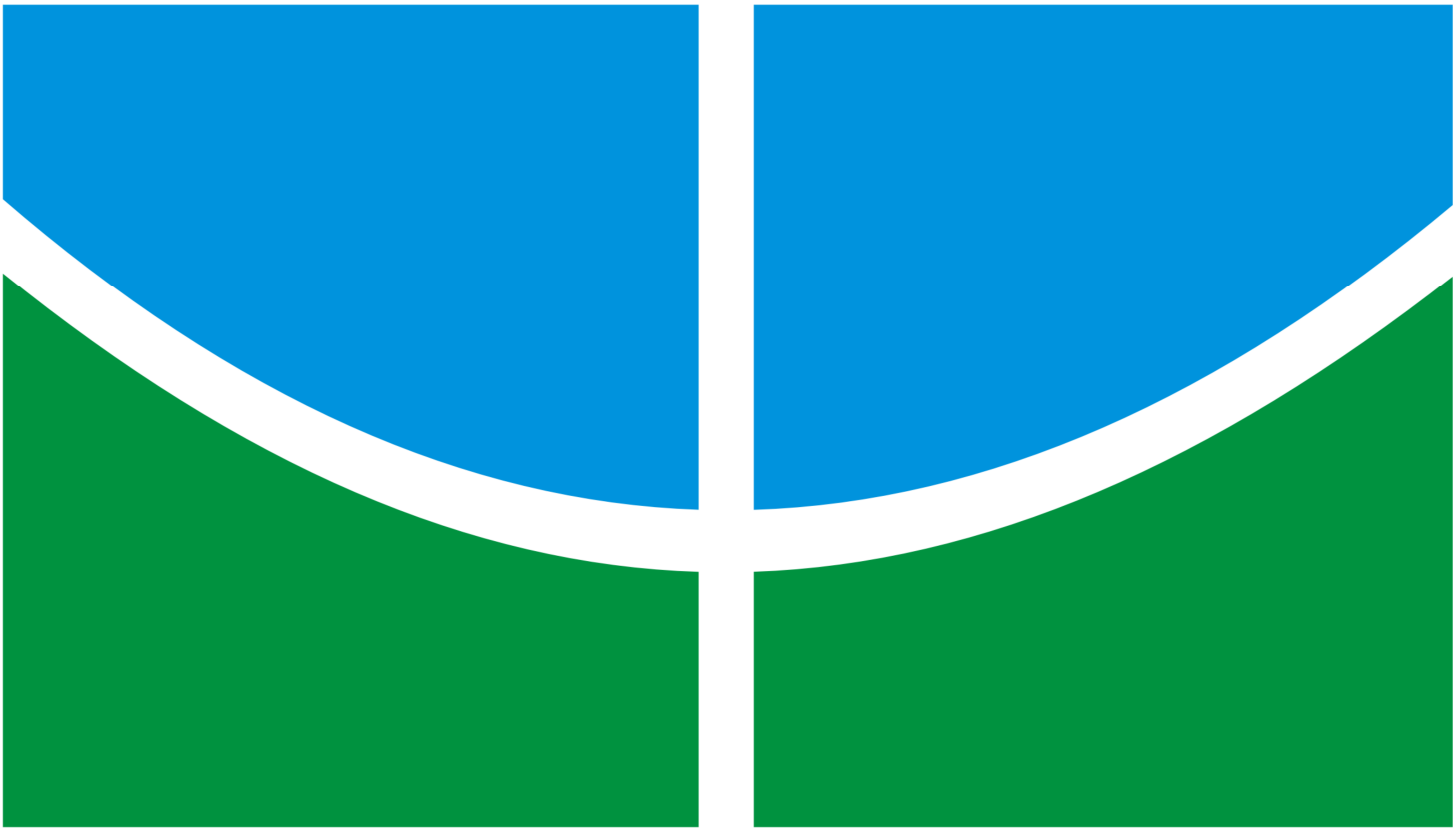

DESENVOLVIMENTO DE UM BLOCO DE CONSTRUÇÃO NANOELETRÔNICO PARA REDES NEURAIS PULSANTES

BEATRIZ DOS SANTOS PÊS

DISSERTAÇÃO DE MESTRADO EM ENGENHARIA DE SISTEMAS ELETRÔNICOS E DE AUTOMAÇÃO DEPARTAMENTO DE ENGENHARIA ELÉTRICA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

DESENVOLVIMENTO DE UM BLOCO DE CONSTRUÇÃO NANOELETRÔNICO PARA REDES NEURAIS PULSANTES

BEATRIZ DOS SANTOS PÊS

DISSERTAÇÃO DE MESTRADO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE.

APROVADA POR:

Prof. Dr. Alexandre Romariz, UnB
(Orientador)

Prof. Dr. Marlio Jose do Couto Bonfim, UFPR
Examinador Externo

Profa. Dra. Janaína Gonçalves Guimarães, UFSC
Examinador Externo

BRASÍLIA, 31 DE OUTUBRO DE 2014.

FICHA CATALOGRÁFICA

PÊS, BEATRIZ DOS SANTOS
Desenvolvimento de um bloco de construção nanoeletrônico para Redes Neurais Pulsantes [Distrito Federal] 2014.
xi, 97p., 210 x 297 mm (ENE/FT/UnB, Engenheiro Eletricista, Engenharia Elétrica, 2014).
Dissertação de Mestrado – Universidade de Brasília, Faculdade de Tecnologia.
Departamento de Engenharia Elétrica

1. Nanoeletrônica	2. Redes Neurais
3. Redes Cross	4. Neurônios Pulsantes
I. ENE/FT/UnB	II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

PÊS, B.S. (2014). Desenvolvimento de um bloco de construção nanoeletrônico para Redes Neurais Pulsantes, Dissertação de mestrado em Engenharia de Sistemas Eletrônicos e de Automação, Publicação 578/14 DM/PGEA, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 97p.

Beatriz dos Santos Pês
Departamento de Eng. Elétrica (ENE) - FT
Universidade de Brasília (UnB)
Campus Darcy Ribeiro
CEP 70919-970 - Brasília - DF - Brasil

Dedico este trabalho a minha família, pelo suporte em todos os momentos, e a Profa. Janaína Guimarães, por ser uma pessoa excepcional.

AGRADECIMENTOS

"Não pense que a cabeça aguenta, se você parar."
Raul Seixas

Neste trabalho tenho muitos obrigadas para distribuir, vou deixar aqui registrada minha gratidão para com pessoas absolutamente especiais e que marcaram minha trajetória com apoio, carinho e incentivo.

Aos meus pais, por me verem como uma super heroína, capaz de tudo.

Às minhas irmãs, por serem a própria definição da palavra "irmã". Amigas, companheiras, confidentes, enfim, muito djanhas!

Ao meu marido, que esteve ao meu lado o tempo todo. Que suportou os momentos difíceis, que me desafiou, incentivou e me fez desejar continuar. Muito obrigada.

À colega, Beatriz Câmara, obrigada por ser a minha "matéria escura".

Ao colega, Alex Yuzo Moroguma, pela companhia nos congressos e por compartilhar conhecimentos e templates no Latex!

À minha orientadora, Janaína Guimarães, por quem a minha admiração só cresce. Obrigada por ser tão especial, tão atenciosa e cuidadosa. Você faz seus alunos sentirem-se seguros e determinados. Eu não poderia ter encontrado uma orientadora melhor!

Hantalë!

RESUMO

DESENVOLVIMENTO DE UM BLOCO DE CONSTRUÇÃO NANOELETRÔNICO PARA REDES NEURAIS PULSANTES

Autor: Beatriz dos Santos Pês

Orientador: Prof. Dr. Alexandre Romariz, UnB

Departamento de Engenharia Elétrica

Brasília, 31 de outubro de 2014

A habilidade de simular de forma mais realista o comportamento do cérebro humano fez com que as redes neurais pulsantes (SNNs, *Spiking Neural Networks*) se tornassem populares entre os pesquisadores. Estes circuitos, altamente densos, apresentam grande capacidade de processamento de dados. Inicialmente, acreditava-se que arquiteturas reconfiguráveis, como FPGAs, *Field Programmable Gate Arrays*, poderiam ser usadas como protótipos para a construção de SNNs. Entretanto, FPGAs não suportam os altos níveis de conectividade entre neurônios em uma SNN densa. Além disso, a implementação através de FPGAs não fornece melhorias quanto à dissipação de potência ou área ocupada. Por este motivo, os pesquisadores começaram a utilizar NoCs, *Networks-on-Chip*, para interconectar SNNs. O uso de NoCs é capaz de reduzir o número de interconexões e apresenta uma grande vantagem relativa à tolerância a falhas: redundância. Neste contexto, algumas configurações combinando neurônios e roteadores foram propostas. Estes componentes constituem o bloco básico, presente em cada nó da NoC. Vários modelos de neurônios pulsantes e vários algoritmos de roteamento foram usados. Todas estas propostas buscam a implementação de redes cada vez mais densas, reduzindo a dissipação de potência e a área ocupada. No entanto, nenhum dos trabalhos anteriores usa um modelo nanoeletrônico para o neurônio pulsante. A implementação nanoeletrônica é bem conhecida pelos ganhos que apresenta justamente nesses dois parâmetros: dissipação de potência e área ocupada. Assim, este trabalho propõe um bloco básico de construção para ser utilizado em uma NoC do tipo 2D *mesh*. Este bloco consiste de um neurônio pulsante nanoeletrônico conectado a um roteador, implementado através de uma LUT, *Look-Up Table*. Primeiramente, o modelo do neurônio foi redimensionado para funcionar a 300 K, a temperatura ambiente. Depois, o comportamento do neurônio foi testado através da implementação de várias portas lógicas, tais como inversora, OU, E e XOR. Um elemento roteador simples é, então, proposto a fim de construir o primeiro bloco para a NoC. Para testar a funcionalidade deste bloco, uma XOR com 2 entradas foi apresentada para a SNN construída com este bloco. Finalmente, um roteador capaz de comunicar neurônios em 4 direções foi proposto e um bloco de construção para a NoC com este roteador foi implementado. O problema da XOR, com 3 e com 5 entradas, foi usado para validar a funcionalidade deste bloco.

ABSTRACT

A NANOELECTRONIC BUILDING BLOCK FOR SPIKING NEURAL NETWORKS

Author: Beatriz dos Santos Pês

Supervisor: Prof. Dr. Alexandre Romariz, UnB

Departamento de Engenharia Elétrica

Brasília, 31th October 2014

The ability to emulate more realisticaly the behavior of the human brain made Spiking Neural Networks (SNNs) gain prominence between researchers. These highly dense circuits feature large capacity of data processing. Searching for reconfigurable devices, computer scientists and engineers used Field Programmable Gate Arrays (FPGAs) as prototypes for SNNs. However, FPGAs cannot support the highlevels of connectivity between neurons in a dense SNN. Besides, implementation with FPGA does not provide improvements regarding power dissipation or scale. Therefore, researchers began to use Networks-on-Chip (NoCs) to interconnect SNNs. The use of NoCs may reduce the number of interconnections and presents a big advantage regarding fault tolerance: redundancy. In this context, several configurations combining neurons and routers were proposed. These devices constitute the basic block, present in every node of the NoC. Various models of spiking neurons were used, combined with various routing algorithms. All these proposals aim the implementation of denser networks, reducing the power dissipation and the occupied area. However, none of the previous works uses a nanoelectronic model for the spiking neuron. Nanoelectronic implementation is well known for the gains that it presents precisely in these two parameters: occupied area and power dissipation. Thus, this work proposes a basic block for a 2D-mesh NoC, consisting of a nanoelectronic spiking neuron connected to a router, implemented with a Look-Up Table (LUT). First, the model for the nanoelectronic neuron is scaled in order to work at 300 K, the room temperature. Then, the behaviour of the neuron is tested through the implementation of various logic gates, such as NOT, AND, OR and XOR gates. A simple routing element is proposed to construct the first building block. In order to test the functionality of this block, a 2 inputs XOR problem is presented to a SNN implemented with this block. Finally, a full directional router is proposed and a building block using this router is implemented. The XOR problem, with 3 and with 5 inputs, is used to validate the functionality of this block.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

Siglas

NoCs	Networks-on-Chip
XOR	Exclusive OR
MOS	Metal-Oxide Semiconductor
MOSFET	Metal-Oxide Semiconductor Field Effect Transistor
SET	Single-Electron Tunneling Transistor
RTD	Resonant Tunneling Diode
QD	Quantum Dots
SNNs	*Spiking Neural Networks*
FPGAs	*Field Programmable Gate Arrays*
CMOS	*Complementary Metal Oxide Semiconductor*
SoC	*System on Chip*
EMBRACE	*EMulating Biologically-inspiRed ArChitectures in hardwarE*
SPICE	*Simulation Program with Integrated Circuit Emphasis*
LTSPICE	*Linear Technology* SPICE
PSPICE	*Personal* SPICE
SIMON	*Simulation of Nanostructures*
LUT	*Look-Up Table*
FCFS	*First Come First Served*
SECS	*Single-electron Circuit Simulator*

1 INTRODUÇÃO

"Nem tudo o que é ouro fulgura.
Nem todo o vagante está perdido.
O velho que é forte perdura (...)
Das sombras a luz irá jorrar."
J. R. R. Tolkien

Pouco antes do início dos anos 2000, estimou-se que o limite da tecnologia utilizada nos transistores seria alcançado em 2010. Neste ano, o comprimento mínimo do *gate* (terminal de porta, no qual é aplicada uma tensão que definirá o comportamento do dispositivo: corte, saturação ou região ativa) foi estimado em 70 nm [1]. Hoje, pode-se afirmar que este limite foi vencido: em 2011, a Intel lançou a família *core i7* que possui transistores com *gates* de 32 nm e, em 2014, a mesma empresa apresentou a família *Ivy Bridge*, com *gates* de 22 nm. Este avanço se deveu a pesquisas impulsionadas pela promessa de circuitos cada vez menores e mais rápidos. Tais pesquisas resultaram em tecnologias inovadoras, que possibilitaram a continuação da tendência de miniaturização dos circuitos enunciada por Gordon Moore, em 1965. A nanoeletrônica aparece como uma destas tecnologias.

