X Congresso Brasileiro de Inteligência Computacional (CBIC'2011), 8 a 11 de Novembro de 2011, Fortaleza, Ceará

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

M. C. Oliveira Júnior, C. J. A. Bastos-Filho
Escola Politécnica de Pernambuco, Universidade de Pernambuco
marcos@ecomp.poli.br, carmelofilho@ieee.org

Resumo – As Redes Neurais de Hopfield (HNN) são sistemas recorrentes que podem ser utilizados para resolver o problema de roteamento em redes de comunicação. Apesar de sua eficácia e capacidade de adaptação, as HNNs têm tempo de resposta mais alto quando comparadas aos algoritmos tradicionais, considerando que ambos são executados em plataformas sequênciais. Por outro lado, as HNN são apropriadas para implementações em plataformas paralelas, como os FPGA. Neste trabalho, é apresentada uma implementação das HNN em FPGA para o problema de roteamento. O modelo proposto é 78 vezes mais rápido do que a versão sequêncial da HNN executada em um computador QuadCore com 8GB de memória RAM e 15 vezes mais rápido do que a versão paralela em GPU para o caso estudado.

Palavras-chave – Algoritmos de Roteamento, Redes Neurais de Hopfield, FPGA.

Abstract – Hopfield Neural Networks (HNN) are recurrent systems which can be applied for routing in communications networks. Despite their efficiency and adaptation ability, HNNs running in sequential platforms present a slower response when compared to classical routing algorithms. On the other hand, HNNs are suitable to be implemented in parallel platforms, such as FPGA. In this paper, we depict a HNN implementation for FPGA. We show a speedup of 78 and 15, when compared to the sequential version of the HNN and the parallel version of the HNN running on GPUs, respectively.

Keywords – Routing algorithms, Hopfield Neural Networks, FPGA.

1. INTRODUÇÃO

Os algoritmos de roteamento utilizados nas redes de comunicação têm um grande impacto no desempenho destas redes. Desta forma, torna-se necessário o desenvolvimento de algoritmos de roteamento que otimizem o uso dos recursos da rede.

Redes Neurais de Hopfield (HNN, *Hopfield Neural Networks*) são sistemas recorrentes que podem ser utilizados para resolver o problema de roteamento [1, 2]. O seu uso vai além do roteamento em redes de computadores, sendo possível também sua aplicação em redes de alta capacidade [3]. As HNNs podem ser consideradas como uma alternativa interessante devido à sua característica de adaptação. Este atributo é interessante para algoritmos de roteamento, uma vez que as redes de comunicação estão sujeitas a constantes mudanças. Contudo, apesar de sua eficácia, as HNN implementadas em plataformas sequênciais apresentam tempo de resposta alto quando comparadas como outros algoritmos clássicos de roteamento, como por exemplo o algoritmo de Dijkstra.

Por outro lado, por causa do comportamento paralelo intrínseco, as HNNs são apropriadas para implementações em plataformas paralelas. Esta otimização no processo de roteamento com HNNs já foi demonstrada em alguns dispositivos massivamente paralelos, como em Unidades de Processamento Gráfico (GPUs, *Graphic Processing Units*) [4].

Matrizes de Blocos Lógicos Programáveis em Campo (FPGA, *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 são plataformas inerentemente paralelas e possibilitam a implementação de algoritmos paralelos.

Dados o atributo de paralelismo das HNNs e dos FPGAs, a adaptação do processo de roteamento nestas plataformas usando este paradigma pode levar a desempenhos comparáveis aos algoritmos tradicionais. Além disso, existe a característica de adaptação das HNNs, podendo ser esta uma vantagem em ambientes reais.

Este trabalho propõe um modelo paralelo para implementação de HNNs para roteamento em redes de comunicação em FPGAs. As dificuldades encontradas estão relacionadas a aspectos de implementação, como a precisão numérica e a codificação da função de ativação.

Este artigo está organizado da seguinte forma: Na Seções 2 e 3 são apresentadas visões gerais sobre as HNN para Roteamento em Redes de Comunicação e sobre os FPGAs, respectivamente. A motivação e a proposta do modelo paralelo de HNN para FPGA são descritas na seção 4. Os resultados das simulações são apresentados no Capítulo 5. E, por fim, no Capítulo 6, são apresentadas as conclusões e considerações finais.

