paralelo.

- **Paralelismo no Treinamento:** Várias sessões de treinamento em uma rede neural podem ser executada paralelamente.

Deve-se notar que pode haver conflitos entre níveis de paralelismo com relação ao *hardware*. Por exemplo, fazer uso do paralelismo dos neurônios em conjunto com o paralelismo nas saídas deles pode levar ao esgotamento dos recursos do dispositivo.

Nas Redes Neurais de Hopfield para roteamento de redes de comunicação, não há preocupação nem com camadas nem com treinamento pois não existe nenhum dos dois conceitos na rede. Ademais, o paralelismo de saída em um FPGA pode levar a mais custo de área, uma vez que mais conexões são necessárias. Portanto, já que cada neurônio processa suas entradas independentemente, o foco do paralelismo neste projeto é o Paralelismo dos Neurônios.

O benefício de aproveitar esse paralelismo nas Redes Neurais de Hopfield para roteamento de redes de comunicação já foi demonstrado na plataforma paralela CUDA [35, 10]. Desta forma, o desempenho do modelo apresentado no projeto pode ser comparado com essa abordagem.

4.1.1 Modelo paralelo de uma Rede Neural de Hopfield para roteamento de uma rede de comunicação

O comportamento de cada neurônio é governado por uma máquina de estados ilustrada na Figura 4.1. Os valores dentro dos retângulos são as representações binárias dos estados utilizadas no modelo VHDL. Além disso, esses estados são compartilhados entre os neurônios, desta forma cada neurônio se mantém informado sobre a situação atual dos outros. Essa informação é utilizada como forma de barreira de sincronia entre os estados.

O estado inicial S_0 do neurônio corresponde ao momento da atualização da entrada do neurônio, vide Equação 2.17 no Capítulo 2. O principal gargalo de todo o algoritmo de roteamento utilizando redes neurais de Hopfield está presente nesse estado, correspondente ao cálculo da Equação (2.17).

Após o neurônio ter a sua entrada atualizada, ele passa para o estado S_1. Quando todos os neurônios estão nesse estado, as saídas de cada um são atualizadas. Deve-se notar que essa barreira de sincronia implícita é necessária, uma vez que o somatório no estado inicial S_0 utiliza as saídas atuais dos neurônios.

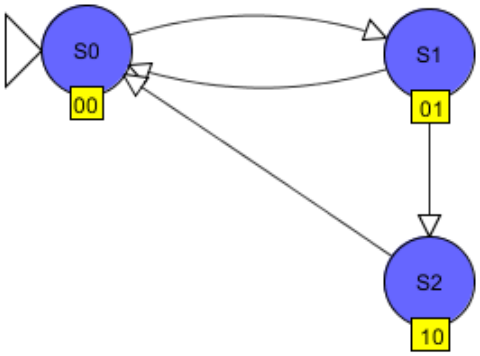

Figura 4.1: Máquina de estados de um neurônio modelado em FPGA.

Ainda no estado S_1, o teste de limiar do neurônio é efetuado, conforme mencinoado no Algoritmo 1. Caso o limiar foi alcançado o neurônio entra no estado S_2, caso não ele vai para o estado S_0. No estado S_2, a saída binarizada é calculada, vide Equação (2.15).

Os estados possíveis dos neurônios são resumidos na Tabela 4.1. Entretanto, deve-se salientar que cada neurônio apenas efetua suas operações em um determinado estado quando há concordância entre todos os outros neurônios. Por exemplo, a saída de um neurônio é apenas atualizada quando todos os neurônios estiverem no mesmo estado de atualização.

Estado	Ação
S_0	Atualização das entradas dos neurônios.
S_1	Atualização das saídas dos neurônios e teste de limiar.
S_2	Cálculo da saída binarizada.

Tabela 4.1: Resumo dos estados de um neurônio no modelo proposto da rede neural de Hopfield.

Na realidade, pode-se dizer que a rede neural tem estados baseado nos estados dos neurônios. O diagrama dessa máquina de estados finita para uma rede de dois neurônios é exemplificado na Figura 4.2. Os valores dentro dos retângulos representam os estados dos neurônios na rede neural.

Devido a necessidade de concordância entre os blocos, pode existir um estado de estagnação (representado na Figura 4.2 como estado S_3) quando algum neurônio alcançar limiar e outro não. Portanto, quando um neurônio alcança o limiar e algum outro não atinge, o primeiro volta para o estado inicial.

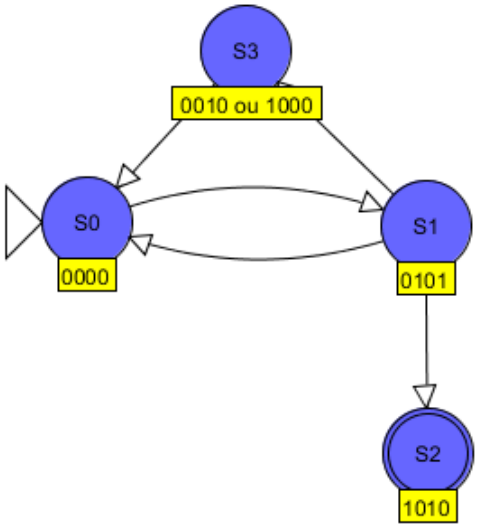

Figura 4.2: Máquina de estados da rede neural modelada em FPGA.

Bloco do Neurônio de uma Rede Neural de Hopfield para Roteamento de Rede de Comunicação

A Figura 4.3 ilustra o bloco proposto para cada neurônio da Rede Neural de Hopfield para roteamento de uma rede de comunicação. Esse neurônio é relativo a uma rede neural com 12 neurônios para o roteamento de uma rede de comunicação de 4 nós.

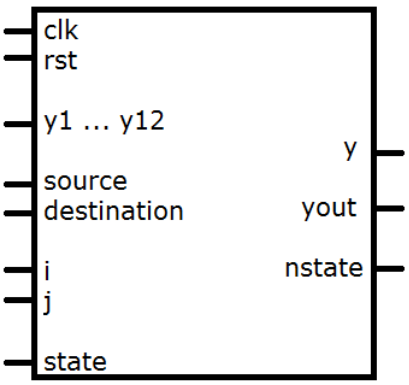

Figura 4.3: Bloco conceitual com as saídas e entradas de um neurônio de uma rede neural de Hopfield para roteamento de uma rede de 4 nós.