Estudos exploratórios especulam o desenvolvimento de circuitos CMOS com *gates* de até 4 nm [2]. Contudo, a nanoeletrônica apresenta vantagens que vão além da simples redução da área [3]. Os dispositivos nanoeletrônicos já propostos, como o transistor de tunelamento mono-elétron (SET, *Single-Electron Tunneling Transistor*), têm seu funcionamento regido pela física quântica [4]. Esta característica permite, por exemplo, a condução de corrente via tunelamento, um fenômeno quântico que possibilita o transporte de elétrons de forma individual [5]. Dessa forma, o SET possui ótimo controle de corrente, o que reduz consideravelmente a potência consumida pelo dispositivo [6]. A alta velocidade de chaveamento, a consequente operação em altas frequências (na ordem dos GHz), o baixo consumo de potência e a redução da área ocupada fazem da nanoeletrônica uma tecnologia promissora [7].

Dentre as várias aplicações já desenvolvidas com circuitos nanoeletrônicos, destacam-se as propostas de redes neurais, que aplicam dispositivos nanoeletrônicos como componentes de neurônios artificiais. Guimarães [5] desenvolve o modelo para uma rede de Hamming mono-elétron, e Akazawa *et. al* [8] propõem a rede de Hopfield mono-elétron. Esta rede foi, posteriormente, redimensionada por Nogueira [9], para que funcionasse à temperatura ambiente. As redes mono-elétron de Hamming e de Hopfield têm em comum o fato de serem classificadas como redes neurais tradicionais. Ou seja, seus neurônios não possuem a característica de transportar a informação através de pulsos. A seguir serão discutidos alguns aspectos das chamadas redes neurais pulsantes (SNNs, *Spiking Neural Networks*).

Características únicas do cérebro humano como tolerância a falhas, alto nível de para-

lelismo e baixo consumo de potência são muito desejáveis em circuitos eletrônicos [10]. A tentativa de reproduzi-las levou ao desenvolvimento das redes neurais artificiais (RNA) [11]. Durante anos, vários modelos de neurônios artificiais e arquiteturas de redes neurais foram propostos. As aplicações destes circuitos, ditos inteligentes, cresceram consideravelmente. Dentre estas destacam-se aplicações em identificação de imagens, que podem ser úteis em várias áreas: Oroski [12] utiliza uma rede neural para identificar falhas em espaçadores de linhas de alta tensão através de fotografias apresentadas à rede. Outra aplicação frequente reside na identificação de imagens médicas em diagnósticos por imagem [13].

O grande sucesso alcançado pelas RNA fez com que as pesquisas na área ganhassem proeminência. Assim, pesquisadores começaram a buscar modelos de neurônios capazes de simular de maneira mais realista o comportamento do neurônio biológico [14]. Atualmente, esta busca levou os cientistas até o modelo de neurônio pulsante e às redes neurais pulsantes [15]. Estas redes interagem utilizando pulsos e, portanto, reproduzem melhor o comportamento do cérebro humano [10] [14]. As redes neurais pulsantes apresentam potencial para a construção de sistemas altamente densos (com um número elevado de neurônios), massivamente paralelos e totalmente interconectados. Este sistema possuiria uma capacidade de processamento bastante elevada e, devido ao alto grau de paralelismo, boa tolerância a falhas [10]. Ou seja, seria tudo aquilo que os pesquisadores da área de redes neurais artificiais têm buscado. Contudo, a construção física de um sistema com todas estas características tem se mostrado desafiadora [14].

Sistemas densos devem ser, em algum nível, tolerantes a falhas para que haja confiabilidade [14]. A fim de se obter circuitos robustos e com grande capacidade de processar sinais, pesquisadores buscaram formas reconfiguráveis de se implementar as SNNs. Desse modo, os FPGAs, *Field Programmable Gate Arrays*, pareceram a escolha óbvia [16].

Quando os primeiros FPGAs foram introduzidos, em 1980, já se pretendia a implementação de circuitos grandes e de alta performance [17]. De forma geral, um FPGA consiste em vários módulos lógicos programáveis independentes, que podem ser interconectados para criar funções maiores [18]. Os blocos podem conter registradores e há necessidade de blocos de memória que armazenem as conexões [17].

Entretanto, apesar da característica reconfigurável e do paralelismo inerente, os FPGAs mostraram-se incapazes de suportar a densidade de conexões necessária para a construção de SNNs com alto número de neurônios e totalmente interconectadas. Na verdade, o mapeamento das SNNs através de blocos lógicos, efetuado pelos FPGAs, limita o número de neurônios que a rede pode conter, uma vez que estes blocos não são eficientes quanto à dissipação de potência e não otimizam a área ocupada [19]. Além disso, as estruturas de roteamento presentes nos FPGAs não conseguem atender o número elevado de interconexões presente em SNNs densas [20].

A próxima tentativa de construção de SNNs via *hardware* consistiu na utilização de bar-

ramentos. Embora a topologia de barramento compartilhado pareça uma boa escolha para emular a interconexão entre neurônios, esta aproximação também se mostrou ineficiente. Para uma rede totalmente interconectada, o número de barramentos necessários para ligar os neurônios é proporcional ao produto entre o número de neurônios da camada pré-sináptica e o número de neurônios da camada pós-sináptica. Esta característica faz com que a área ocupada cresça de forma exponencial, dependendo do número de neurônios da rede [10].

Com o insucesso das duas primeiras tentativas, teve início a busca por uma solução capaz de suportar altos níveis de interconexão entre os componentes, apresentando bom rendimento e escalabilidade. Desse modo, foi proposto o chamado paradigma de interconexão NoCs, *Networks-on-Chip* [21]. As NoCs são redes de interconexões que possuem, em cada nó, um elemento processador e um roteador. Devido ao elemento roteador, uma NoC é capaz de reduzir o número de interconexões necessário. Assim, reduz também o consumo de potência e a área ocupada. Como o modelo é inerentemente redundante, a NoC apresenta melhorias no que concerne a tolerância a falhas [22]. Sendo assim, vários circuitos já foram propostos para a realização de SNNs com NoCs. Estes circuitos diferenciam-se pelo modelo de neurônio pulsante, pelo algoritmo de roteamento utilizado e pela arquitetura da NoC. Dentre estes pode-se citar: EMBRACE [14], SpiNNaker [23], FACETS [24] e a proposta de Theocharides *et. al* [25]. Estas aproximações conseguiram redes de 108 a até 180 mil neurônios.

Nenhum dos trabalhos anteriores, contudo, apresenta a utilização de um modelo de neurônio pulsante nanoeletrônico. Uma vez que um sistema denso deve ter consumo eficiente de potência e uma área reduzida, a utilização da nanoeletrônica é atrativa. Portanto, neste trabalho, será proposta a implementação de um bloco de construção para SNNs. Este bloco será composto de um neurônio pulsante nanoeletrônico, como elemento processador, e, seguindo a implementação de Harkin *et. al* [14], o elemento roteador será construído através de uma LUT, *Look-Up Table*. A validação do comportamento do bloco é feita através do *benchmark* da função lógica XOR, que será utilizado com 2, 3 e 5 entradas. A implementação das SNNs para a solução destes problemas atingiu o número de 44 neurônios para 5 entradas.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

O objetivo principal deste trabalho é desenvolver um bloco nanoeletrônico para construção de redes neurais pulsantes. Este bloco, constituído por um neurônio nanoeletrônico pulsante e um roteador, será colocado em cada nó de uma NoC do tipo 2D-mesh. Para validar o comportamento do bloco foi utilizado o *software* LTSpice IV.

1.1.2 Objetivos Específicos

Para que se consiga chegar ao objetivo geral deste trabalho, foram necessários vários passos intermediários, aqui chamados de objetivos específicos:

1. Validar o comportamento do modelo de neurônio escolhido à temperatura ambiente (300 K);

2. Implementar portas lógicas (inversora, OU, E e XOR) a fim de testar o funcionamento conjunto de mais de um neurônio;

3. Desenvolver um modelo simplificado de roteador, apenas para realizar a interconexão entre os neurônios;

4. Conectar este protótipo simplificado de roteador ao neurônio pulsante, criando o bloco de construção. Apresentar o *benchmark* da XOR com duas entradas a uma rede neural pulsante construída com este bloco;

5. Desenvolver um roteador capaz de interconectar neurônios em todas as direções, utilizando endereçamento através de uma LUT, *Look-Up Table*;

6. Apresentar o *benchmark* da XOR, com três entradas, para uma rede neural e testar a funcionalidade do conjunto com o novo roteador;

7. Apresentar o *benchmark* da XOR, com cinco entradas, a fim de ilustrar a flexibilidade do roteador desenvolvido através da LUT. Demonstrando, assim, a capacidade de adequar o roteador à necessidade do usuário.

1.1.3 Organização do Trabalho

Este trabalho foi dividido em 5 capítulos, incluindo a introdução. O conteúdo de cada capítulo será brevemente explicado a seguir.

No capítulo 2 é feita uma introdução que detalha algumas vantagens apresentadas pela nanoeletrônica na implementação de circuitos integrados. Além disso, são explicitados os conceitos básicos necessários para o bom entendimento do trabalho. Estes conceitos foram, com o objetivo de conferir maior clareza, divididos nos seguintes tópicos:

- Dispositivos de tunelamento mono-elétron. Esta seção compreende as seguintes subseções: ilhas, tunelamento, efeito de carregamento, bloqueio de Coulomb e transistor mono-elétron. Cada uma delas destinada a explicar um fenômeno ou elemento indispensável ao funcionamento do transistor mono-elétron, o principal dispositivo que compõe o modelo de neurônio utilizado neste trabalho;

- Redes Neurais Artificiais. Compreendendo as seguintes subseções: neurônio artificial, redes neurais pulsadas e neurônios pulsantes;
- *Networks-on-Chip* - NoCs, que apresenta as principais ideias do paradoxo de interconexão representado pelas NoCs e mostra algumas de suas principais vantagens na interconexão de componentes em um chip;
- Roteamento de Informação. Este tópico é dividido nas seguintes subseções: princípio de otimização, roteamento pelo caminho mais curto, roteamento por inundação ou *flooding*, roteamento com vetor de distância, roteamento por estado de enlace, algoritmos *Round Robin* e *First Come First Served - FCFS* e, finalmente, roteamento via LUT. Cada seção explica, de forma básica, um algoritmo de roteamento e apresenta suas vantagens e desvantagens;
- *EMBRACE*, esta seção é destinada a apresentar a proposta de Harkin *et. al*, que inspirou a realização deste trabalho;
- Ferramentas de Simulação, destinada a explicar rapidamente a interface do *software* LTSpice, versão IV, utilizado para realizar as simulações deste trabalho. É realizada, também, uma comparação entre os modelos disponíveis para o SET nas ferramentas SIMON, SECS e LTSpice e uma discussão sobre os parâmetros de simulação do LTSpice IV.
- Análise Crítica do modelo de Wen-peng *et. al*, esta seção discorre sobre o modelo de neurônio escolhido, suas vantagens e limitações.

No capítulo 3 são detalhados os procedimentos realizados a fim de efetuar a validação de comportamento para o modelo de neurônio pulsante de Wen-peng *et. al*. Este capítulo foi dividido nas seguintes seções:

- Modelo de Wen-peng a temperatura ambiente. Esta seção apresenta o procedimento utilizado no redimensionamento do neurônio pulsante para que ele funcione a $T = 300$ K;
- Codificação da informação, que mostra as dificuldades encontradas e as soluções propostas para que uma cdificação padrão possa ser utilizada;
- Projeto de Portas Lógicas. Dividida nas seguintes subseções: NOT - inversora, OR - OU, NAND - NÃO-E, AND - E e XOR - Ou exclusivo. Cada uma delas explica como estas portas foram realizadas com a utilização do neurônio pulsante.