2. REDES NEURAIS DE HOPFIELD

As Redes Neurais de Hopfield (HNN, *Hopfield Neural Networks*) são redes com realimentação apresentadas por John Hopfield em 1982 [1]. As HNNs podem ser utilizadas como memórias associativas, na qual a HNN é capaz de armazenar informações

baseadas em alguns de seus estados [5]. Além disso, as HNNs podem resolver problemas de otimização combinatória. Neste caso, a HNN possui uma função de energia que provê uma medida de desempenho para o problema de otimização, que por sua vez possui um conjunto grande, mas finito, de possíveis soluções [6].

O modelo de Hopfield consiste em um conjunto de neurônios e um conjunto correspondente de atrasos unitários, formando um sistema realimentado de múltiplos laços paralelos. 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) [5].

As HNN são construídas com neurônios de McCulloch-Pitts [7]. Desta forma, cada neurônio possui um somatório do conjunto de entradas ponderadas (U_i) e uma saída (V_i), que é o resultado da aplicação de uma função de ativação sobre U_i. A saída V_i após um atraso de tempo, é aplicada às entradas dos outros neurônios ponderada por um peso sináptico (T_{ij}), e todas as saídas ponderadas são somadas a uma polarização externa (*bias*) (I_i). As saídas dos neurônios podem ser calculadas em função de U_i utilizando a função logística.

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

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

onde τ é a constante de tempo [8].

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

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

O segundo e o terceiro termo da Equação (1) representam a variação de energia da HNN. Assim, a dinâmica do *i*-ésimo neurônio da HNN pode ser descrita em termos da função de energia:

$$\frac{dU_i}{dt} = -\frac{U_i}{\tau} - \frac{\partial E}{\partial V_i}. \tag{3}$$

2.1. Redes Neurais de Hopfield para Roteamento em Redes de Comunicação

A utilização das HNNs para solucionar o problema do menor caminho a partir de um determinado par origem-destino foi iniciado por Rauch e Winarske [2]. Posteriormente, Ali e Kamoun propuseram um novo algoritmo adaptativo no qual a matriz de pesos apenas leva consigo informações de convergência. As informações sobre custo dos enlaces e topologia são informadas pelas entradas de polarização externa dos neurônios [9]. Ali e Kamoun apresentaram uma forma de modelar o menor caminho a partir do estado final da atingido pela HNN. 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 ou não dos enlaces. Cada elemento na matriz C_{xi} é representado por um neurônio. Desta forma, o número de neurônios da HNN para este problema é $n(n-1)$. Cada neurônio pode ser externamente excitado por uma polarização (*bias*), a partir de uma entrada externa I_{xi}, como definido pela Equação (4):

$$I_{xi} = -\frac{\mu_1}{2}C_{xi}(1-\delta_{xd}\delta_{is}) - \frac{\mu_2}{2}\rho_{xi}(1-\delta_{xd}\delta_{is}) - \frac{\mu_4}{2} + \frac{\mu_5}{2}\delta_{xd}\delta_{is}, \tag{4}$$

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

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

$$\delta_{ab} = \begin{cases} 1, & \text{se } a = b; \\ 0, & \text{caso contrário.} \end{cases} \tag{5}$$

Esta polarização ajusta o nível de excitação da rede inteira e repassa os dados sobre a topologia da rede e o par origem-destino. A matriz de sinapses ($T_{xi,yj}$) é definida por:

$$T_{xi,yj} = \mu_4\delta_{xy}\delta_{ij} - \mu_3\delta_{xy} - \mu_3\delta_{ij} + \mu_3\delta_{jx} + \mu_3\delta_{iy}. \tag{6}$$

Para a atualização das entradas dos neurônios, Bastos-Filho *et al.* [10] propuseram uma simples equação diferença finita em substituição à equação diferencial proposta por Ali e Kamoun, utilizando as entradas prévias do neurônio $U_{xi}[n]$, $U_{xi}[n-1]$ e $U_{xi}[n-2]$. Seguindo essa simplificação, Schuler *et al.* [11] propuseram uma versão ainda mais simplificada da equaçao sem as entradas prévias $U_{xi}[n-2]$. A equação utilizada para atualização das entradas está mostrada abaixo:

$$U_{xi}[n+1] = U_{xi}[n] - AU_{xi}[n-1] + B\sum_{y=1}^{n}\sum_{\substack{j=1 \\ j\neq y}}^{n} T_{xi,yj}V_{yj}[n] + CI_{xi}, \tag{7}$$

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

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

$$Y_{xi} = \begin{cases} 1, & \text{se } V_{xi} \geq 0.5; \\ 0, & \text{se } V_{xi} < 0.5. \end{cases} \tag{8}$$

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 usado para a HNN está apresentado no Algoritmo 1.