O bloco tem como entradas a saída de cada outro neurônio presente na rede neural, o par destino/origem da requisição de roteamento, o par de índices que representa a qual enlace este neurônio corresponde, o estado dos outros neurônios na rede, o sinal de clock e o sinal de reset. Essas saídas e entradas do bloco são definidas no código VHDL presente na Figura 4.4.

```
COMPONENT Neuron IS
    GENERIC (   bits:       INTEGER := 16;
                nodes:      INTEGER := 4;
                neurons:    INTEGER := 4*(4-1)
                );
    PORT (
                state:      INOUT   bit_vector (2*neurons-1 downto 0);
                yDBG:       OUT     sfixed(bits downto -bits);
                yout:       OUT     sfixed(bits downto -bits);
                y1:         IN      sfixed(bits downto -bits);
                y2:         IN      sfixed(bits downto -bits);
                y3:         IN      sfixed(bits downto -bits);
                y4:         IN      sfixed(bits downto -bits);
                y5:         IN      sfixed(bits downto -bits);
                y6:         IN      sfixed(bits downto -bits);
                y7:         IN      sfixed(bits downto -bits);
                y8:         IN      sfixed(bits downto -bits);
                y9:         IN      sfixed(bits downto -bits);
                y10:        IN      sfixed(bits downto -bits);
                y11:        IN      sfixed(bits downto -bits);
                y12:        IN      sfixed(bits downto -bits);
                clk:        IN      STD_LOGIC;
                rst:        IN      STD_LOGIC;
                y:          OUT     bit;
                source:     IN      INTEGER RANGE 0 TO nodes-1;
                destination:IN      INTEGER RANGE 0 TO nodes-1;
                tx:         IN      INTEGER RANGE 0 TO nodes-1;
                ty:         IN      INTEGER RANGE 0 TO nodes-1;
                nstate:     INOUT   bit_vector (1 downto 0)
        );
END COMPONENT;
```

Figura 4.4: Definição das saídas e entradas em VHDL de um neurônio de uma rede neural de Hopfield para roteamento de uma rede de 4 nós.

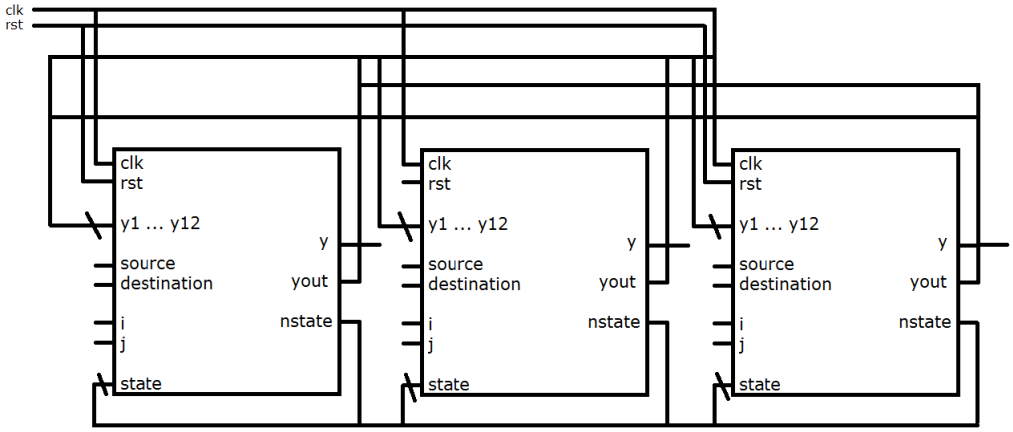

Figura 4.5: Blocos de neurônios interconectados de uma rede neural com 12 nós.

A Figura 4.5 ilustra três neurônios interligados de uma rede neural para o roteamento de uma rede de comunicação de 4 nós. Os outro neurônios (ao total 12 neurônios) não estão na imagem para deixá-la mais visível. A barra nas entrada dos neurônios na figura significa que são várias entradas provenientes de vários neurônios.

4.1.2 Particularidades no Código

Algumas abordagens utilizada na codificação do algoritmo de roteamento utilizando redes neurais de Hopfield devem ser ressaltadas. Nas seções seguintes, os pontos relacionados a criação da Matriz de Sinapses dos Neurônios, ao Cálculo da Polarização dos Neurônios e ao Somatório das Entradas dos Neurônios são discutidos.

Matriz de Sinapses dos Neurônios

A matriz de sinapses $T_{xi,yj}$, definida pela Equação (2.11), tem quatro dimensões, isso torna seu armazenamento no *hardware* impraticável. Para contornar esse problema, os pesos são gerados *on-demand*, isto é, apenas quando o peso da conexão entre neurônios for necessário ele irá ser calculado. Essa abordagem é discutida em [10].

Cálculo da Polarização do Neurônio

A matriz de excitação externa dos neurônios I_{xi}, definida pela Equação (2.9), tem como principal entrada os custos C_{xi} dos enlaces da rede de comunicação. Para deixar o modelo do neurônio dinâmico, esse valor é calculado apenas quando necessário e portanto não é gravado na memória do *hardware*.

Somatório das Entradas dos Neurônios

A atualização das entradas dos neurônios dada pela Equação (2.17) é o principal gargalo na implementação do algoritmo. Devido ao somatório $\sum_{y=1}^{n}\sum_{\substack{j=1\\ j\neq y}}^{n} T_{xi,yj}V_{yj}[n]$ na equação, esse trecho do algorimo pode ter uma ordem de complexidade $O(n^2)$.

Contudo, uma abordagem descrita em [10] deixa o somatório com complexidade $O(n)$. Porém, apesar da diminuição da complexidade, ela traz também um maior número de comparações. Dessa forma, as duas formas são codificadas a fim de analisar se há otimização.

4.2 Precisão Numérica nas Redes Neurais

Uma das questões mais importantes na implementação de uma rede neural em FPGA concerne na relação entre a precisão da representação numérica e a área utilizada no dispositivo [34]. O uso de precisão numérica flutuante dupla ou simples tem o benefício de levar menos erros para rede, entretanto requer mais recursos de *hardware*. Por outro lado, representação de ponto fixo apesar de necessitar de menos recursos, pode introduzir erros significantes a rede neural.

Além da representação numérica, outro ponto importante na implementação está relacionado a função de ativação presente nos neurônios de MCP, vide Equação (2.3). Apesar de várias funçoes lineares poderem ser utilizadas, a função sigmóide é a que se faz mais uso. Entretanto, o custo para implementar esta função é impraticável em *hardware*. Na realidade, várias alternativas podem ser utilizadas levando em consideração a acurácia, a performance e o custo em *hardware*.

Nas subseções 4.2.1 e 4.2.2, a forma na qual cada um dos dois aspectos foram resolvidos no projeto é apresentada.

4.2.1 Representação Numérica

Holt e Banker [36] demonstraram que a representação 16-bit de ponto fixo (ilustrado na Figura 4.6) é o mínimo de precisão necessária para uma rede MLP utilizando o algoritmo *back-propagation*, no qual utiliza uma saída normalizada $[0, 1]$ a partir da função sigmóide. Entretanto, não há nenhuma citação direta ao uso em uma rede neural Hopfield nesse trabalho.