No capítulo 4 são apresentadas as simulações dos circuitos das redes neurais artificiais utilizadas baseadas em NoCs. Estas redes foram construídas com objetivo de validar a funcionalidade do bloco de construção proposto. Este capítulo é constituído pelas seguintes seções:

- Roteador Simples, dedicada à apresentação do primeiro protótipo de roteador implementado;
- XOR de duas entradas com roteador simples, cujo objetivo é apresentar como foi validada a funcionalidade do bloco neural composto pelo neurônio e pelo roteador simples;
- XOR com 3 entradas, que ilustra os procedimentos seguidos para a construção deste circuito;
- XOR com 5 entradas, idem ao anterior;
- Roteador *full*, explica a construção do roteador capaz de interconectar neurônios nas quatro direções;
- XOR de 3 entradas com roteador *full*, que explicita como foi realizada a validação do comportamento do bloco de construção desenvolvido e;
- XOR de 5 entradas com roteador *full*, cujo objetivo é ilustrar a flexibilidade fornecida pela implementação do roteador via LUT.

O capítulo 5 apresenta uma discussão dos resultados obtidos neste trabalho. Por fim, o capítulo 6 apresenta as principais conclusões deste trabalho, bem como as expectativas de realizações futuras.

2 FUNDAMENTAÇÃO TEÓRICA

"Podemos facilmente perdoar uma criança que teme a escuridão.
A real tragédia da vida são os homens que temem a luz."
Platão

2.1 NANOELETRÔNICA

A nanoeletrônica vem sendo cada vez mais explorada por pesquisadores. Esse avanço das pesquisas na área se deve, principalmente, à antecipação do conhecimento de que a tecnologia MOS, *Metal-Oxide Semiconductor*, atingirá seu limite [5]. A busca por dispositivos cada vez mais rápidos coincide com a busca por um processo produtivo mais barato [26]. Estes dois requisitos são facilmente atendidos pela redução da dimensão do *gate* do transistor: quanto menor ele for, mais transistores caberão em um *chip* e mais rápido será o chaveamento do dispositivo [27]. A redução do *gate* do transistor é a imposição que pode fazer com que o MOSFET, *Metal-Oxide Semiconductor Field Effect Transistor*, perca sua posição de destaque na indústria eletrônica. Como um exemplo prático pode-se citar a Intel, que já comercializa processadores com a tecnologia de 22 nm (nova geração *Ivy Bridge* com transistores 3D).

Atualmente, há vários projetos que integram as duas tecnologias: o processamento efetuado em tecnologia nanométrica e as interfaces em tecnologia MOS [28] [14] [10]. Este fato demonstra que a indústria eletrônica adentra uma área de transição e que, em um futuro não muito distante, deve chegar ao domínio da nanoeletrônica. Até lá muitos desafios ainda precisam ser vencidos pelos pesquisadores, tais como o funcionamento correto de um circuito puramente nanoeletrônico em temperatura ambiente e as questões de fabricação em escala nanométrica [29].

Os dispositivos nanoeletrônicos possuem várias características extremamente atrativas. Dentre elas, a capacidade de manter o chaveamento dos dispositivos MOS com redução da área ocupada e da potência dissipada [5]. Além disso, a dimensão reduzida faz com que o dispositivo possa chavear em frequências mais altas que os dispositivos MOS, garantindo, assim, um ganho considerável na capacidade de processamento [30]. Dentre os principais dispositivos nanoeletrônicos já propostos, pode-se citar o transistor de tunelamento monoelétron (SET, *Single-Electron Tunneling Transistor*) [31], os diodos de tunelamento ressonante (RTD, *Resonant Tunneling Diode*) [32] e os pontos quânticos (QD, *Quantum Dots*) [33].

Neste trabalho, o SET será utilizado como principal elemento de um neurônio em uma

rede neural pulsante. Sendo assim, a próxima seção é dedicada à explanação do funcionamento deste dispositivo.

2.2 DISPOSITIVOS DE TUNELAMENTO MONO-ELÉTRON [5]

A corrente elétrica é capaz de fluir em materiais considerados condutores. Esta corrente é definida pela quantidade de carga transferida por unidade de tempo, conforme a Equação 2.1.

$$I = \frac{dQ}{dt} \tag{2.1}$$

em que, I é a corrente, Q é a carga elétrica e t é o tempo.

Em condutores comuns, a carga elétrica transferida pode ter qualquer valor múltiplo da carga do elétron. Portanto, pode-se dizer que a corrente elétrica não é quantizada.

Em dispositivos mono-elétron, a quantidade de carga transferida, dQ, é quantizada. Sendo assim, idealmente, o dispositivo poderia controlar o transporte de um único elétron. Na prática este controle não é tão preciso, de modo que um pequeno fluxo de elétrons passa através do dispositivo quando há condução.

Para que o controle da quantidade de elétrons fluindo durante a condução seja efetivo, é necessário um dispositivo através do qual as cargas possam fluir de maneira discreta. Este dispositivo, chamado de junção-túnel, é apresentado na Figura 2.1.

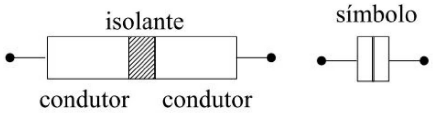

Figura 2.1: Junção-túnel [5].

A junção-túnel é descrita por dois parâmetros macroscópicos: sua resistência de tunelamento (R_T) e a capacitância da junção (C_j). Estes parâmetros são de extrema importância no comportamento do dispositivo. Seus valores podem, por exemplo, determinar se o dispositivo funciona ou não à temperatura ambiente, 300 K. Para um correto funcionamento da junção-túnel, tanto R_T quanto C_j precisam satisfazer algumas condições que serão apresentadas na seção 2.2.3.

2.2.1 Ilhas

A ilha é uma região constituída de material condutor cercado por finas barreiras isolantes. Essas barreiras impedem a movimentação de elétrons através da ilha. Desse modo, o único meio de transporte possível para os elétrons é o tunelamento (seção 2.2.2). Por esse motivo, a ilha é também conhecida como a região de confinamento de elétrons. A Figura 2.2 ilustra dois eletrodos separados por uma ilha.

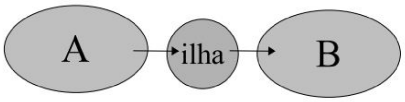

Figura 2.2: Eletrodos A e B separados por uma ilha [5].

Para se movimentarem do eletrodo A até o eletrodo B, os elétrons devem tunelar duas vezes: primeiro para dentro da ilha e, depois, para fora dela. No caso ideal, apenas um elétron por vez tunela através da ilha. Sendo assim, a carga na ilha terá uma variação exatamente igual a e - a carga elementar, $e = 1,6 \cdot 10^{-19}$ C. Em dispositivos eletrônicos comuns, pacotes de carga chegam a conter 10^6 elétrons, de modo que essa pequena variação de carga pode parecer desprezível. Contudo, em dispositivos nanoeletrônicos, as dimensões da ilha são tão pequenas que mesmo essa pequena variação gera potencial suficiente para influenciar as probabilidades de tunelamento.

2.2.2 Tunelamento

O tunelamento é o movimento de cargas que se dá em uma região classicamente proibida. Essa proibição advém de barreiras de energia potencial maiores do que a energia total da partícula pontual que tenta atravessá-las. Dessa forma, a partícula não possui energia suficiente para atravessar a barreira [9]. A Figura 2.3 ilustra o tunelamento de uma partícula, que possui energia igual a *E*, através de uma barreira de potencial $V_0 > E$.

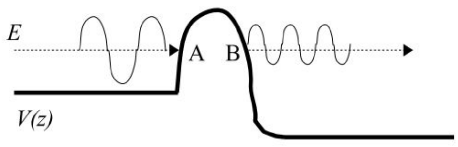

Figura 2.3: Tunelamento através de uma barreira de potencial. Os pontos A e B são classificados como pontos críticos clássicos [5].

Uma partícula clássica seria totalmente refletida nos pontos A e B, representados na Figura 2.3. Contudo, o elétron possui natureza dual partícula/onda, de modo que ele atravessa a barreira como uma onda. Para melhor explicar esse conceito, são necessárias algumas considerações vindas da mecânica quântica. Por exemplo, na fronteira entre duas regiões com energias potenciais diferentes sabe-se que a equação de onda e sua derivada devem ser

contínuas. Ou seja, a equação de onda não pode se extinguir na barreira de potencial. Dessa forma, haverá transmissão e reflexão parciais da onda incidente. Como o quadrado da função de onda de um elétron traduz a densidade de probabilidade de se encontrar essa partícula em uma dada região, conclui-se que há uma probabilidade finita de se encontrar o elétron do outro lado da barreira.

Averin e Likharev propuseram, em 1987, o modelo mais utilizado para efetuar a análise do tunelamento mono-elétron. Esse modelo, conhecido como Teoria Ortodoxa do Tunelamento Mono-elétron, é composto de 4 postulados, explicitados a seguir [5].

1. Modelo de dimensão zero: as dimensões das ilhas são desprezíveis;
2. O evento de tunelamento é instantâneo;
3. A redistribuição das cargas após o tunelamento também é instantânea;
4. O espectro de energia em condutores e ilhas é considerado contínuo: a quantização da energia do elétron é ignorada dentro dos condutores.

Com este modelo o efeito de carregamento chamado de Bloqueio de Coulomb (seção 2.2.4) é descrito quantitativamente. O principal resultado destes postulados é que a taxa com que o tunelamento ocorre depende fortemente da mudança de energia livre causada pelo tunelamento [34].

2.2.3 Efeito de carregamento

O efeito de carregamento ocorre quando os elétrons estão prestes a entrar na ilha (seção 2.2.1) e é a base do funcionamento dos dispositivos mono-elétron [9]. Uma vez que as dimensões da ilha são muito pequenas, o tunelamento de um único elétron para dentro da ilha causa uma variação de potencial considerável, 160 mV para C_j = 1 aF [35]. Esse aumento de potencial, que dá origem ao Bloqueio de Coulomb (seção 2.2.4), cessa o fluxo de elétrons pela ilha. Desse modo, outro elétron só poderá tunelar para dentro da ilha, se uma tensão de polarização externa, que reduza o potencial da ilha, for aplicada.

Para que os dispositivos mono-elétron funcionem da maneira esperada, é essencial que o efeito de carregamento dite seu comportamento. Sendo assim, serão apresentados os requisitos para que os efeitos de carregamento controlem o transporte de elétrons em um dispositivo. O primeiro deles deriva do Princípio da Incerteza de Heisenberg, expresso na Equação 2.2.

$$\Delta E.\Delta t \geq \frac{h}{2\pi}, \tag{2.2}$$

em que, ΔE é a incerteza quanto à energia, Δt é a incerteza quanto ao tempo e h = 6,626068 $\cdot 10^{-34}\ m^2.kg/s$ é a constante de Planck.

O tempo característico para flutuações de carga será dado pela constante de tempo da junção, definida na Equação 2.3.

$$\Delta t \cong R_T.C, \tag{2.3}$$

em que, R_T é a resistência de tunelamento e C é a capacitância de carregamento da junção.

A incerteza relativa à energia é dada pela variação de energia associada a um elétron em excesso, conforme a Equação 2.4.

$$\Delta E = \frac{e^2}{C} \tag{2.4}$$

A substituição das equações 2.3 e 2.4 em 2.2 resulta na primeira condição que deve ser satisfeita a fim de que o efeito de carregamento controle o fluxo de cargas no dispositivo mono-elétron. Essa condição, expressa na Equação 2.5, impõe um valor mínimo para R_T. A partir desse valor, a natureza corpuscular do elétron predominará sobre a ondulatória.

$$R_T > \frac{\hbar}{e^2}, \tag{2.5}$$

em que $\hbar$ é a constante de Planck dividida por 2π.

Assim, o valor mínimo para R_T, aqui chamado de R_K, equivale a:

$$R_K = \frac{\hbar}{e^2} = 25{,}8\ k\Omega \tag{2.6}$$

A segunda condição diz respeito às dimensões da ilha. Para uma dada temperatura, T, estas dimensões devem ser tais que a energia eletrostática associada, E_C, seja muito maior que as flutuações térmicas existentes àquela temperatura:

$$E_C >> k_B \cdot T, \tag{2.7}$$

em que, $k_B = 1,3806488 \cdot 10^{-23}\ m^2\ kg\ s^{-2}\ K^{-1}$ é a constante de Boltzmann.

Uma vez que estas condições sejam atendidas, pode-se afirmar que a carga se movimenta de dois modos através de um condutor comum interrompido por uma junção-túnel: contínua e discretamente. Ela flui continuamente pelo condutor e se acumula na superfície do eletrodo em contato com a camada isolante até que as condições de energia sejam favoráveis para que haja tunelamento através da barreira isolante.

2.2.4 Bloqueio de Coulomb

A forma mais fácil de se observar o efeito de carregamento (seção 2.2.3) é através da simulação da caixa mono-elétron. Este circuito é conhecido como o circuito mono-elétron mais simples, pois possui apenas uma junção-túnel. A Figura 2.4 ilustra a caixa mono-elétron.