Algorithm 1: Pseudocódigo do algoritmo de roteamento utilizando HNN.

```
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} estar abaixo do limiar;
Calcula Y_{xi} (binarização de V_{xi});
Obtém caminho a partir de Y_{xi};
```

3. FPGAs

Matriz de Blocos Lógicos Programáveis em Campo (FPGA, *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 [12]. A estrutura conceitual de um FPGA está apresentada na Figura 1.

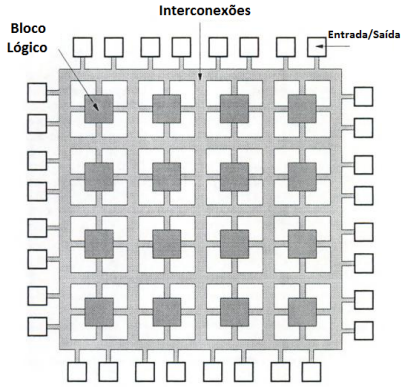

Figura 1: Arquitetura básica de um FPGA.

FPGAs utilizam o conceito de Bloco Lógico Configurável (CLB, *Configurable Logic Block*). 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 a construção de máquinas sequênciais. 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 [12]. Desta maneira, circuitos lógicos são implementados em FPGA partindo a lógica em blocos menores, e depois interconectando-os.

Um FPGA padrão pode ter centenas de milhares de CLBs de diferentes tipos em um único disposito, permitindo a implementação de dispositivos complexos em um único chip. FPGAs modernos têm capacidade de sintetizar processadores de 32-bits em um único dispositivo [13].

4. MODELO PARALELO DE HNN PARA ROTEAMENTO EM REDES DE COMUNICAÇÃO

Redes Neurais apresentam uma variedade de tipos de paralelismo [14]. Entretanto, o foco do modelo proposto é o paralelismo dos neurônios, no qual cada neurônio realiza seu processamento individualmente a partir de suas entradas. O benefício de aproveitar esse paralelismo nas HNN para roteamento de redes de comunicação já foi demonstrado em plataformas GPU [4].

No modelo proposto, o comportamento de cada neurônio é governado por uma máquina de estados, ilustrada na Figura 2(a). 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 neurônios. Essa informação é utilizada como barreira de sincronização entre os estados.

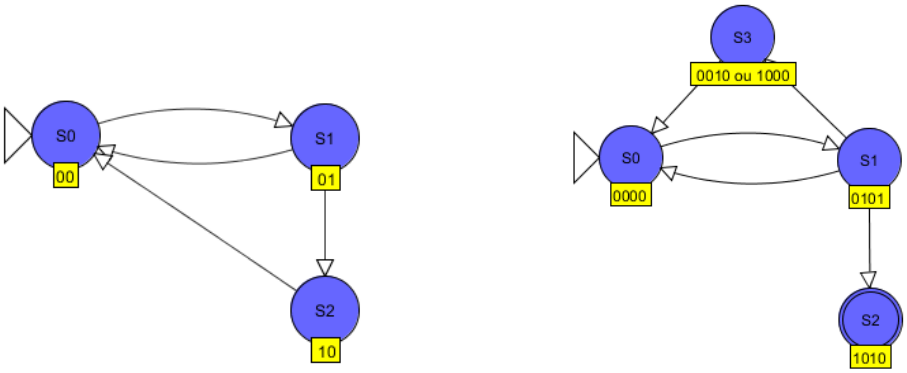

Figura 2: Máquina de estados dos modelos para FPGA: (a) do neurônio e (b) da HNN.

O estado inicial S_0 do neurônio corresponde ao momento da atualização da entrada do neurônio, segundo equação (7). O principal gargalo de todo o algoritmo está neste cálculo. 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 neurônio 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.