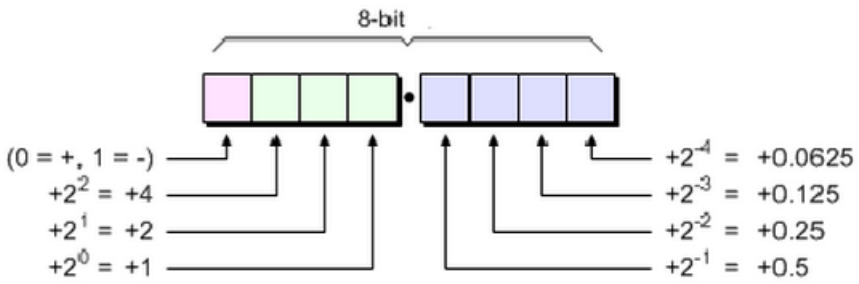

Figura 4.6: Estrtutura da representação numerica utilizando ponto fixo.

Por outro lado, uma vez que as duas redes neurais são baseada no mesmo neurônio MCP, pode-se esperar que elas tenham a mesma característica. Desta forma, a

representação utilizada no projeto foi 16-bit ponto fixo.

Para fazer uso da representação de ponto fixo em VHDL, a biblioteca utilizada foi a *fixed_pkg_c*. Ela é sintetizável em *hardware* e é compatível com VHDL-93 [37].

4.2.2 Função de Ativação

A função sigmóide, ilustrada na Figura 4.7, é utilizada como função de ativação pelas Redes Neurais de Hopfield para Roteamento de Redes de Comunicação e pode ser avaliada de formas mais simples. Contudo, essas simplificaçoes levam, em geral, erros ao cálculo da função. Deve ser ressaltado, ainda, que a função da ativação desempenha um papel importante na ativação dos neurônios e que sua qualidade deve ser garantida a fim de não gerar estados indesejados a rede neural.

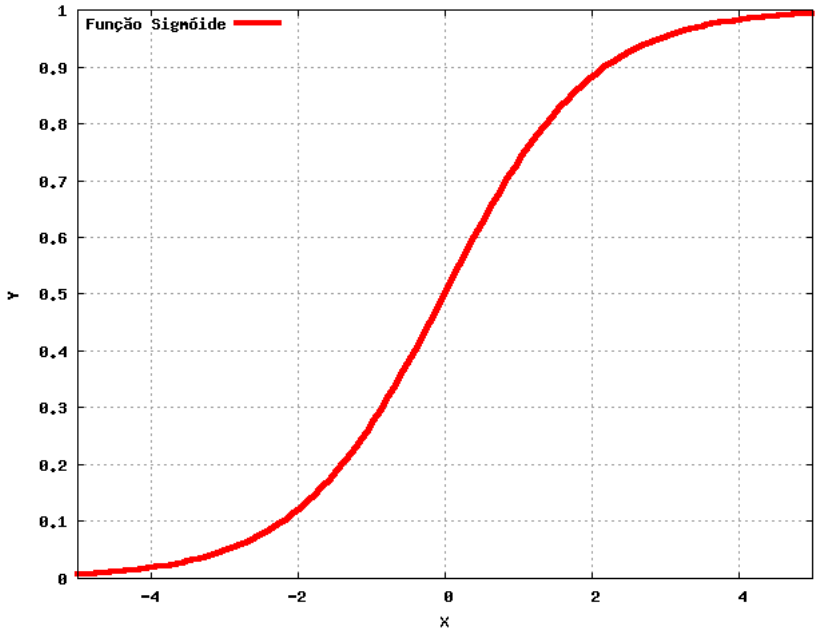

Figura 4.7: A função sigmóide logística com $\lambda = 1$.

As principais alternativas para o cálculo da função sigmóide são Tabela de Consulta e Interpolação.

A Tabela de Consulta consiste em uma tabela gravada na memória do *hardware* na qual cada intervalo no espaço é associado a um valor específico. O erro que essa quantização traz é proporcional ao tamanho de cada intervalo. A Figura 4.8 mostra a função sigmóide e o valor atribuído a cada faixa da Tabela de Consulta.

No projeto, criou-se uma Tabela de Consulta com 17 intervalos. Esses intervalos foram gerados a partir da faixa $[-4,4]$ com um variação $\Delta = 0,5$. Um valor específico

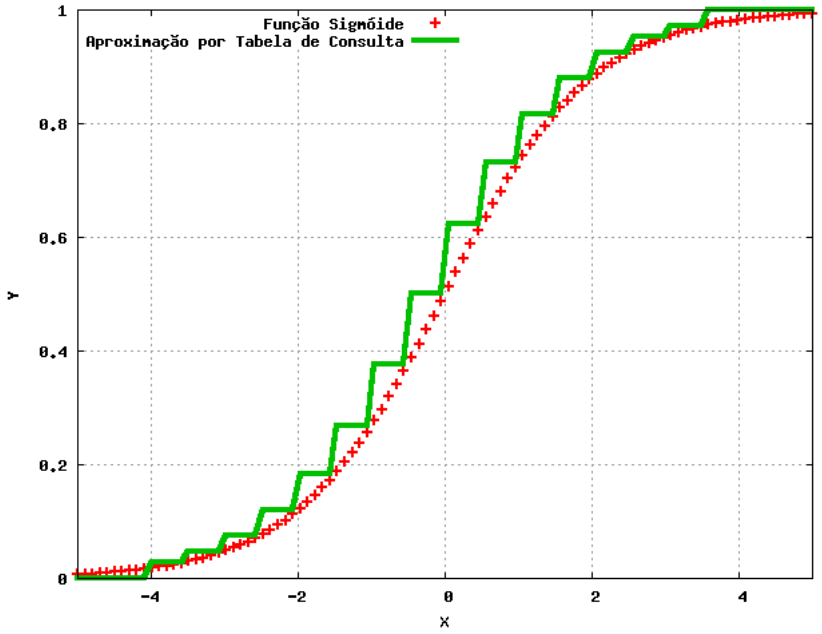

Figura 4.8: Aproximação da função sigmóide a partir de uma tabela de consulta.

da função sigmóide é atribuído a cada intervalo. Esses valores estão descritos na Tabela 4.2.

Por outro lado, a interpolação pode ser feita utilizando a aproximação por partes da função sigmóide. Nessa abordagem, intervalos da função são associados a uma equação com coeficientes específicos guardados na memória do FPGA. A Figura 4.9 ilustra a aproximação com relação a função sigmóide.