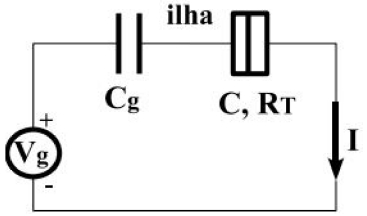

Figura 2.4: Caixa mono-elétron [5].

Neste circuito, a ilha é a região condutora entre a capacitância C_g e a junção-túnel. Definindo C_Σ como a soma entre C_g e a capacitância de carregamento da junção, aqui chamada de C, tem-se:

$$C_\Sigma = C_g + C \tag{2.8}$$

Assim, considerando que há um elétron na ilha, sua energia eletrostática aumenta de e^2/2 ·C_Σ. Nessa situação, ilustrada na Figura 2.5(a), outro elétron só poderá tunelar para dentro da ilha, se tiver energia suficiente para ocupar o primeiro nível de energia vazio. Ou seja, sua energia deve ser maior que E_C. A Figura 2.5(b) ilustra a situação em que uma tensão de polarização externa, V_g, é aplicada ao circuito. Quando isso ocorre, o circuito começa a apresentar quedas de tensão. Se essa tensão V_g alcançar um dado valor de limiar, ela poderá fornecer energia suficiente para que um elétron ocupe um dos níveis de energia vazios da ilha, permitindo que haja uma corrente através da ilha. A interrupção do fluxo de elétrons é chamada de Bloqueio de Coulomb. A tensão de limiar é chamada de tensão de Bloqueio de Coulomb e é dada pela Equação 2.10.

$$V_C = \frac{e}{C_\Sigma} \tag{2.9}$$

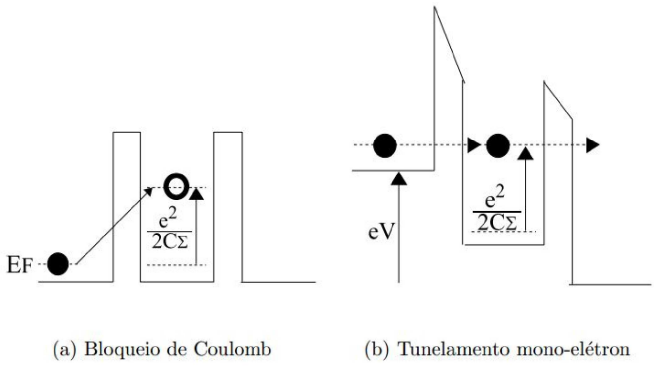

Figura 2.5: Diagramas de Energia [5].

A curva característica do Bloqueio de Coulomb é mostrada na Figura 2.6.

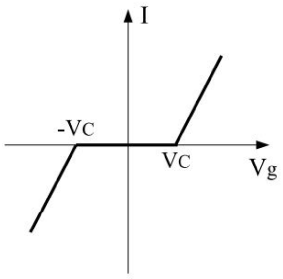

Figura 2.6: Característica do Bloqueio de Coulomb [5].

2.2.5 Transistor de Tunelamento Mono-Elétron (SET)

O SET é um dos mais promissores candidatos a substituto do transistor MOS como principal dispositivo da indústria eletrônica [36]. Ele é constituído, basicamente, por duas junções-túnel em série, formando uma ilha. Para controlar a tensão na ilha e, portanto, permitir o fluxo dos elétrons (seção 2.2.4), uma tensão externa é aplicada na ilha através de uma capacitância. Essa tensão, V_g, é chamada de tensão de porta (*gate*) do transistor. A capacitância C_g será a capacitância de porta ou *gate*. A Figura 2.7 ilustra o SET.

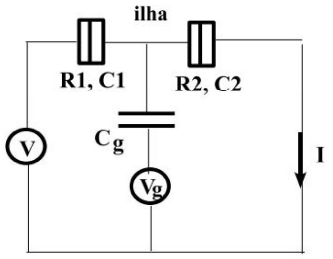

Figura 2.7: Transistor mono-elétron [5].

A ilha pode ser carregada através de tunelamento pela junção 1 e descarregada, também

por tunelamento, pela junção 2. Assim, o fluxo de cargas é controlado e dá origem a uma corrente elétrica.

A curva característica *I versus* V_g do SET é mostrada na Figura 2.8. Observa-se um fenômeno conhecido como Oscilações de Coulomb. Este fenômeno é consequência do Bloqueio de Coulomb (seção 2.2.4) e da tensão V_g.

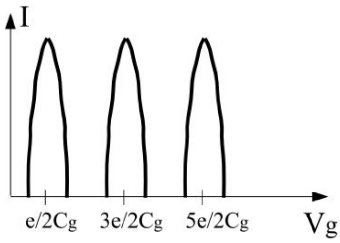

Figura 2.8: Característica I *versus* V_g do SET [5].

Se a tensão de entrada *V* for mantida constante, com *V* << *e*/C_Σ, e a tensão de porta for aumentada até um valor $V_g = e/2 \cdot C_g$, esse valor de tensão permitirá que um elétron tunele para a ilha. Conforme mostra a Figura 2.8, apenas em múltiplos dessa tensão V_g (V_g, 3 ·V_g, 5 ·V_g,...,$(2 \cdot n + 1) \cdot V_g$) poderá haver corrente fluindo no circuito. A capacitância C_Σ, no caso do SET, é dada por:

$$C_\Sigma = C_1 + C_2 + C_g \tag{2.10}$$

Cabe ressaltar que as oscilações mostradas na Figura 2.8 são muito similares aos pulsos de corrente apresentados por um neurônio pulsante (seção 2.3.3). Dessa forma, pode-se dizer que o SET possui, inerentemente, a característica de funcionamento esperada de um neurônio pulsante [37].

2.3 REDES NEURAIS ARTIFICIAIS (RNA)

Há muito tempo os pesquisadores têm tentado reproduzir a capacidade de processamento do cérebro humano em circuitos eletrônicos [11]. O altíssimo nível de paralelismo das redes neurais biológicas permite que o cérebro processe uma grande quantidade de informação em pouquíssimo tempo [14]. Essa característica é extremamente interessante para circuitos eletrônicos. Além da alta capacidade de processamento, esse nível de paralelismo em agrupamentos de neurônios biológicos provê boa tolerância a falhas [38]. Desse modo, quando, em meados do século XIX, Dubois Reymond observou manifestações elétricas no comportamento de neurônios biológicos, as ciências exatas começaram a ver tais unidades como uma potencial aplicação.

Ao agrupamento de unidades simples, os neurônios, dá-se o nome de Redes Neurais. As Redes Neurais Artificiais, RNA, são os circuitos criados pelas ciências exatas, em especial a Engenharia Elétrica, que reproduzem algumas características das redes biológicas [11]. Dentre essas, pode-se citar a auto-organização e a tolerância a falhas.

Impulsionadas pela perspectiva da criação de um circuito capaz de simular, mesmo que parcialmente, o poder de processamento do sistema nervoso biológico, as pesquisas na área de RNA cresceram [10]. Atualmente há aplicações de RNA nas mais variadas áreas, desde o processamento de imagens em Engenharia Biomédica [13] até o reconhecimento de um surto em um sistema elétrico de potência [39].

Boa parte dos trabalhos em RNA envolve redes estáticas [11]. Estas redes aprendem a mapear a informação em sua configuração interna, mudando-a conforme for necessário [38]. Neste trabalho, serão utilizadas redes neurais dinâmicas, as quais processam a informação em aproximações dependentes do tempo.

A nanoeletrônica também voltou-se para as características atrativas das RNA. Guimarães [5] propôs a Rede de Hamming utilizando o SET como o elemento essencial dos neurônios, e Akazawa *et. al* [8] propuseram a Rede de Hopfield nanoeletrônica, fazendo uso de uma junção-túnel como elemento processador.

Apesar das várias aplicações promissoras e dos vários resultados positivos, as RNA convencionais ainda encontram-se longe de reproduzir toda a capacidade de processamento de uma rede neural biológica. Dessa forma, pesquisadores começaram a procurar elementos capazes de simular com maior proximidade a característica de operação de um neurônio biológico: os pulsos [14] [10]. Assim, modelos de neurônios capazes de codificar a informação em pulsos começaram a ser propostos [40] [41]. Estes neurônios viriam a constituir os elementos processadores de redes neurais pulsadas. Tais redes apresentam um nível de paralelismo muito maior do que aquele apresentado por redes convencionais, uma vez que o projetista busca uma capacidade de processamento cada vez maior [14]. Dessa forma, elas possuem maior capacidade de processamento e maior tolerância a falhas. Estas redes serão melhor detalhadas na seção 2.3.2.

Embora a maior quantidade de elementos processadores e o maior nível de paralelismo apresentado pelas redes neurais pulsadas constituam vantagens buscadas pelos pesquisadores, eles também acabaram por apresentar um problema para a construção física de tais circuitos: o grande número de interconexões [10]. Para solucionar este problema, utilizou-se o paradoxo das *Networks-on-Chip*, NoCs. Este conceito será detalhado na seção 2.4. A próxima seção apresenta uma rápida explanação sobre o funcionamento básico de neurônios artificiais.

2.3.1 Neurônio Artificial

O primeiro artigo sobre redes neurais artificiais foi escrito por Warren McCulloch e Walter Pitts em 1943. Neste artigo, intitulado "*A Logical Calculus of the Ideas Immanent in Nervous Activity*", eles propuseram o primeiro modelo de neurônio artificial. Este neurônio possuía uma saída binária: caso houvesse pulsos o nível lógico seria 1, caso não houvesse, seria 0. Além da saída, o modelo era capaz de tratar várias entradas que poderiam ser excitatórias ou inibitórias [11]. Embora proposto há 71 anos, o neurônio de McCulloch, como ficou conhecido, traduz a operação de um neurônio artificial como é descrita até hoje.

A rede neural é composta pela conexão de vários neurônios, que são os elementos básicos nos quais se dá o processamento da informação [5]. A Figura 2.9 mostra o esquema de um neurônio artificial.

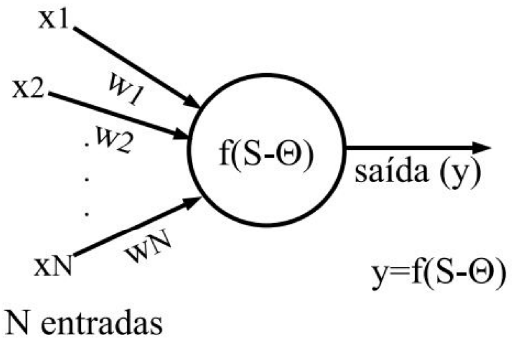

Figura 2.9: Modelo matemático de um neurônio artificial [5].

Na Figura 2.9, os x_i representam as entradas e os w_i os pesos pelos quais estas entradas são ponderadas, com i variando de 1 até N. A saída é representada por y.

Como unidade de processamento, o neurônio fará a soma de suas entradas ponderadas pelos seus pesos, e aplicará uma função de ativação ao resultado. A função de ativação pode ser descrita como o limiar que deve ser rompido pelos sinais elétricos na entrada do neurônio para que este seja ativado [12].

Seguindo o modelo de McCulloch, as entradas podem ser excitatórias (tendem a aumentar o valor total do somatório das entradas) ou inibitórias (tendem a diminuir o valor total do somatório das entradas). A descrição matemática do comportamento do neurônio está expressa na Equação 2.11.

$$S = \sum_{i=1}^{N} x_i \cdot w_i \tag{2.11}$$

Na saída do neurônio tem-se a aplicação da função de ativação ao resultado da soma dada pela Equação 2.11:

$$y = f(S - \theta) \tag{2.12}$$

onde θ é o valor de limiar para a função de ativação f do neurônio [5].

2.3.2 Redes Neurais Pulsadas

Pode-se considerar que a grande capacidade de processamento do sistema nervoso biológico e sua alta tolerância a falhas se devem, principalmente, a duas características: o alto número de unidades processadoras e o padrão complexo com o qual elas se interconectam [14]. O cérebro humano, por exemplo, possui em torno de 10^{10} neurônios conectados através de 10^{15} ramos paralelos [42]. A fim de reproduzir estas características, pesquisadores buscam a construção de arranjos de redes neurais altamente interconectadas e reconfiguráveis.

Redes Neurais Pulsadas, ou SNNs, do inglês *Spiking Neural Networks*, apresentam-se como uma alternativa interessante para a construção de tais arranjos. Conforme citado na seção 2.3, neurônios pulsantes, as unidades processadoras das SNNs, são capazes de simular de forma mais próxima o comportamento de um neurônio biológico. Desse modo, as SNNs passaram a ser vistas como computacionalmente mais poderosas do que redes neurais convencionais [14]. Além disso, SNNs exibem a propriedade de se adaptarem rapidamente a um problema e são capazes de prover tolerância a falhas [43]. Nestas redes, a informação é codificada através do tempo dos pulsos, da topologia da rede e dos pesos sinápticos [10].