Ainda no estado S_1, o teste de limiar do neurônio é efetuado, conforme mencionado no Algoritmo 1. Caso o limiar seja alcançado, o neurônio entra no estado S_2. Caso este estado não seja alcançado, ele vai para o estado S_0. No estado S_2, a saída binarizada é calculada segundo a equação (8).

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. Dessa forma, pode-se dizer que a HNN tem estados baseado nos estados dos neurônios. O diagrama dessa máquina de estados finita para uma rede de dois neurônios é exemplificado na Figura 2(b). 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 quando algum neurônio alcançar limiar e outro não. Este estado está representado na Figura 2(b) como o estado S_3. Quando um neurônio alcançar o limiar e algum outro não atingir, o primeiro neurônio volta para o estado inicial.

4.1. Bloco do Neurônio de HNN para Roteamento de Rede de Comunicação

A Figura 3 ilustra o bloco proposto para cada neurônio da HNN para roteamento de uma rede de comunicação. Esse neurônio é relativo a uma rede neural com 12 neurônios para roteamento em uma rede de comunicação com 4 nós. O bloco tem como entradas as saídas dos outros neurônios, 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 e os sinais de *clock* e *reset*.

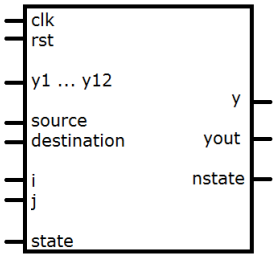

Figura 3: Bloco conceitual com as saídas e entradas de um neurônio de uma HNN para roteamento de uma rede de 4 nós.

4.2. Particularidades no Código

Algumas abordagens utilizadas na codificação do algoritmo de roteamento utilizando HNN devem ser ressaltadas.

A matriz de sinapses $T_{xi,yj}$, definida pela Equação (6), 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 [4].

A matriz de excitação externa dos neurônios I_{xi}, definida pela Equação (4), 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*.

A atualização das entradas dos neurônios dada pela Equação (7) é o principal gargalo na implementação do algoritmo. Devido ao somatório duplo da equação, esse trecho do algorimo pode ter ordem de complexidade $O(n^2)$. Contudo, uma abordagem descrita em [4] deixa o somatório com complexidade $O(n)$. Apesar da diminuição na complexidade, um maior número de comparações é necessário. Dessa forma, as duas abordagens são codificadas a fim de se comparar qual a melhor opção para FPGAs.

5. Precisão Numérica nas Redes Neurais

Nas próximas subseções são apresentados os problemas relacionados à precisão da representação numérica no FPGA e a implementaçao da função de ativação do neurônio no FPGA.

5.1. Representação Numérica

Holt e Banker [15] demonstraram que a representação 16-bits de ponto fixo é a mínima precisão necessária para uma rede MLP utilizando o algoritmo *back-propagation* que utiliza uma saída normalizada no intervalo $[0, 1]$ a partir de uma função sigmóide. Entretanto, não há nenhuma citação direta ao uso de HNNs nesse trabalho.

Por outro lado, é de se esperar que os dois tipos de rede tenham características similares, uma vez que são baseada no mesmo neurônio MCP. Desta forma, a representação utilizada neste projeto foi a de 16-bits de ponto fixo.

5.2. Função de Ativação

A função sigmóide logística é utilizada como função de ativação pelas HNN para Roteamento em Redes de Comunicação. O cálculo dessa funçao é bastante custosa em *hardware*, porém ela pode ser avaliada de formas mais simples. Contudo, essas simplificaçoes implicam em um erro associado ao cálculo da função. As alternativas abordadas no projeto para o cálculo da função sigmóide são: Tabela de Consulta e Interpolação.

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(a) 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 entre $[-4, 4]$ com um variação $\Delta = 0,5$. Um valor específico da função sigmóide é atribuído a cada intervalo.

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(b) ilustra a aproximação com relação a função sigmóide.

Um maior número de intervalos nas duas abordagens significa uma melhor aproximação da sigmóide. Entretanto, uma 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.