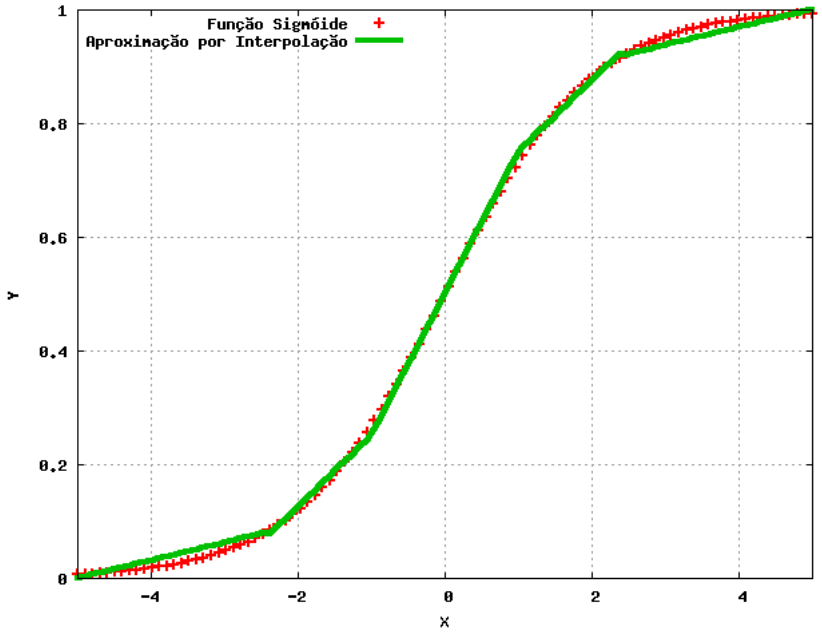

Figura 4.9: Aproximação da função sigmóide a partir de uma interpolação.

Intervalo	Valor
$x < -4,0$	$0,0$
$-4,0 \geq x < -3,5$	$0,029312$
$-3,5 \geq x < -3,0$	$0,047426$
$-3,0 \geq x < -2,5$	$0,075858$
$-2,5 \geq x < -2,0$	$0,119203$
$-2,0 \geq x < -1,5$	$0,182426$
$-1,5 \geq x < -1,0$	$0,268941$
$-1,0 \geq x < -0,5$	$0,377541$
$-0,5 \geq x < 0,0$	$0,5$
$0,0 \geq x < 0,5$	$0,622459$
$0,5 \geq x < 1,0$	$0,731059$
$1,0 \geq x < 1,5$	$0,817575$
$1,5 \geq x < 2,0$	$0,880797$
$2,0 \geq x < 2,5$	$0,924142$
$2,5 \geq x < 3,0$	$0,952574$
$3,0 \geq x < 3,5$	$0,970688$
$x \geq 4,0$	$1,0$

Tabela 4.2: Tabela com os valores atribuídos aos 17 intervalos para a aproximação da função sigmóide logística.

A interpolação utilizada no projeto utiliza os coeficientes descritos na Tabela 4.3. Esses valores são baseados no trabalho em [38].

Operação	Condição
$Y = 1$	$\|X\| \geq 5$
$Y = 0,03125 \cdot \|X\| + 0,84375$	$2,375 \leq \|X\| < 5$
$Y = 0,125 \cdot \|X\| + 0,625$	$1 \leq \|X\| < 2,375$
$Y = 0,25 \cdot \|X\| + 0,5$	$0 \leq \|X\| < 1$

Tabela 4.3: Tabela com os valores atribuídos aos 17 intervalos para a aproximação da função sigmóide logística.

Um maior número de intervalos nas duas abordagens significa uma melhor aproximação da sigmóide. Entretanto, maior quantidade de memória será usada no FPGA. Em particular, a interpolação necessita de cálculo de produtos na equação enquanto o uso de tabelas de consultas necessita de condicionais adicionais. Desta forma, as duas abordagens foram utilizadas no projeto a fim de ilustrar o custo de área necessária e a eficácia de cada uma.

4.3 Arranjo Experimental

O algoritmo de roteamento baseado em redes neurais de Hopfield modelado em VHDL foi simulado utilizando a ferramenta Quartus II 9.1 Web Edition. O dispositivo configurado na aplicação como alvo foi o modelo EP3SE50F780I4L da família Stratix III. Esse dispositivo dispõe de 38.000 *Combinational ALUT's*, 19.000 *Memory ALUT's*, 38.000 *Dedicated Logic Registers*, 488 pinos de entrada/saída e 48 blocos *DSP*.

Os parâmetros utilizados na simulação da rede neural de Hopfield foram baseados nos valores contidos em [15]. Esse valores estão resumidos na Tabela 4.4.

Parâmetro	Valor
A	$0,001$
B	$0,001$
C	$0,001$
μ_1	950
μ_2	2500
μ_3	1500
μ_4	475
μ_5	2500

Tabela 4.4: Valores dos parâmetros da rede neural de Hopfield para roteamento de redes de comunicação.

O processo de roteamento foi efetuado em três diferentes redes de comunicação: rede com 4 nós, rede com 5 nós e rede com 6 nós, respectivamente ilustradas nas figuras: Figura 4.10, Figura 4.11 e Figura 4.12. O objetivo é analisar a escalabilidade do modelo relacionando a área utilizada pelo modelo no dispositivo com o número de nós e também relacionando o tempo para convergência da rede com a quantidade de nós.

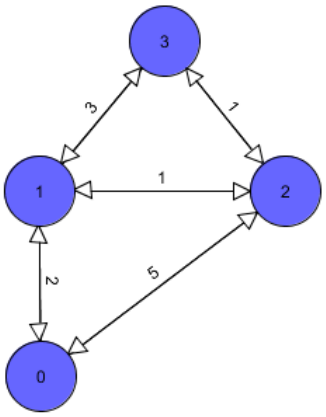

Figura 4.10: Rede de comunicação com quatro nós utilizada na modelagem da arquitetura de rede neural de Hopfield para roteamento.

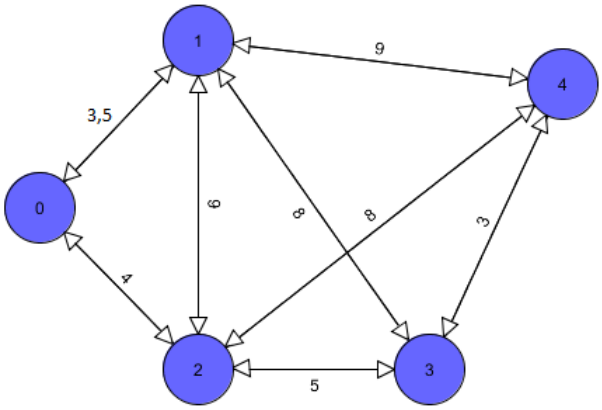

Figura 4.11: Rede de comunicação com cinco nós utilizada na modelagem da arquitetura de rede neural de Hopfield para roteamento.