O número de unidades processadoras que precisam ser colocadas em uma SNN para que ela apresente um nível de paralelismo que permita tolerância a falhas e processamento elevado tornou-se muito alto [10]. Já existem SNNs com 180.000 neurônios [24]. Este número fez com que a simulação e a construção física de redes pulsadas se tornassem um problema. Ferramentas de simulação comuns executadas em computadores comuns não conseguem simular, de forma eficiente, redes de tamanha densidade. Assim, para efetuar uma simples simulação comportamental, sistemas computacionais de alto desempenho são necessários. De fato, os pesquisadores logo notaram que as redes implementadas em *software* tornavam-se cada vez mais lentas à medida que o número de neurônios crescia [10]. Como o aumento do número de neurônios em uma SNN é uma tendência necessária, a abordagem via *software* começou a ser substituída por abordagens via *hardware* [14].

A implementação via *hardware*, contudo, também apresenta seus problemas. Desta vez, o número de interconexões e o padrão complexo que tais interconexões têm que seguir para que a rede seja totalmente interconectada dificultam a realização da SNN [19]. A utilização de barramentos para superar os problemas de interconexão foi testada sem êxito [10]. Desta forma, foram elencadas algumas características que o sistema de interconexões deve possuir para uma efetiva implementação das SNNs [10]:

1. O sistema deve ser capaz de suportar a densidade de ligações entre os neurônios;

2. O consumo de potência deve ser baixo;

3. O sistema deve ser reconfigurável;

4. O padrão apresentado deve ser intrinsecamente paralelo;

5. O sistema deve ser capaz de suportar o aumento do número de neurônios na rede, não exibindo problemas de escala.

Na Tabela 2.1 são apresentadas as aproximações disponíveis para implementação de SNNs e seus *trade-offs*. Nesta tabela, A_{oc} representa a área ocupada, P_{diss} é a potência dissipada e V_{ex} é a velocidade de execução.

Tabela 2.1: *Trade-off entre as diferentes formas de implementação de SNNs [10]*

Implementação	A_{oc}	P_{diss}	V_{ex}	Reconfigurabilidade
Software	Alta	Alta	Baixa	Alta
Firmware	Média	Média	Média	Média
Hardware	Baixa	Baixa	Alta	Média

Pela observação da Tabela 2.1 fica claro o melhor desempenho das aproximações via *hardware*. Sendo assim, uma vez que a aplicação de barramentos não foi capaz de solucionar os problemas de interconexão, os pesquisadores voltaram sua atenção para os FPGAs, *Field Programmable Gate Arrays* [44]. Esta tecnologia apresenta uma arquitetura intrinsecamente paralela e sua topologia de interconexão, conhecida como 2D-*mesh*, é vantajosa para sistemas densos. O padrão 2D-*mesh* será detalhado na seção 2.4, dedicada às NoCs.

Mais uma vez a solução encontrada mostrou-se ineficiente. O principal problema dos FPGAs é o fato de que eles não conseguem prover os altos *fan-in/out* necessários para sistemas densos como as SNNs, nas quais os estágios anteriores alimentam os posteriores. Assim, pode-se dizer que os FPGAs falham em satisfazer o requisito número 5, citado acima: a tecnologia não suporta o aumento do número de neurônios da rede [10][44].

Recentemente, as pesquisas têm se focado no paradigma de interconexão apresentado pelas NoCs. Esta tecnologia promete uma implementação altamente reconfigurável, intrinsecamente paralela e cujo consumo de potência é mantido em um nível baixo. A seção 2.4 é dedicada à apresentação das NoCs. Na próxima seção, 2.3.3, serão apresentados alguns modelos de neurônios pulsantes.

2.3.3 Neurônios Pulsantes

Nesta seção serão apresentados os três modelos de neurônios pulsantes nanoeletrônicos considerados na elaboração deste trabalho. Dentre estes modelos, um foi escolhido para a

construção das SNNs usadas neste trabalho. Como exemplo de neurônio pulsante microeletrônico, em tecnologia CMOS, *Complementary Metal Oxide Semiconductor*, pode-se citar o desenvolvido por Liu [40].

2.3.3.1 Neurônio Pulsante de Oya *et. al* [45]

Impulsionados pelas características do SET, baixíssimo consumo de potência e alto nível de integração, Oya *et. al* desenvolveram um circuito nanoeletrônico para funcionar como unidade processadora de uma SNN. Buscando um circuito nanoeletrônico capaz de operar a temperaturas $T > 0$ K, os autores se inspiraram em um neurônio MOS, desenvolvido por Kanazawa *et. al* [46]. Neste trabalho, Kanazawa *et al.* perceberam que redes neurais com sinapses depressivas que diferenciam entradas pulsadas em rajada de pulsos aleatórios respondem melhor ao ruído. Na verdade, segundo os autores, o ruído seria capaz de ajudar a rede a discriminar os padrões.

A Figura 2.10 apresenta o circuito proposto pelos autores.

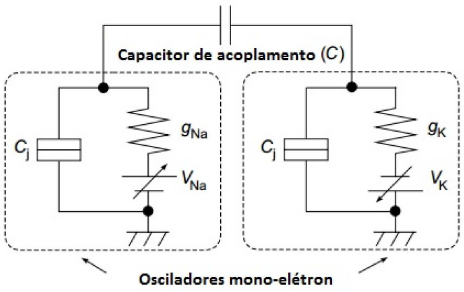

Figura 2.10: Modelo proposto por Oya *et. al*. Dois osciladores acoplados através de um capacitor.

Na figura 2.10, g representa qualquer dispositivo condutor (sua unidade é o *Siemens* - S), C_j é a capacitância da junção-túnel e Na e K representam os elementos químicos Sódio e Potássio, em uma analogia às sinapses biológicas.

Após a realização de algumas simulações com redes construídas com este modelo de neurônio, os autores constataram que a resposta do modelo é satisfatória apenas a baixas temperaturas. A temperatura máxima de funcionamento obtida foi de apenas $T = 0,5$ K.

Os dispositivos mono-elétron são bastante sensíveis ao ruído térmico, conforme pode ser visto na seção 2.2.3. Contudo, já existem propostas de neurônios pulsantes nanoeletrônicos que funcionam a temperatura ambiente, $T = 300$ K. Dessa forma, este trabalho utilizará uma destas propostas.

2.3.3.2 Neurônio Pulsante de Guimarães *et. al* [15]

Guimarães *et al.* utilizaram como inspiração o modelo de neurônio pulsante proposto por Rietman *et. al* [47]. O modelo proposto implementa a versão nanoeletrônica do chamado *Nv-Neuron*. Este neurônio é composto por um filtro passa-alta e um inversor digital, *Schmitt-Trigger*, com histerese. A Figura 2.11 mostra a proposta original, (a), e sua versão nanoeletrônica, (b).

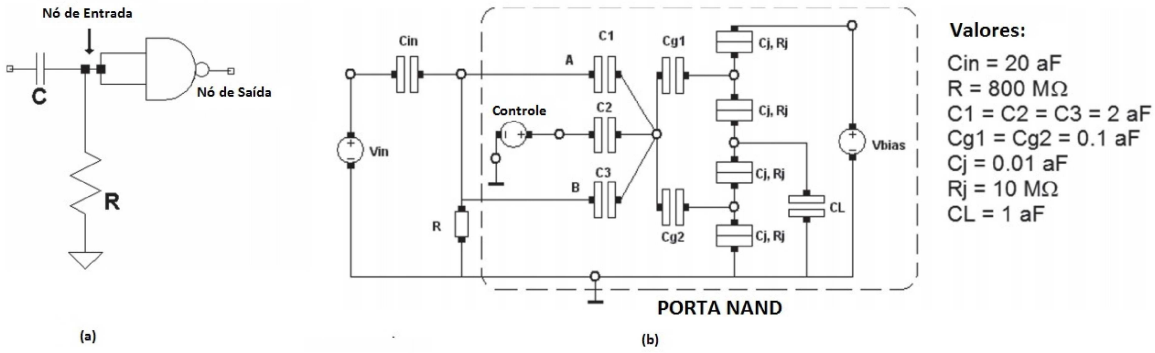

Figura 2.11: Modelo proposto por Guimarães *et al.* (a) Esquema do modelo *Nv-Neuron*. (b) Implementação nanoeletrônica do *Nv-Neuron*.

Na Figura 2.11, C_{in} é o peso aplicado à entrada, representada por V_{in}. A parte circulada, marcada como porta NAND, pode funcionar também como porta NOR. A fonte de tensão aplicada como controle define se essa parte do circuito funcionará como NAND ou NOR: se $V_{controle} = 0$, será uma porta NOR, se $V_{controle} = 1$, será uma NAND. Então, para que o circuito opere corretamente como neurônio pulsante deve-se fazer $V_{controle} = 1$.

Os autores apresentaram simulações com redes pequenas, compostas de 3 neurônios. O modelo funciona em temperatura ambiente e apresenta um padrão de pulsos bem definido. Contudo, alguns testes feitos pela autora deste trabalho deixaram a desejar quanto à utilização de entradas inibitórias. Como os pesos são representados por capacitâncias, pesos negativos poderiam ser implementados através da redução do módulo da capacitância. Porém, o modelo não respondeu bem a tal abordagem, fazendo com que não fosse escolhido para a realização deste trabalho.

2.3.3.3 Neurônio Pulsante de Wen-peng *et al.* [37]

Conforme mostrado na Figura 2.8, a característica I *versus* V do SET é pulsada. Dessa forma, Wen-peng *et al.* propuseram utilizar o próprio SET como neurônio pulsante nanoeletrônico. Inspirados no modelo proposto por Izhikevich [41], os autores buscaram um neurônio capaz de apresentar vários padrões de pulsos. Cabe ressaltar que Izhikevich [41]

fez várias comparações e concluiu que o modelo apresentado por ele é o melhor candidato para a implementação de SNNs. Nessas comparações, foram considerados, principalmente, aspectos relativos à eficiência computacional.

Wen-peng *et al.* utilizaram o algoritmo *Master Equation* no *software* de simulação PSPICE, *Personal Simulation Program with Integrated Circuit Emphasis*, para modelar o comportamento do SET. O símbolo utilizado para o SET e o circuito do neurônio proposto podem ser vistos na Figura 2.12 (a) e (b), respectivamente.

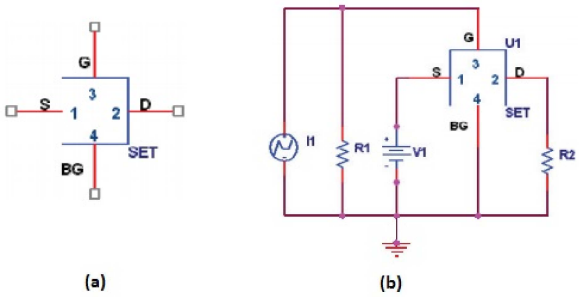

Figura 2.12: Modelo proposto por Wen-peng *et al.* (a) O símbolo do SET utilizado pelos autores, em SPICE. (b) O circuito do neurônio.

Na Figura 2.12, S, G, D e BG representam os terminais do SET: fonte - *Source*, porta - *Gate*, dreno - *Drain* e a segunda porta - *Back Gate*. O modelo apresenta uma entrada, I_1, e uma saída V_{R2}. O peso aplicado à entrada é representado por V_1, para $V_1 > 0$ a entrada será excitatória e, para $V_1 < 0$, será inibitória. O resistor R_1 deve possuir valor alto para que a tensão no *gate* varie linearmente. A variação linear desta tensão garante as Oscilações de Coulomb.

Os valores recomendados pelos autores estão representados na Tabela 2.2.

Tabela 2.2: Parâmetros do Neurônio de Wen-peng *et al.*

Parâmetro	Valor
I_1	$\sim 20pA$
R_1	$10^{15}\ \Omega$
V_1	6 - 10 mV
R_2	$10^{8}\ \Omega$
C_g	6 aF

O parâmetro C_g representa o valor da capacitância do *gate* utilizado na modelagem do SET.

Este modelo, após algumas alterações nos valores dos parâmetros, funcionou perfeita-

mente à temperatura ambiente. Além disso, vale dizer que este neurônio é capaz de codificar a informação de várias maneiras diferentes, tais como por fase, frequência e tempo de pulso. Devido a essas características, à simplicidade do modelo e à facilidade em determinar os pesos que ponderam as entradas, este modelo foi o escolhido para o desenvolvimento do trabalho.

2.4 NETWORKS-ON-CHIP

A construção física de sistemas densos em um *chip* depende da solução dos problemas criados pela interconexão destes sistemas. A grande quantidade de elementos em tais sistemas faz com que as interconexões sejam muito numerosas e, em alguns casos, muito longas. Esta seção faz uma breve introdução e apresenta as principais ideias de uma das melhores soluções já propostas: as NoCs.