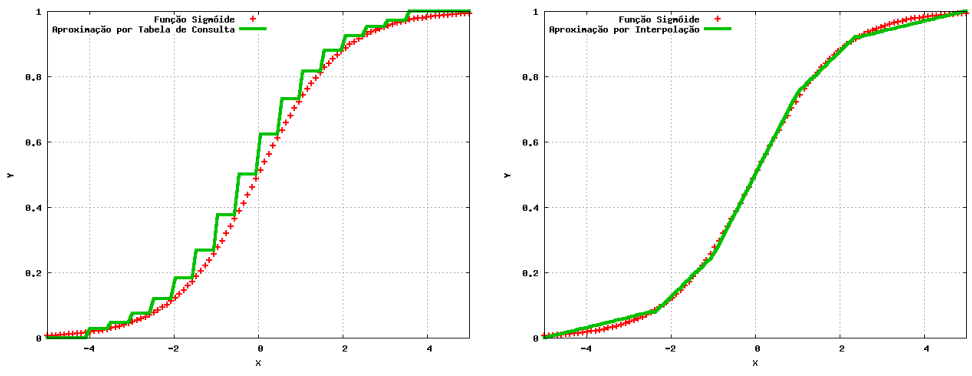

Figura 4: Aproximação da função sigmóide a partir de (a) tabela de consulta e de (b) interpolação.

6. ARRANJO EXPERIMENTAL

O algoritmo de roteamento baseado em HNN foi modelado em VHDL e simulado utilizando a ferramenta *Quartus II 9.1 Web Edition*. O dispositivo alvo configurado na aplicação 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 HNN foram baseados nos valores contidos em [6].

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. O objetivo é analisar a escalabilidade do modelo, relacionando a área utilizada pelo modelo no dispositivo com o número de nós. Também é objetivo deste trabalho relacionar o tempo para convergência da rede com a quantidade de nós.

7. RESULTADOS E ANÁLISES

Nas subseções que seguem, a validação do modelo, as relações de custo de área com as abordagens escolhidas e a comparação de tempos são apresentadas.

7.1. Validação

O valor final na saída de um neurônio indica se o enlace representado pelo neurônio está presente no menor caminho encontrado ou não. 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 diagrama temporal das saídas dos neurônios da HNN para o roteamento na rede de seis nós está apresentado na Figura 5. A requisição tem como origem o Nó 0 e destino o Nó 3. Na convergência, os neurônios 5, 16, 24 e 30 estão ativos e representam, respectivamente, os enlaces $(0, 5)$, $(3, 0)$, $(4, 3)$ e $(5, 4)$. A Figura 6 ilustra o menor caminho encontrado.

Nas três redes simuladas, os caminhos encontrados estão corretos e são os menores caminho possíveis para a requisição. Portanto, a codificação em VHDL da HNN pode ser considerada válida.

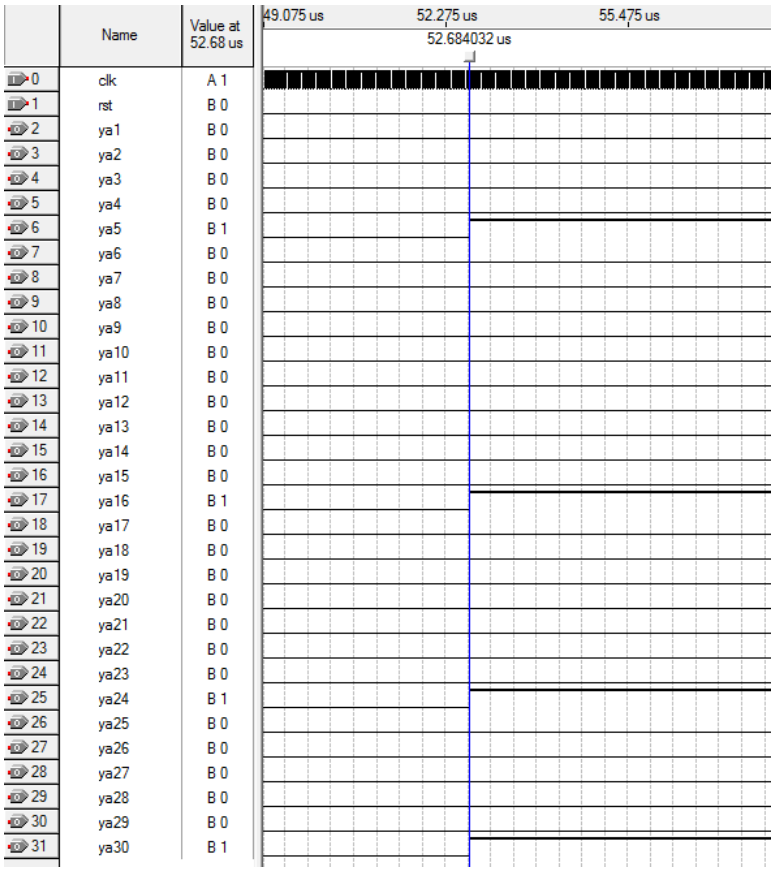

Figura 5: Diagrama temporal das saídas dos neurônios da HNN para roteamento em uma rede de comunicação com seis nós no simulador do Quartus II.