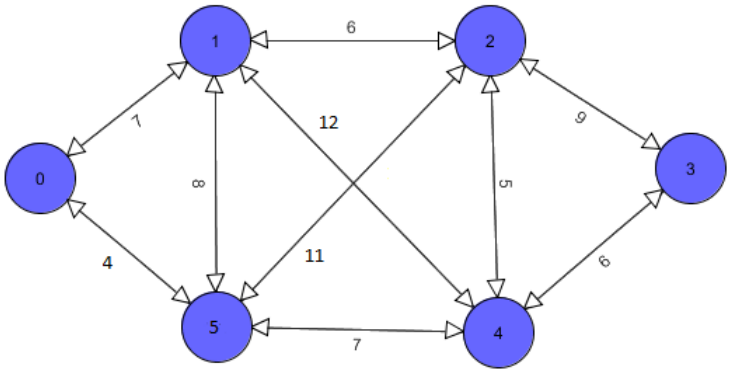

Figura 4.12: Rede de comunicação com seis nós utilizada na modelagem da arquitetura de rede neural de Hopfield para roteamento.

5 Resultados e Discussão

"We shall go on to the end. We shall fight with growing confidence and growing strength in the air; We shall defend our island, whatever the cost may be; We shall never surrender."

– Sir. Winston Churchill.

Após as simulações concluídas com o modelo codificado em VHDL da rede neural de Hopfield para roteamento de redes de comunicação, algumas análises foram realizadas. Essas análises estão relacionadas com a validação do modelo, com o custo de área utilizada pelo modelo e com o tempo para convergência da rede neural.

Primeiramente, o resultado ao final das simulações deve ser analisado a fim de checar se os neurônios ativos estão corretos e coesos. Desta forma, checa-se se o modelo foi codificado sem erros e se está respondendo corretamente a uma requisição de roteamento.

Além disso, deve ser analisada a abordagem escolhida no modelo para atualização dos neurônios, discutida na Seção 4.1.2, com respeito a otimização do somatório das entradas do neurônio. Assim, pode-se garantir se, no FPGA, essa otimização é escalável com a quantidade de nós na rede, levando em consideração o custo de área utilizada por cada abordagem.

A aproximação da função de ativação é outra decisão de projeto que deve ser julgada. Cada simplificação da função sigmóide, discutida na Seção 4.2.2, precisa ser relacionada com o custo de área necessário no FPGA, a fim de encontrar a melhor alternativa.

Por fim, o tempo para a rede neural convergir e retornar um resultado válido a uma requisição é um atributo que deve ser comparado com outras abordagens de implementação da rede neural presentes na literatura.

Nesta seção, a validação do modelo, as relações de custo de área com as abor-

dagens escolhidas e a comparação de tempos são apresentadas.

5.1 Validação

A saída final de um neurônio indica se o enlace no qual o neurônio representa está presente no menor caminho encontrado. Por exemplo, se o neurônio (i, j) está ativado quando a rede alcançar a convergência, o trajeto partindo do Nó i para o Nó j faz parte do menor caminho.

O resultado da simulação de uma requisição de roteamento da rede de quatro nós utilizando as redes neurais de Hopfield é ilustrado na Figura 5.1. A requisição feita foi para o menor caminho partindo do Nó 0 ao Nó 3. Os neurônios 1, 5, 9 e 10 estão ativos quando a convergência é alcançada. Esses neurônios representam, respectivamente, os enlaces $(0,1)$, $(1,2)$, $(2,3)$ e $(3,0)$. A Figura 5.2 ilustra o menor caminho.

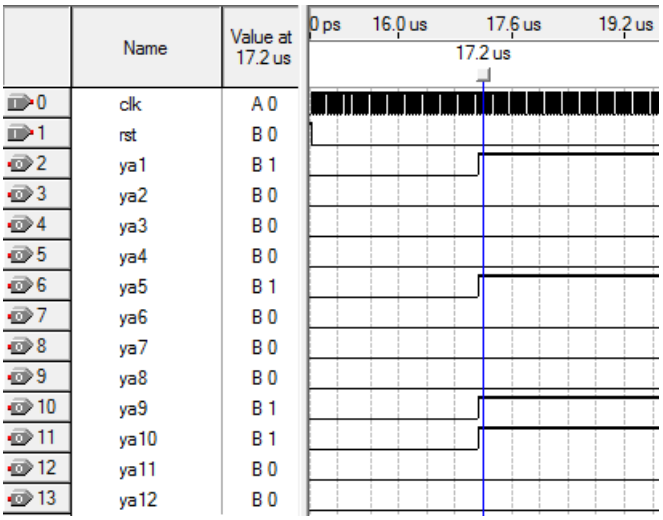

Figura 5.1: Resultado da simulação da rede neural de Hopfield para roteamento de uma rede de comunicação de quatro nós no simulador do Quartus II.

Na Figura 5.3, o resultado final da simulação de uma requisição de roteamento da rede de cinco nós utilizando as redes neurais de Hopfield é ilustrado. A requisição tem como origem o Nó 0 e tem como destino o Nó 4. Pelo diagrama, os neurônios 2, 11, 16 e 17 estão ativos quando a convergência é alcançada. Esses neurônios representam, respectivamente, os enlaces $(0,2)$, $(2,3)$, $(3,4)$ e $(4,0)$. O caminho encontrado pela rede neural é ilustrada na Figura 5.4.

Por fim, o estado final dos neurônios da rede neural para o roteamento da rede de seis nos é apresentado na Figura 5.5. A requisição tem como origem o Nó 0 e

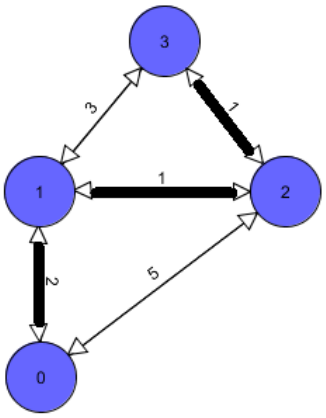

Figura 5.2: Caminho encontrado pela rede neural de Hopfield para a requisiçao Nó origem 0 e Nó destino 3 em uma rede de quatro nós.

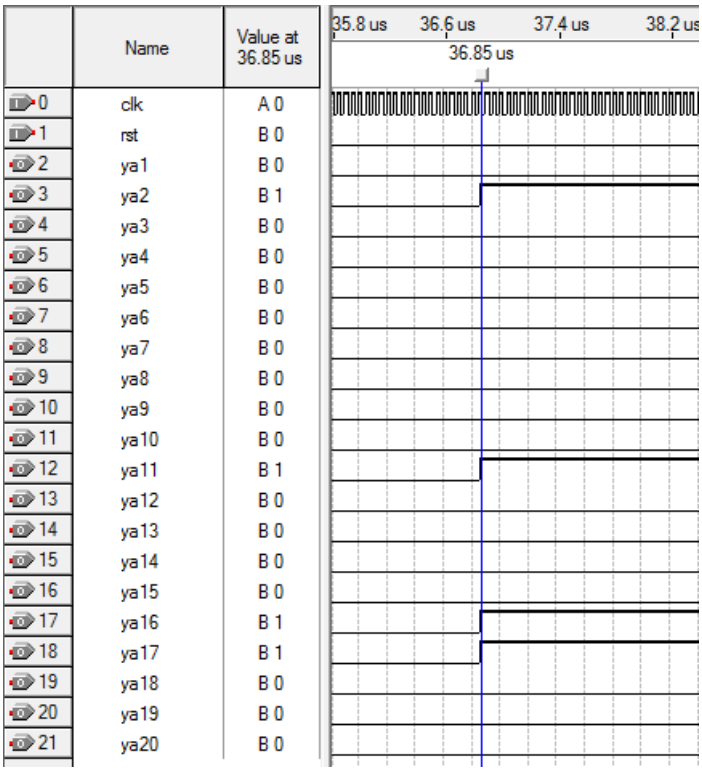

Figura 5.3: Resultado da simulação da rede neural de Hopfield para roteamento de uma rede de comunicação de 5 nós no simulador do Quartus II.