Com a tendência de redução do tamanho do *gate* dos transistores, cada vez mais transistores caberão em um *chip*. Desta forma, a integração de sistemas em um *chip*, SoC - *System on Chip*, pode apresentar vários problemas devido à sua densidade. Dentre estes, pode-se citar sincronização, consumo de potência e escala. Todos estes problemas têm uma origem comum: os atrasos devidos às interconexões [48].

Várias tentativas de solucionar os problemas criados pelas interconexões foram feitas pelos pesquisadores, que fizeram uso de paradigmas já conhecidos, como a trilha dedicada e o barramento compartilhado. Em uma trilha dedicada os barramentos são comuns a dois componentes. Neste sistema, o número de interconexões cresce exponencialmente com o número de componentes, o que gera um problema de escala e impossibilita o uso desta aproximação. Em um barramento compartilhado, um conjunto de trilhas é utilizado por vários componentes. Esta aproximação sofre menos com o aumento do número de componentes do que as trilhas dedicadas. Contudo, ela apresenta limitações quanto à largura de banda e também não consegue acompanhar o crescimento da densidade dos sistemas em *chip* [48]. A falha destas duas aproximações deu início ao paradigma das NoCs.

A arquitetura das NoCs foi inspirada pelas redes de computadores pessoais, nas quais os computadores estão interconectados através de protocolos dedicados que entregam os dados de um ponto para outro. A informação é transportada, usualmente, através de várias camadas de comunicação, que promovem uma integração entre os componentes da rede, mas também desacoplam a computação da comunicação. Apesar de servir como inspiração inicial, os métodos e algoritmos das redes de computadores não podem ser aplicados às NoCs. Isso ocorre, pois as NoCs possuem requisitos específicos quanto à utilização da área e à potência dissipada, que não são as preocupações principais em redes de computadores tradicionais [10].

Geralmente, a arquitetura de uma NoC é composta por um conjunto de núcleos ou ele-

mentos processadores, roteadores e conexões. Estes elementos são arranjados em uma topologia específica, que varia conforme a aplicação desejada.

A NoC é uma arquitetura regular, baseada em blocos que se repetem, que provê a estrutura de comunicação necessária [48]. Cada NoC possui duas partes principais: um elemento de processamento e um elemento roteador [29]. Este último é a chave da solução apresentada pelas NoCs: como um dos objetivos é reduzir o número de interconexões, o roteador se faz necessário para administrar o fluxo de dados pelas vias. A NoC é, literal e basicamente, uma rede de interconexões em cujos nós se encontram os componentes do sistema. Dentre algumas topologias propostas, pode-se citar a Torus, a Mesh e a SPIN. Estas topologias podem ser vistas na Figura 2.13.

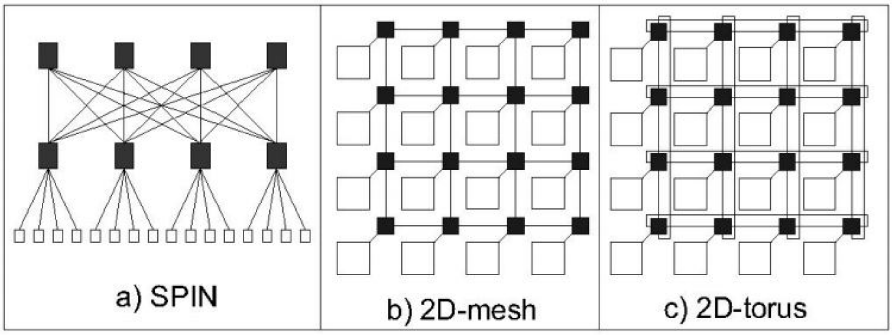

Figura 2.13: Algumas arquiteturas de NoCs: a) SPIN, b) 2D-mesh e c) 2D-torus. Os quadrados brancos representam os elementos processadores e os pretos, os roteadores [48].

Nota-se, na Figura 2.13, que os elementos processadores comunicam-se apenas com o elemento roteador presente no mesmo nó. Este elemento roteador é o responsável por toda a comunicação que se dá na rede.

A arquitetura SPIN (Figura 2.13 (a)), foi proposta por Guerrier *et al.* [49]. Ela consiste de uma arquitetura do tipo árvore, em que cada nó possui quatro ramificações, ou filhos. A arquitetura *Mesh* (Figura 2.13 (b)) foi proposta Kumar *et al.* [50] e consiste de um arranjo de M x N roteadores. Cada roteador se comunica com quatro vizinhos, com exceção dos roteadores nas extremidades, que se comunicam com apenas dois vizinhos. A arquitetura Torus (Figura 2.13 (c)) é semelhante à *Mesh*, a diferença está nos roteadores das extremidades que, neste caso, se comunicam também com os roteadores da extremidade oposta. Esta conexão extra pode gerar atrasos excessivos [48]. A Torus foi proposta por Dally *et al.* [51]. Devido à simplicidade da proposta, este trabalho utilizará a arquitetura 2D-*mesh*.

2.5 *LOOK-UP TABLES* - LUTS

Uma LUT pode também ser chamada de mapa de dados, ou *data map*. LUTs são amplamente aceitas na representação da relação entre a(s) entrada(s) e a(s) saída(s) de um sistema, uma vez que representam esta relação de forma transparente e flexível [52]. Também pode-se dizer que uma LUT assemelha-se com uma tabela verdade, já que funciona da mesma forma: para cada combinação de entradas a LUT fornece o valor da saída [18]. Uma de suas aplicações mais frequentes é a identificação de sistemas não-lineares, neste caso, o usuário deve realizar um conjunto de medições e armazenar os dados obtidos na LUT [52].

A LUT é preenchida pelo usuário, e pode conter dados absolutos (por exemplo, para um dado valor de entrada faça a saída igual a 1) ou expressões lógicas e matemáticas que calculem a saída, dada uma entrada (para uma entrada maior que um dado valor, faça a saída igual à metade da entrada) [18]. A Tabela 2.3 ilustra estes exemplos.

Tabela 2.3: Exemplos de LUTs.

	Valor da Entrada x	Valor da Saída y
LUT com valores absolutos	$x = 2$	$y = 1$
LUT com expressões	$x > 2$	$y = \frac{x}{2}$

Uma LUT pode ter os mais variados tamanhos, dependendo do número de entradas e saídas do sistema que está sendo descrito. Dessa forma, a relação entre LUTs e roteamento fica bastante clara. Conforme será explicitado na próxima seção, um roteador precisa de uma tabela que lhe informe as direções e caminhos que devem ser seguidos pelos dados de entrada. Quando o roteamento é totalmente mapeado por uma LUT, ele recebe o nome de roteamento via *look-up table*, seção 2.6.7.

2.6 ROTEAMENTO DE INFORMAÇÃO [54]

Esta seção dedica-se à apresentação de conceitos básicos relacionados ao roteamento da informação, bem como à apresentação de alguns algoritmos de roteamento.

A função principal de um roteador é rotear pacotes da máquina de origem para a máquina de destino. Para que isto seja feito da maneira mais eficiente possível, o roteador funcionará com base em um algoritmo de roteamento que escolha o melhor caminho para o pacote seguir desde sua fonte até seu destino final. A rota determinada pelo algoritmo pode ser alterada toda vez que um novo pacote chegar ou pode ser fixa (roteamento por sessão). Esta característica depende da topologia da rede: se houver a possibilidade de o melhor caminho ter sido alterado desde o envio do último pacote de dados, a melhor opção é uma rota dinâmica.

Pode-se dizer que um roteador possui dois processos básicos em seu interior. O primeiro processo, chamado de encaminhamento, ocorre quando o pacote chega ao roteador e este procura a linha de saída que será utilizada pelo pacote. Tal linha, em geral, está listada na tabela do roteador. A atualização das tabelas e o preenchimento delas é de responsabilidade do algoritmo de roteamento, esta etapa consiste no segundo processo: o roteamento propriamente dito.

Há algumas propriedades desejáveis em um algoritmo de roteamento, dentre elas pode-se citar: correção, simplicidade, robustez, estabilidade, equidade e otimização. A robustez diz respeito, principalmente, à capacidade do algoritmo para suportar mudanças na topologia da rede sem que haja interrupção no funcionamento da rede. Equidade e otimização, às vezes, podem não coexistir. Por exemplo, para haver otimização pode ser necessária a interrupção do tráfego de dados por uma linha, devido a um congestionamento, o que prejudica a equidade. Para evitar maiores confrontos, deve-se primeiro listar os parâmetros que se deseja otimizar. O tempo médio de atraso de pacote é um dos parâmetros cuja otimização é sempre interessante, já a equidade tende a reduzir o tamanho da banda consumida, o que também é sempre buscado, pois aumenta o rendimento da rede. A seguir serão apresentados alguns algoritmos de roteamento e suas vantagens. Mas, primeiramente, será apresentado o princípio de otimização.

2.6.1 Princípio de otimização

O princípio da otimização demonstra que é possível criar uma descrição geral das rotas ótimas da rede, mesmo que a topologia e o tráfego desta rede sejam desconhecidos. Considere a Figura 2.14, na qual cada ponto preto representa um roteador. O princípio da otimização estabelece que, se o roteador *J* estiver no caminho ótimo entre os roteadores *I* e *K*, então o o caminho ótimo de *J* até *K* deve pertencer a esta rota também. Por exemplo, chame a rota entre *I* e *J* de r_1, e o restante de r_2. Se houvesse uma rota melhor que r_2 entre *J* e *K*, ela seria concatenada com r_1 para melhorar o caminho e $r_1 r_2$ não seria mais a rota ótima.

Seguindo o princípio de otimização, o conjunto de rotas ótimas de todas as origens para um determinado destino forma uma árvore com raiz no destino. Tal árvore é chamada árvore de escoamento e pode ser vista na Figura 2.14 (b). O objetivo de todos os algoritmos de roteamento é descobrir e utilizar a árvore de escoamento em todos os roteadores.

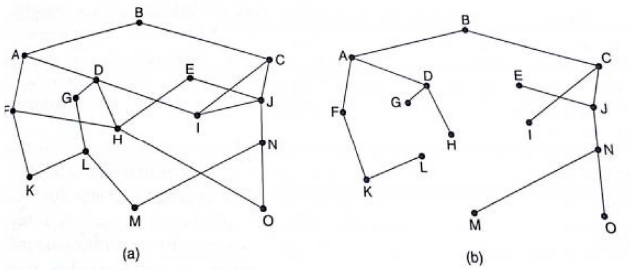

Figura 2.14: (a) Uma sub-rede. (b) Uma árvore de escoamento para o roteador B [53].

A seguir serão apresentadas as principais ideias de alguns dos algoritmos de roteamento mais conhecidos.

2.6.2 Roteamento pelo caminho mais curto

Este algoritmo é amplamente utilizado. Sua principal vantagem é sua simplicidade. A ideia aqui é desenhar um gráfico da rede, em que cada nó representará um roteador e cada arco, uma linha de comunicação. Isto feito, para estabelecer a rota dos pacotes de dados, o roteador apenas calcula a rota mais curta entre fonte e destino.

O caminho mais curto, no entanto, depende da unidade de medida de distância determinada. Em geral, programadores definem os caminhos pelo número de saltos, entre um roteador e outro, que um pacote precisa fazer até chegar ao destino. Em algumas redes, contudo, a melhor definição de caminho é a distância geográfica, dada em quilômetros. Em outros casos, o caminho mais curto pode ser determinado pelo caminho mais rápido, quando o programador o define como sendo o caminho que gera menor retardo médio de pacote.

Dentre os vários algoritmos criados para o cálculo do caminho mais curto, o algoritmo de Dijkstra é um dos mais utilizados. Desenvolvido em 1959, este algoritmo consiste em rotular cada roteador com o caminho mais curto até seus vizinhos. No início do algoritmo, nenhum caminho é conhecido, portanto todos os roteadores são rotulados com infinito. Conforme o algoritmo avança e os caminhos vão sendo encontrados, os rótulos são modificados. Um rótulo torna-se permanente quando o melhor caminho é encontrado. O critério de parada é definido pelo programador. Dessa forma, o algoritmo de roteamento pelo caminho mais curto é um algoritmo estático, que não se adapta ao fluxo da rede.

2.6.3 Roteamento por inundação ou *Flooding*

Neste algoritmo, cada pacote de entrada é enviado para todas as linhas de saída, exceto para aquela em que ele chegou. Este algoritmo não é eficiente, uma vez que gera uma grande quantidade de pacotes duplicados. Para evitar que esta quantidade chegue ao infinito, recomenda-se que o programador insira um contador de saltos que se inicie com o número

de saltos que o pacote faz da origem até o destino. Este contador é decrementado até zero. Esta técnica acaba gerando outro problema: nem sempre o programador sabe o número de saltos que o pacote dará até chegar ao destino.

Devido à sobrecarga imposta à rede, uma versão mais prática deste algoritmo foi desenvolvida: a inundação seletiva. Nesta versão, os pacotes de entrada são enviados somente para as linhas de saída que possuem alta probabilidade de se encontrarem na direção certa até o destino. Por exemplo, uma pacote com destino na direção leste não precisa ser enviado para a direção oeste.