7.2. Custo da Redução da Complexidade da Função de Atualização dos Neurônios

A utilização dos recursos do FPGA para a HNN para roteamento nas três redes está descrita na Tabela 1. São apresentados resultados para três abordagens: as duas abordagens por Tabela de Consulta mencionadas na Sub-seção 4.2, sendo estas a versão otimizada de ordem $O(n)$ e a versão não otimizada de ordem $O(n^2)$; e a abordagem por Interpolação.

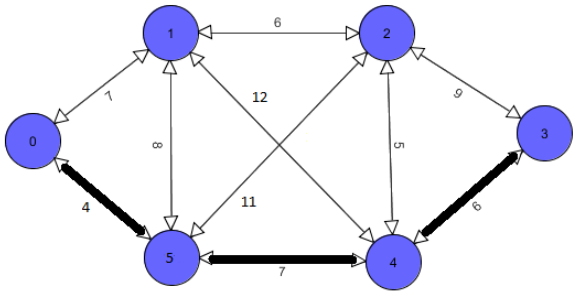

Figura 6: Caminho encontrado pela HNN para a requisiçao entre os nós origem 0 e destino 3 em uma rede de seis nós.

Para a rede com quatro nós, a redução da complexidade tem impacto razoável no uso dos recursos do FPGA. Por outro lado, à medida que o tamanho da rede aumenta, essa otimizaçao já não apresenta bons resultados.

Os erros introduzidos pelas aproximações não afetaram a capacidade de acerto do algoritmo de roteamento. Entretanto, a Interpolação faz uso de muito mais recursos do que a Tabela de Consulta, chegando a ocupar quase 20% mais área no FPGA.

Rede	Recurso do FPGA	Interpolação $O(n)$	Tabela de Consulta $O(n)$	Tabela de Consulta $O(n^2)$
Quatro Nós	ALUTs	7,350 (19%)	5,529 (15%)	6,365 (17%)
	Registros Dedicados	2,425 (6%)	2,221 (6%)	2,569 (7%)
	Lógica Total	25%	18%	21%
Cinco Nós	ALUTs	15,419 (41%)	11,591 (31%)	11,564 (30%)
	Registros Dedicados	4,621 (12%)	3,701 (10%)	4,281 (11%)
	Lógica Total	55%	41%	40%
Seis Nós	ALUTs	26,168 (69%)	20,463 (54%)	18,895 (50%)
	Registros Dedicados	6,693 (18%)	5,551 (15%)	6,421 (17%)
	Lógica Total	96%	78%	70%

Tabela 1: Recursos utilizados no FPGA por cada abordagem de atualização de neurônios.

7.3. Tempo para Convergência

Os tempos médios para convergência da HNN e resposta a uma requisição de rota na rede com quatro nós estão descritos na Tabela 2. Este tempo considera o tempo médio calculado para todas as possibilidades de pares origem-destino na rede de comunicação. Nessa tabela, o tempo é comparado com os valores presentes em [4]. A coluna CPU refere-se ao tempo necessário em uma implementação sequêncial em um computador Quadcore com 8GB de memória RAM, a coluna GPU contém os dados para uma implementaçao paralela na plataforma CUDA modelo GTX 280 [4]. A implementação em FPGA é 78 vezes mais rápida do que a versão sequêncial e 15 vezes mais rápida do que a versão GPU.

Rede	CPU	GPU	FPGA
Seis Nós	2598 μs	480 μs	33,30 μs

Tabela 2: Tempo médio necessário para a HNN convergir na rede com 4 nós.

8. CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho, foi desenvolvido um modelo em VHDL de HNN para roteamento em redes de comunicação. Esse modelo foi criado de forma a tornar a HNN mais eficiente quando sintetizada em dispositivos FPGA. Além disso, as HNNs foram simuladas em uma ferramenta conhecida de simulação com fins de validação do código e avaliação de desempenho. O modelo proposto para a HNN pode ser considerado válido, uma vez que as requisições feitas à rede de comunicação nas simulações foram respondidas corretamente.