destino o Nó 3. Na convergência, os neurônios 5, 16, 24 e 30 estão ativos, os quais representam, respectivamente, os enlaces $(0,5)$, $(3,0)$, $(4,3)$ e $(5,4)$. A Figura 5.6 ilustra o menor caminho encontrado.

Nas três redes simuladas, os caminhos encontrados são coesos e são os menores para a requisição. Portanto, a codificação em VHDL da rede neural de Hopfield para roteamento de redes de comunicação pode ser considerada válida.

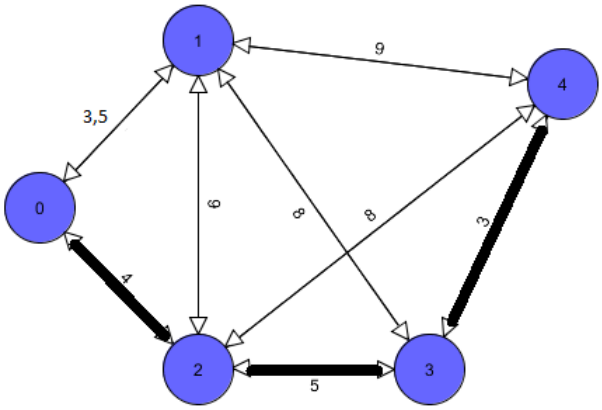

Figura 5.4: Caminho encontrado pela rede neural de Hopfield para a requisiçao Nó origem 0 e Nó destino 4 em uma rede de cinco nós.

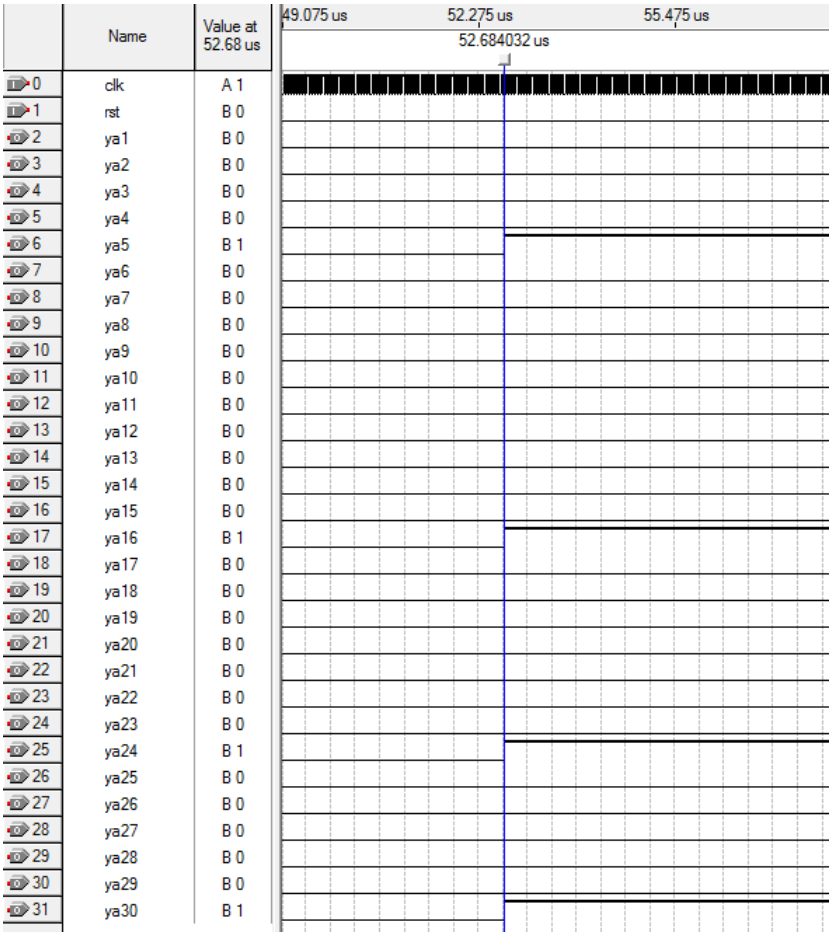

Figura 5.5: Resultado da simulação da rede neural de Hopfield para roteamento de uma rede de comunicação de seis nós no simulador do Quartus II.

5.2 Custo da Redução da Complexidade da Função de Atualização dos Neurônios

A utilização dos recursos do FPGA para a rede neural de Hopfield para roteamento das três redes é descrita na Tabela 5.1. As duas abordagens para atualização dos

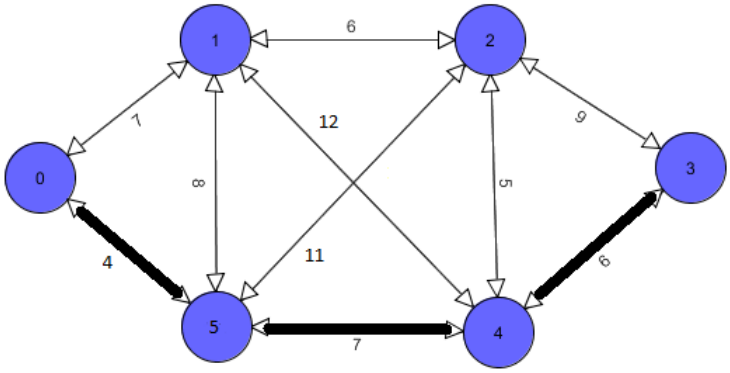

Figura 5.6: Caminho encontrado pela rede neural de Hopfield para a requisiçao Nó origem 0 e Nó destino 3 em uma rede de seis nós.

neurônios mencionadas na Seção 4.1.2 foram simuladas, a versão otimizada de ordem $O(n)$ e a versão não otimizada de ordem $O(n^2)$.