Este algoritmo encontra aplicação em redes com finalidades militares, em que muitos roteadores podem ser destruídos pelo caminho. Desta forma, a robustez apresentada pelo algoritmo é altamente desejável. Outra aplicação são bancos de dados que necessitam ser atualizados, em sua totalidade, ao mesmo tempo.

2.6.4 Roteamento com vetor de distância

O algoritmo de roteamento com vetor de distância é um dos mais conhecidos modelos de roteamento dinâmico. Ou seja, ele considera, em seus cálculos, a carga atual da rede. Por esta razão, redes modernas preferem algoritmos dinâmicos.

Neste algoritmo, cada roteador mantém uma tabela, ou um vetor, que fornece a melhor distância conhecida até cada destino e determina qual linha deve ser usada para se alcançar este destino. A diferença aqui é que estas tabelas são constantemente atualizadas através da troca de informações entre roteadores vizinhos. A tabela possui uma entrada correspondente a cada roteador da rede. Esta entrada é composta por duas partes: a linha de saída preferencial e uma estimativa da distância até o destino (que pode ser dada pelo tempo que o pacote levará até chegar ao destino). Como no roteamento pelo caminho mais curto, esta estimativa pode ser dada em várias unidades de medida: o número de saltos, o número de pacotes enfileirados na linha e o atraso médio de pacote são alguns exemplos.

Este tipo de roteamento requer que haja um tráfego de informação entre os roteadores. Ou seja, cada roteador deve enviar aos vizinhos dados para que a estimativa de distância seja calculada. Por exemplo, se a unidade escolhida for o número de pacotes enfileirados, cada roteador envia aos vizinhos o número de pacotes enfileirados em cada uma de suas linhas de chegada. Somente este tráfego, dependendo do tamanho da rede, já pode ser considerável. Desta forma, o uso deste algoritmo se justifica em redes mais complexas, cuja intensidade de tráfego exija um algoritmo dinâmico.

2.6.5 Roteamento por estado de enlace

As linhas de comunicação entre os roteadores são chamadas de enlaces. Este algoritmo foi desenvolvido como uma alternativa ao roteamento com vetor de distância, já que este

apresentava dois problemas importantes. O primeiro diz respeito à largura de banda da linha de comunicação, que não era considerada no roteamento com vetor de distância. Com a mudança das linhas de 56 kbps para 1,544 Mbps, a largura de banda passou a desempenhar um papel considerável no roteamento. O segundo problema é o tempo de convergência do algoritmo com vetor de distância, que é muito longo.

Este algoritmo estabelece cinco tarefas para os roteadores, quais sejam:

1. Descobrir seus vizinhos e aprender seus endereços;
2. Medir o retardo até cada um de seus vizinhos;
3. Criar um pacote informativo, que contenha tudo o que o roteador aprendeu sobre seus vizinhos;
4. Enviar este pacote aos outros roteadores da rede;
5. Calcular o caminho mais curto até os outros roteadores.

O caminho mais curto é determinado pelo algoritmo de Dijkstra. O algoritmo por estado de enlace possui muitos algoritmos derivados e é, atualmente, o mais utilizado em redes sofisticadas.

2.6.6 Algoritmos *Round Robin* e *First Come First Served* - FCFS

Os algoritmos *Round Robin* e FCFS são algoritmos de agendamento de processos utilizados para organizar as linhas de comunicação nos roteadores. São os chamados algoritmos de arbitragem. Aqui será feita uma rápida explanação destes algoritmos devido ao seu uso por Harkin *et. al* [14] e Carrillo *et. al* [10], referências que serviram de inspiração para este trabalho.

O algoritmo *Round Robin* se encontra entre os mais simples meios de organização de processos. Ele é um algoritmo de escalonamento de tarefas que não atribui uma prioridade a cada tarefa. Esta característica faz com que ele seja totalmente imune ao problema conhecido como *starvation*, quando uma tarefa, por possuir prioridade baixa, nunca é executada.

No *Round Robin*, os processos são organizados de forma circular e cada um deles possui um determinado intervalo de tempo para ser executado. Em um roteador com várias linhas de chegada, ou entradas, o algortimo determinará qual linha ele atenderá primeiro. O atendimento a esta linha, que corresponde ao encaminhamento do pacote nela presente, deve ser executado em um determinado tempo, δt. Após este tempo, o algoritmo atenderá a próxima entrada. Esse processo se repete de forma circular, de modo que pode-se considerar que a prioridade mais baixa é dada à última tarefa atendida pelo algoritmo. Este algoritmo é excelente para redes com tráfego pesado, quando todas as portas dos roteadores estão ocupadas ao mesmo tempo [10].

No algoritmo FCFS, a prioridade mais alta é dada ao primeiro evento que ocorre. No caso, ao primeiro pacote que chega no roteador. Por esta característica, ele é um algoritmo adequado para redes com tráfego baixo ou médio, nas quais poucas portas estão ocupadas ao mesmo tempo. Em redes de tráfego pesado, o uso deste algoritmo para arbitrar o atendimento aos pacotes pode gerar altas taxas de perdas de pacotes. Uma característica atraente deste algoritmo é o fato de ele não gastar tempo verificando portas inativas, uma vez que ele só reage à chegada de pacotes [10].

2.6.7 Roteamento via *Look-Up Table*

O roteamento através de uma LUT é uma das formas mais simples de se implementar roteadores [14]. Tal simplicidade pode limitar o tamanho da rede cujas conexões estarão armazenadas na LUT: pode ser trabalhoso preencher a LUT quando a rede for muito grande.

A LUT confere bastante flexibilidade à lógica de roteamento. Não há restrições quanto aos algoritmos nos quais o projetista pode se basear para gerar o mapa de entradas e saídas. Sendo possível, inclusive, fazer uso de mais de um algoritmo e criar regras novas. A flexibilidade também se traduz no fato de que a LUT pode ser alterada conforme a necessidade de uso do roteador. Ou seja, o mesmo roteador poderá tratar o fluxo de várias redes, demandando, para tanto, apenas alterações rápidas na LUT.

Neste trabalho, os roteadores, tanto na versão simples quanto na versão *full*, serão implementados através de LUTs. A flexibilidade será demonstrada para o roteador *full*, através da realização do problema da XOR com 3 e com 5 entradas. Na próxima seção, será explicada a proposta do artigo que inspirou a realização deste trabalho.

2.7 *EMBRACE* [8]

Esta seção dedica-se a apresentar o *EMBRACE*, *EMulating Biologically-inspiRed ArChitectures in hardwarE*, a proposta que inspirou este trabalho.

Harkin *et al.* perceberam a similaridade do problema de interconexão de SoCs e de SNNs. Assim, eles propuseram a utilização de uma NoC para a interconexão de uma SNN. Desta forma, o elemento processador é o chamado Bloco Neural, que pode ser programado para operar como um neurônio pulsante. O roteador usado baseia-se no algoritmo *Round Robin*, ver seção 2.6.6, e foi implementado através de uma LUT. A Figura 2.15 apresenta a proposta como um arranjo bidimensional de blocos neurais cercados por blocos de entrada e saída.

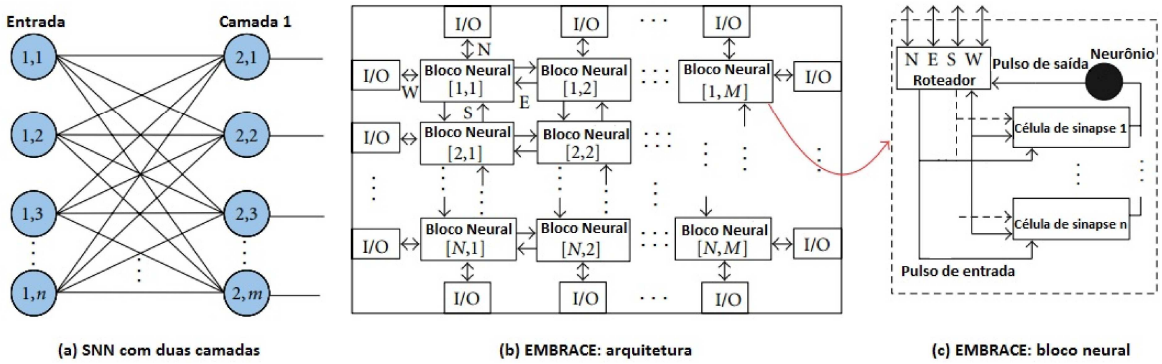

Figura 2.15: *EMBRACE*: arquitetura [14] - modificada.

Na Figura 2.15 (a), percebe-se que, quando um neurônio na camada de entrada dispara, o sinal se propaga pelas conexões dedicadas entre as duas camadas. Usando a NoC, esta conectividade é atingida através da multiplexação no tempo dos canais de conexão. Isto reduz a densidade de interconexões, uma vez que a estrutura de interconexão fornecida pela NoC é fixa, de *layout* regular e controlada pelos roteadores. A multiplexação no tempo pode levar a um aumento do intervalo de pulso entre os neurônios. Ou seja, os dados demoram mais para passar de um neurônio para outro. Apesar desta degradação, os autores apontam que, em comparação com o cérebro humano no qual o intervalo de pulso entre neurônios chega a 10 ms, a arquitetura ainda apresenta ganho.

Na Figura 2.15 (c) cada célula de sinapse representa uma entrada. Estas entradas são enviadas ao neurônio e nele somadas, gerando ou não um pulso de saída, conforme seção 2.3.1. As entradas e saídas das células de sinapse são controladas pelos roteadores. Um trem de pulsos é recebido como um pacote de dados vindo dos roteadores vizinhos. Cada pacote inclui um endereço fonte, que identifica o neurônio no qual o pulso se originou, e um endereço destino, que identifica o neurônio para o qual o pacote deve ser enviado. Cada pacote contém 22 bits. Destes, os dois primeiros destinam-se a identificar o tipo de informação (que pode ser dados de pulsos, configuração ou erros), 10 serão para o endereço da fonte e outros 10 para o endereço de destino. Totalizando, assim, 1024 endereços diferentes. Pode-se, então, concluir que este é o número máximo de neurônios que uma rede *EMBRACE* pode interconectar. Informações específicas, tais como quais são os modos de configuração do neurônio e quais bits os representam podem ser vistas em [14]. Nesta seção ressaltam-se apenas as informações que geraram a inspiração deste trabalho, o qual propõe a utilização de um neurônio pulsante nanoeletrônico como elemento processador e um roteador implementado via LUT. Este bloco, presente em cada nó de uma rede 2D-mesh, interconecta uma SNN nanoeletrônica.

2.8 FERRAMENTAS DE SIMULAÇÃO

Esta seção apresenta características básicas da ferramenta de simulação utilizada neste trabalho. Há, também, uma discussão sobre os parâmetros de simulação do LTSpice IV.

2.8.1 LTSpice IV

O LTSpice é uma versão de *software* livre, que não necessita de licença de uso, do *software* ORCAD. A interface de usuário altamente intuitiva do LTSpice permitiu que ele se tornasse bastante utilizado em projetos e simulações de circuitos elétricos e eletrônicos. O crescente uso desta plataforma entre estudantes de engenharia elétrica gerou bastante documentação *online*, como fóruns e grupos de *e-mail* destinados à discussão de erros e modelos de circuitos. Esta característica é vantajosa para os usuários, que acabam encontrando soluções para alguns problemas recorrentes da plataforma.

Os principais tipos de análise do LTSpice são as análises DC, AC e transiente. Na análise DC, as tensões DC de todos os nós do circuito são estimadas. A análise AC mostra a resposta em frequência do circuito. E, por último, a análise transiente fornece a resposta do circuito no domínio do tempo [54]. Neste trabalho será utilizada a simulação transiente, cuja tela de comando é apresentada na Figura 2.16. Os parâmetros são o tempo de simulação, definido pelo usuário, no caso 20 ns. O tempo no qual os dados começarão a ser salvos, se for deixado sem preenchimento, os dados são salvos a partir do início da simulação. O *timestep* máximo que, algumas vezes, precisa ser definido pelo usuário mas, se não preenchido, é igualado ao tempo de impressão dos dados no gráfico de simulação.

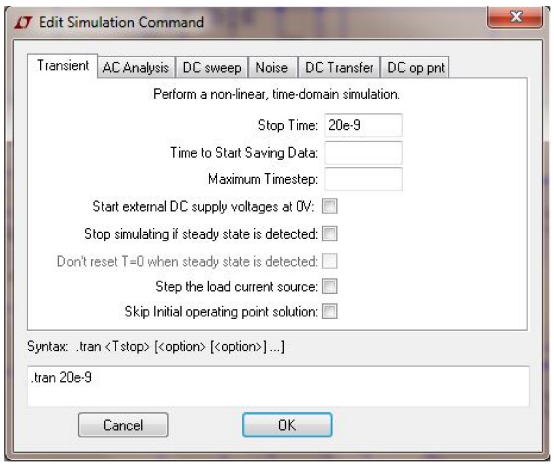

Figura 2.16: Tela de comando de simulação do LTSpice, versão IV, no *Windows* 7.