As simplificações da função de ativação utilizadas no modelo não implicaram em erros do algoritmo de roteamento. Ademais, a abordagem de Tabela de Consulta, apesar de apresentar uma aproximação ruim da função de ativação, obteve resultados iguais aos resultados obtidos para Interpolação. No entanto, o método baseado na Tabela de Consulta utiliza menos recursos do FPGA. Portanto, ela deve ser a escolhida em implementações futuras.

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

Por fim, o modelo proposto é 78 vezes mais rápido do que a versão sequêncial da HNN em um computador comum com processador Quadcoree 15 vezes mais rápida do que a versão GPU. Portanto, além de ser possível a implementação das HNN em FPGA, elas apresentaram um ótimo desempenho neste tipo de plataforma.

Como trabalhos futuros, deve-se focar na escalabilidade do modelo HNN em FPGA. Uma forma de melhorar esse ponto é diminuir ainda mais a utilização de área do dispositivo. Para isto, pode-se utilizar outras larguras de ponto fixo na representação numérica e outras aproximações ainda menos custosas da função de ativação. Além disso, menos área utilizada significa maior velocidade do modelo implementado.

REFERÊNCIAS

[1] J. J. Hopfield. "Neural networks and physical systems with emergent collective computational abilities." *Proceedings of the National Academy of Sciences of the United States of America*, vol. 79, no. 8, pp. 2554–2558, Abril 1982.

[2] H. E. Rauch and T. Winarske. "Neural networks for routing communication traffic". *Control Systems Magazine, IEEE*, vol. 8, no. 2, pp. 26–31, Abril 1988.

[3] C. J. A. Bastos-Filho, R. A. Santana, D. R. C. Silva, J. F. Martins-Filho and D. A. R. Chaves. "Hopfield Neural Networks for Routing in Optical Networks". *12th International Conference on Transparent Optical Networks, 2010, Munique*, vol. 1, pp. 1 – 6, 2010.

[4] C. J. A. Bastos-Filho, M. A. C. Oliveira Junior, D. R. C. Silva and R. A. Santana. "Optimizing a Routing Algorithm Based on Hopfield Neural Networks for Graphic Processing Units". In *IEEE Symposium on Foundations of Computational Intelligence. IEEE FOCI'11*, 2011.

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

[6] R. A. Santana. *Roteamento em Redes pticas Transparentes Utilizando Redes Neurais de Hopfield*. Dissertao de Mestrado, UPE. 2010, 2010.

[7] W. Mcculloch and W. Pitts. "A logical calculus of the ideas immanent in nervous activity". *Bulletin of Mathematical Biology*, vol. 5, no. 4, pp. 115–133, Dec 1943.

[8] J. J. Hopfield. "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*, vol. 81, no. 10, pp. 3088–3092, May 1984.

[9] M. K. M. Ali and F. Kamoun. "Neural networks for shortest path computation and routing in computer networks". *Neural Networks, IEEE Transactions on*, vol. 4, no. 6, pp. 941–954, Nov 1993.

[10] C. J. A. Bastos-Filho, R. A. Santana and A. L. I. Oliveira. "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*, pp. 363–369, April 2007.

[11] W. H. Schuler, C. J. A. Bastos-Filho and A. L. I. Oliveira. "A novel hybrid training method for hopfield neural networks applied to routing in communications networks". *Int. J. Hybrid Intell. Syst.*, vol. 6, no. 1, pp. 27–39, 2009.

[12] S. Brown, R. Francis, J. Rose and Z. Vranesic. *Field-Programmable Gate Arrays*. Series: The Springer International Series in Engineering and Computer Science, Vol. 180. Springer-Verlag New York, Inc., Secaucus, NJ, USA, fourth edition, 1992.

[13] P. Wilson. *Design Recipes for FPGAs: Using Verilog and VHDL*. Elsevier, first edition, 2007.

[14] A. R. Omondi and J. C. Rajapakse. *FPGA Implementations of Neural Networks*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[15] J. Holt and T. Baker. "Back propagation simulations using limited precision calculations". In *Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on*, volume ii, pp. 121 –126 vol.2, jul 1991.