Rede	Recurso do FPGA	$O(n)$	$O(n^2)$
Quatro Nós	ALUTs	5,529 (15%)	6,365 (17%)
	Registros Dedicados	2,221 (6%)	2,569 (7%)
	Lógica Total	18%	21%
Cinco Nós	ALUTs	11,591 (31%)	11,564 (30%)
	Registros Dedicados	3,701 (10%)	4,281 (11%)
	Lógica Total	41%	40%
Seis Nós	ALUTs	20,463 (54%)	18,895 (50%)
	Registros Dedicados	5,551 (15%)	6,421 (17%)
	Lógica Total	78%	70%

Tabela 5.1: Recursos utilizados no FPGA por cada abordagem de atualização de neurônios.

Para a rede com quatro nós, a redução da complexidade impacta razoavelmente no uso dos recursos do FPGA. Por outro lado, a medida que o tamanho da rede aumenta, essa otimizaçao já não é mais visível e chegando, por fim, aumentar a utilização do dispositivo.

5.3 Custo das Aproximações da Função de Ativação

A utilização dos recursos do FPGA para as duas aproximações da função sigmóide nas redes neurais para as três redes de comunicação é descrita na Tabela 5.2. Em todos os casos apresentados na tabela, fez-se uso da a otimização da atualização dos neurônios.

Rede	Recurso do FPGA	Tabela de Consulta	Interpolação
Quatro Nós	ALUTs	5,529 (15%)	7,350 (19%)
	Registros Dedicados	2,221 (6%)	2,425 (6%)
	Lógica Total	18%	25%
Cinco Nós	ALUTs	11,591 (31%)	15,419 (41%)
	Registros Dedicados	3,701 (10%)	4,621 (12%)
	Lógica Total	41%	55%
Seis Nós	ALUTs	20,463 (54%)	26,168 (69%)
	Registros Dedicados	5,551 (15%)	6,693 (18%)
	Lógica Total	78%	96%

Tabela 5.2: Recursos utilizados no FPGA por cada abordagem de aproximação da função sigmóide.

As duas abordagens trouxeram a mesma saída final dos neurônios. Desta forma, os erros introduzidos pelas aproximações não afetaram no algoritmo de roteamento. Por outro lado, a Interpolação faz uso de muito mais recursos do que a Tabela de Consulta. Ela utiliza, por exemplo, quase 20% mais área do FPGA.

5.4 Tempo para Convergência

O tempo médio de uma requisição à rede neural de Hopfield para roteamento da rede de quatro nós convergir é descrito na Tabela 5.3. Nessa tabela, o tempo é comparado com os valores presentes em [10]. A coluna CPU refere-se ao tempo necessário em uma implementação sequêncial em um computador normal, a coluna GPU contém o de uma implementaçao paralela na plataforma CUDA [10].

Rede	CPU	GPU	FPGA
Seis Nós	2598 μs	480 μs	33.30 μs

Tabela 5.3: Tempo médio necessário para a rede neural de Hopfield para roteamento de redes de comunicaçao convergir.

A implementação em FPGA gera uma aceleração no processo de roteamento de 78 vezes em relação a versão sequêncial. Por outro lado, o *speed-up* alcançado com relação a versão GPU é de aproximadamente 15.

6 Conclusões e Trabalhos Futuros

"Run, rabbit, run.
Dig that hole, forget the sun,
And when at last the work is done
Don't sit down it's time to dig another one. "
Pink Floyd – Breathe

Neste trabalho, foi desenvolvido um modelo em VHDL das redes neurais de Hopfield para roteamento de redes de comunicação. Esse modelo foi criado com foco na sua sintetizabilidade em dispositivos FPGA e na utilização dos recursos desses dispositivos. Além disso, ele foi simulado em uma ferramenta popular de simulação a fim de validar seu código e avaliar o seu desempenho em redes de comunicação.

O modelo proposto para a rede neural de Hopfield pode ser considerado válido, uma vez que as requisições feitas à rede nas simulações foram respondidas corretamente. Desta forma, as redes neurais de Hopfield são possíveis em dispositivos FPGA.

Particularmente ao modelo proposto, a simplificação da função de ativação utilizada no modelo não traz erros significativos ao algoritmo de roteamento. Ademais, a abordagem de Tabela de Consulta, apesar de ter uma baixa aproximação, obteve resultados iguais à Interpolação, entretanto, ela utiliza menos recursos do FPGA. Portanto, ela deve ser a escolhida em implementações futuras.

Além disso, a otimização utilizada no somatório das atualizações dos neurônios não é escalável. Desta forma, implementações com maiores quantidade de nós na rede deve utilizar a abordagem não otimizada do algoritmo.

Por fim, o modelo proposto simulado é 78 vezes mais rápido do que a versão sequêncial da rede neural de Hopfield em um computador comum. Além disso, o *speed-up* alcançado com relação a uma versão paralela em GPU é de aproximadamente 15. Portanto, além de ser possível a implementação das redes neurais em

FPGA, elas têm um ótimo desempenho na plataforma.

Como trabalhos futuros, deve-se focar na escalabilidade do modelo da rede neural em FPGA. Uma forma de melhorar esse ponto é diminuir ainda mais a utilização de área do dispositivo. Para isto, pode-se utilizar outras abrdagens para representaçao numérica de ponto fixo e outras aproximações ainda menos custosas da função sigmóide. Além disso, menos área utilizada significa maior velocidade no modelo implementado.

Outro ponto que deve ser trabalhado no futuro é a adaptação de outras técnicas em dispositivos FPGA. Como por exemplo, técnicas que trabalhem em conjunto com outras, como o MOPSO, que é uma técnica de Inteligência Computacional baseada em Inteligência de Enxames na qual encontra os parâmetros para as redes neurais de Hopfield.

Referências Bibliográficas

[1] ENGELBRECHT, A. P. *Computational Intelligence: An Introduction*. [S.l.]: Wiley Publishing, 2007. ISBN 0470035617.

[2] HOPFIELD, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, v. 79, n. 8, p. 2554–2558, Abril 1982. ISSN 0027-8424.

[3] RAUCH, H. E.; WINARSKE, T. Neural networks for routing communication traffic. *Control Systems Magazine, IEEE*, v. 8, n. 2, p. 26–31, Abril 1988. ISSN 0272-1708.

[4] KOJIC, N. S.; RELJIN, I. S.; RELJIN, B. D. Routing in optical networks by using neural network. In: *Neural Network Applications in Electrical Engineering, 2006. NEUREL 2006. 8th Seminar on*. [S.l.: s.n.], 2006. p. 65–68.

[5] KOJIC, N. S.; RELJIN, I. S.; RELJIN, B. D. Different wavelength assignment techniques in all-optical networks controlled by neural network. In: *Telecommunications in Modern Satellite, Cable and Broadcasting Services, 2007. TELSIKS 2007. 8th International Conference on*. [S.l.: s.n.], 2007. p. 401–404.

[6] KOJIC, N. S.; RELJIN, I. S.; RELJIN, B. D. All-optical network with simultaneous in-node routing and wavelength assignment. In: *Telfor Journal*. [S.l.: s.n.], 2009. Vol. 1.

[7] KOJIC, N.; RELJIN, I.; RELJIN, B. Neural network for finding optimal path in packet-switched network. In: *Neural Network Applications in Electrical Engineering, 2004. NEUREL 2004. 2004 7th Seminar on*. [S.l.: s.n.], 2004. p. 91–96.

[8] BASTOS-FILHO, C. J. A. et al. Hopfield neural networks for routing in optical networks. *12th International Conference on Transparent Optical Networks, 2010, Munique*, v. 1, p. 1 – 6, 2010. ISSN 1062-922X.

[9] DIJKSTRA, E. A note on two problems in connection with graphs. *Numerische Mathematik*, v. 1, p. 269–271, 1959.

[10] BASTOS-FILHO, C. J. A. et al. Optimizing a routing algorithm based on hopfield neural networks for graphic processing units. In: *IEEE Symposium on Foundations of Computational Intelligence (IEEE FOCI'11)*. [S.l.: s.n.], 2011.

[11] HAYKIN, S. *Neural Networks: A Comprehensive Foundation*. New York: Macmillan, 1994.

[12] RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach (2nd Edition)*. 2. ed. [S.l.]: Prentice Hall, 2002. Hardcover. (Prentice Hall series in artificial intelligence). ISBN 0137903952.

[13] NOBACK, C. R. et al. *The Human Nervous System: Structure and Function*. [S.l.]: Humana Press, 2005. Pdf. ISBN 1588290395.

[14] MCCULLOCH, W.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, v. 5, n. 4, p. 115–133, Dec 1943.

[15] SANTANA, R. A. *Roteamento em Redes Ópticas Transparentes Utilizando Redes Neurais de Hopfield*. [S.l.]: Dissertação de Mestrado, UPE. 2010, 2010.

[16] ROSENBLATT, F. Book. *Principles of neurodynamics; perceptrons and the theory of brain mechanisms*. [S.l.]: Spartan Books Washington„ 1962.

[17] MINSKY, M.; PAPERT, S. Book. *Perceptrons : an introduction to computational geometry*. Expanded ed. [S.l.]: MIT Press, Cambridge, Mass. :, 1988. xv, 292 p. : p. ISBN 0262631113.

[18] HOPFIELD, J. J. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences of the United States of America*, v. 81, n. 10, p. 3088–3092, May 1984. ISSN 0027-8424.

[19] COHEN, M. A.; GROSSBERG, S. Absolute stability of global pattern formation and parallel memory storage by competitive neural networks. IEEE Computer Society Press, Los Alamitos, CA, USA, p. 70–81, 1988.

[20] THOMOPOULOS, S. C. A.; ZHANG, L.; WANN, C. D. Neural network implementation of the shortest path algorithm for traffic routing in communication networks. In: *Neural Networks, 1991. 1991 IEEE International Joint Conference on*. [S.l.: s.n.], 1991. p. 2693–2702 vol.3.

[21] ALI, M. K. M.; KAMOUN, F. Neural networks for shortest path computation and routing in computer networks. *Neural Networks, IEEE Transactions on*, v. 4, n. 6, p. 941–954, Nov 1993. ISSN 1045-9227.

[22] SMEDA, A. A.; EL-HAWARY, M. E. Application of hopfield neural network in routing for computer networks. In: *Electrical and Computer Engineering, 1999 IEEE Canadian Conference on*. [S.l.: s.n.], 1999. v. 1, p. 145–149 vol.1.

[23] PARK, D. C.; CHOI, S. E. A neural network based multi-destination routing algorithm for communication network. In: *Neural Networks Proceedings, 1998. IEEE World Congress on Computational Intelligence. The 1998 IEEE International Joint Conference on*. [S.l.: s.n.], 1998. v. 2, p. 1673–1678 vol.2.

[24] AHN, C. W. et al. Shortest path routing algorithm using hopfield neural network. *Electronics Letters*, v. 37, n. 19, p. 1176–1178, Setembro 2001. ISSN 0013-5194.

[25] BASTOS-FILHO, C. J. A.; SANTANA, R. A.; OLIVEIRA, A. L. I. A novel approach for a routing algorithm based on a discrete time hopfield neural network. In: *Foundations of Computational Intelligence, 2007. FOCI 2007. IEEE Symposium on*. [S.l.: s.n.], 2007. p. 363–369.

[26] SCHULER, W. H.; BASTOS-FILHO, C. J. A.; OLIVEIRA, A. L. I. A novel hybrid training method for hopfield neural networks applied to routing in communications networks. *Int. J. Hybrid Intell. Syst.*, IOS Press, v. 6, n. 1, p. 27–39, 2009. ISSN 1448-5869.

[27] BROWN, S. et al. *Field-Programmable Gate Arrays*. 4. ed. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1992. (Series: The Springer International Series in Engineering and Computer Science, Vol. 180). ISBN 9780792392484.

[28] WILSON, P. *Design Recipes for FPGAs: Using Verilog and VHDL*. 1. ed. [S.l.]: Elsevier, 2007. ISBN 0750668458, 9780750668453.

[29] CHU, P. P. *FPGA Prototyping by VHDL Examples: Xilinx Spartan-3 Version*. [S.l.]: Elsevier, 2008. ISBN 0470185317, 9780470185315.

[30] XIU, L. *VLSI Circuit Design Methodology Demystified: A Conceptual Taxonomy*. [S.l.]: Wiley-IEEE Press, 2007. ISBN 0470127422, 9780470127421.

[31] RAFIQUZZAMAN, M. *Fundamentals of Digital Logic and Microcomputer Design*. 3rd. ed. [S.l.]: Rafi Systems, Incorporated, 2000. ISBN 0966498038.

[32] DUBEY, R. *Introduction to Embedded System Design Using Field Programmable Gate Arrays*. 1st. ed. [S.l.]: Springer Publishing Company, Incorporated, 2008. ISBN 1848820151.

[33] PEDRONI, V. A. *Circuit Design with VHDL*. Cambridge, MA, USA: MIT Press, 2004. ISBN 0262162245.

[34] OMONDI, A. R.; RAJAPAKSE, J. C. *FPGA Implementations of Neural Networks*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006. ISBN 0387284850.

[35] NVIDIA. *NVIDIA CUDA Programming Guide 2.3*. [S.l.: s.n.], 2009.

[36] HOLT, J.; BAKER, T. Back propagation simulations using limited precision calculations. In: *Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on*. [S.l.: s.n.], 1991. ii, p. 121 –126 vol.2.

[37] BISHOP, D. Fixed point package user's guide. p. 49, 2009.

[38] PYETRO, A. F. *Implementação de uma arquitetura de Redes Neurais MLP utilizando FPGA*. [S.l.]: Trabalho de Graduaçao, CIn, UFPE., 2008.

ANEXO A – Código VHDL do Modelo da Rede Neural de Hopfield