Uma funcionalidade bastante interessante do LTSpice é o fato de o *software* gerar, uma vez feita a simulação, um arquivo que contém todas as variáveis do circuito (tensões e correntes). Ou seja, basta fazer a simulação uma única vez para que os dados necessários para

a construção de todos os gráficos do circuito sejam salvos. Esta característica pode poupar ao usuário muito tempo, caso ele esqueça de salvar algum gráfico após a simulação. Além desta vantagem, a construção de circuitos grandes no LTSpice é mais simples e menos trabalhosa do que em outros simuladores nanoeletrônicos famosos, como o SIMON, *Simulation of Nanostructures*. Isto ocorre, pois o LTSpice possui a funcionalidade de copiar partes do circuito, sem precisar construí-las repetidamente. No SIMON, todas as partes devem ser construídas individualmente. Para circuitos como o utilizado neste trabalho, no qual um bloco de construção repete-se várias vezes para compor o circuito como um todo, poder efetuar a montagem do bloco apenas uma vez e copiá-lo posteriormente representa uma economia de tempo considerável. Esta característica e a quantidade de informação disponível *online* fizeram com que o LTSpice fosse escolhido como ferramenta de simulação para este trabalho.

Uma característica importante do LTSpice é o fato de ele apresentar alguns erros que, em sua maioria, não se relacionam diretamente com a topologia do circuito. Estes erros, comumente chamados de erros de simulação ou problemas de convergência, podem ser muito difíceis de eliminar. A origem destes erros decorre do fato de o LTSpice ser um *software* livre. Ou seja, permite-se que os usuários efetuem mudanças no código fonte. Geralmente, tais erros podem ser eliminados com o ajuste de alguns parâmetros de simulação. Desse modo, a próxima seção faz uma breve explanação dos parâmetros de simulação do LTSpive IV.

2.8.2 LTSpice IV - parâmetros de simulação

Conforme dito na seção anterior, o LTSpice é um *software* livre, de modo que as informações dadas neste capítulo foram obtidas no *site* LT Wiki. O conhecimento desta ferramenta é fundamental para os usuários do LTSpice. O LT Wiki consiste de um espaço no qual informações geradas através de discussões entre usuários são disponibilizadas. Assim, o *site* apresenta soluções simples para problemas frequentes.

Os parâmetros de simulação, no LTSpice IV, podem ser divididos em três grandes grupos: método, tolerância e limite. Os parâmetros relativos ao método referem-se ao cálculo que será aplicado durante a simulação. Existem 3 métodos de integração numérica disponíveis: trapezoidal, *modified trap* e *gear*. O método *default* é o *modified trap*. Os detalhes de funcionamento de cada uma destas formas de integração fogem ao escopo deste trabalho. Contudo, o usuário precisa entender o *trade-off* implícito na escolha do método de integração. De modo geral, os métodos estão elencados em ordem decrescente de precisão, sendo o trapezoidal o mais preciso e o *gear* o menos preciso. Outro fator a ser considerado é o tempo de simulação: quanto mais preciso for o método, mais demorada será a simulação. Dessa forma, cabe ao projetista escolher o método de integração sabendo qual o nível de precisão necessário. A Figura 2.17 ilustra a tela do painel de controle do LTSpice.

Há dois tipos de erros bastante frequentes no LTSpice: *timestep too small* e *singular matrix*. Ambos traduzem problemas na convergência da simulação e podem ocorrer por inúmeras razões. Eles são frequentemente referidos como mensagens de erro padrão do LTSpice. Ressalta-se que o primeiro passo do usuário deve ser, sempre, a conferência da topologia do circuito. Uma vez conferida a topologia, pode-se partir para o ajuste dos parâmetros de simulação, sempre tendo em mente as variáveis precisão dos resultados obtidos e tempo de simulação.

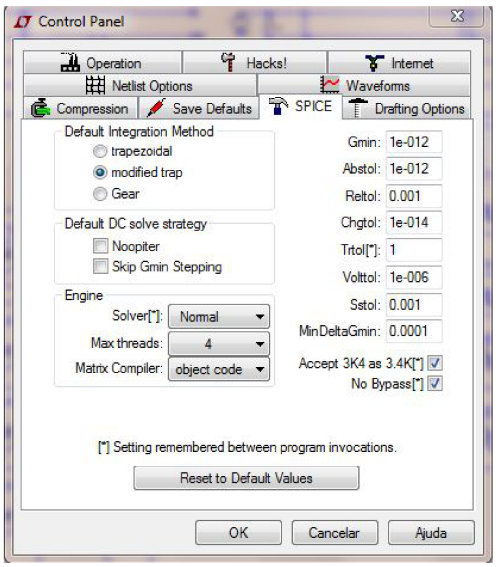

Figura 2.17: Tela do Painel de Controle do LTSpice, versão IV, no *Windows* 7.

Em alguns casos, a mudança do método de integração não é suficiente para que a simulação ocorra conforme o esperado. Sendo assim, pode ser necessário realizar um ajuste fino, efetuando o ajuste de outros parâmetros que se relacionam com o método de integração numérica. Os principais parâmetros, utilizados neste trabalho, estão demonstrados na Tabela 2.4.

Tabela 2.4: Parâmetros de Simulação relacionados com o método de integração numérica.

Parâmetro	Significado
BYPASS	Acelera a simulação, pois não atualiza o estado de componentes latentes. *default* $= 1$, habilitado. Pode causar perda de precisão para análise AC.
CSHUNT	Adiciona um valor de capacitância para o terra em cada nó do circuito. *default* $= 0$. Útil para erros do tipo *time step too small* e *singular matrix*.
DVDT	Permite ao usuário escolher o passo de simulação (tempo) - *timestep*. *default* $= 4$. Opções: 0 (algoritmo original), 1 (rápido), 2 (preciso) e 3,4 (equilibra rapidez e precisão).
GSHUNT	Adiciona um valor de condutância para o terra em cada nó do circuito. *default* $= 0$. Útil para erros do tipo *time step too small* e *singular matrix*.
LVLTIM	Seleciona o algoritmo de *timestep* para análise transiente. Valor *default* depende do método de integração. *Gear*, *LVLTIM* $= 2$ - Algoritmo de truncamento local, maior exatidão e evita propagação de erros. Trapezoidal ou *modified trap*, *LVLTIM* $= 1$ ou 3 - Algoritmo *DVDT*, evita o erro *time step too small*.

A seguir serão explicados os principais parâmetros, utilizados neste trabalho, que se encaixam no grupo de tolerância. Basicamente, estes parâmetros aumentam ou diminuem a tolerância ao erro, também mudando a precisão e o tempo da simulação.

Tabela 2.5: Parâmetros de Simulação relacionados com a tolerância a erros.

Parâmetro	Significado
VNTOL	Determina a tensão mínima para análises DC e transiente. *default* $= 50\ \mu V$. Se a precisão é muito importante, deve ser reduzido. Para circuitos com tensões maiores que 1 kV, pode ser aumentado, 5 a 50 mV.
CHGTOL	Determina a tolerância de erro de carga para *LVLTIM* $= 2$. *default* $= 1 \cdot 10^{-15}$ C.
RELTOL	Afeta o critério de convergência, determina o limite superior do erro. *default* $= 1 \cdot 10^{-3}$. Aumentar *RELTOL* aumenta o erro permitido. Auxilia na solução de erros do tipo *time step too small*.
ABSTOL	Determina a tolerância absoluta para correntes. *default* $= 1 \cdot 10^{-12}$ A. Deve ser menor que o menor valor de corrente do circuito. Útil para erros do tipo *time step too small*.

Os parâmetros do terceiro grupo, limites, representam os critérios de parada da simulação. Não foram necessários ajustes de parâmetros pertencentes a este grupo, de modo que os valores *default* funcionaram perfeitamente nas simulações realizadas neste trabalho. Cabe ressaltar que as Tabelas 2.4 e 2.5 referem-se a parâmetros cujo ajuste foi necessário para que a simulação ocorresse conforme o esperado. O LTSpice possui inúmeros parâmetros que necessitam ou não de ajuste, dependendo dos níveis de tensão e corrente e da topologia do circuito. A próxima seção apresenta uma análise da validade do modelo de SET utilizado neste trabalho.

2.8.3 Validade do modelo de SET

No artigo em que propuseram a utilização do SET como componente principal de um neurônio pulsante nanoeletrônico, Weng-peng *et al.* também desenvolveram um modelo em SPICE para o SET. Contudo, conforme pode ser visto em [37], este modelo não opera em temperatura ambiente, sendo limitado a simulações com T = 0 K. Além disso, o modelo proposto por Weng-peng *et al.* não é funcional para qualquer versão de *software* livre do ORCAD, apenas para o PSPICE, *Personal Simulation Program with Integrated Circuit*

Emphasis. Sendo assim, optou-se por utilizar o modelo de SET desenvolvido por Lientschnig *et. al* [55].

Este modelo baseia-se na teoria ortodoxa do tunelamento mono-elétron, ver seção 2.2.2, e pode ser usado em qualquer versão do SPICE. Além disso, funciona perfeitamente para T = 300 K. Sendo, portanto, um modelo mais completo do que o proposto por Weng-peng *et al*.

Uma das limitações deste modelo é o fato de ele não considerar eventos de ordem elevada, como o co-tunelamento, quando dois elétrons tunelam simultaneamente por barreiras túnel diferentes [5]. Este fato pode impossibilitar o uso do modelo para simular circuitos nos quais o co-tunelamento é um fenômeno essencial, como a rede mono-elétron de Hopfield [27]. Circuitos cujos estados são determinados pela presença ou ausência do elétron em determinado lado da junção-túnel também podem ter seu comportamento deteriorado pela falta de consideração do co-tunelamento no cálculo [30]. O modelo, na verdade, despreza a natureza estocástica do tunelamento. Esta característica limita a operação de circuitos que utilizem estes modelos no que concerne à frequência de operação [55]. Neste trabalho, os estados do circuito são determinados pela presença ou não de pulsos na saída, ver seção 2.9, e o co-tunelamento não é um fenômeno importante para o funcionamento do modelo de neurônio escolhido [37]. Dessa forma, não há restrições para o uso deste modelo nos circuitos aqui implementados.

A Figura 2.18 mostra o circuito desenvolvido por Lientschnig *et. al* e o símbolo que o representa no LTSpice IV.

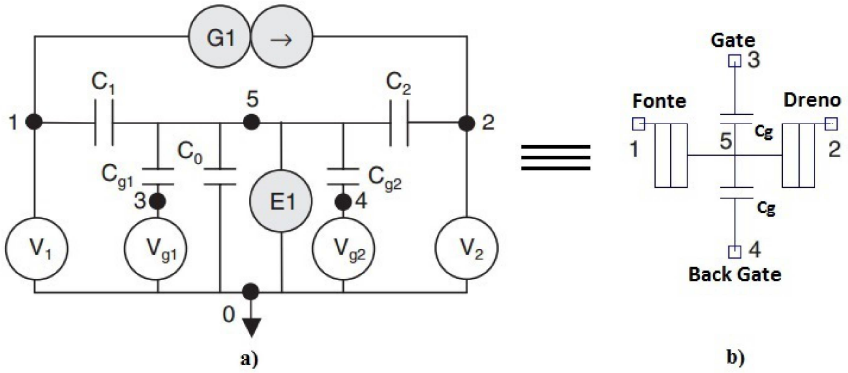

Figura 2.18: O modelo do SET no SPICE. a) Circuito. b) Símbolo. [55]-modificada

Na Figura 2.18, as fontes de tensão em branco são externas ao modelo, as em cinza são internas. A fonte E1 fixa o valor de tensão na ilha, enquanto a fonte de corrente G1 especifica a corrente entre fonte e dreno. Lientschnig *et al*. optaram por criar um modelo com 2 *gates*, a fim de torná-lo aplicável a ambas situações: quando o segundo *gate* é necessário e quando não o é. Uma aplicação clássica na qual dois *gates* são necessários é a realização do somador decimal com SET [56].

2.9 ANÁLISE CRÍTICA DO MODELO DE NEURÔNIO DE WEN-PENG *ET AL.*

Conforme será demonstrado no próximo capítulo, o desenvolvimento das portas lógicas utilizadas neste trabalho seguiu a metodologia proposta por Yellamraju *et. al* [57]. Contudo, neste artigo, os autores utilizam um neurônio pulsante com tecnologia CMOS. Dessa forma, são aqui apontadas características interessantes do comportamento do modelo de neurônio de Wen-peng *et al.*

Uma das características interessantes do modelo de Wen-peng *et al.*, seção 2.3.3.3, é a relação direta existente entre a frequência dos pulsos na saída e a amplitude da entrada, mostrada na Figura 2.19. Esta propriedade comprova que o modelo é capaz de reproduzir os chamados pulsos excitáveis, nos quais a frequência da saída pulsada aumenta com a amplitude da entrada [41].

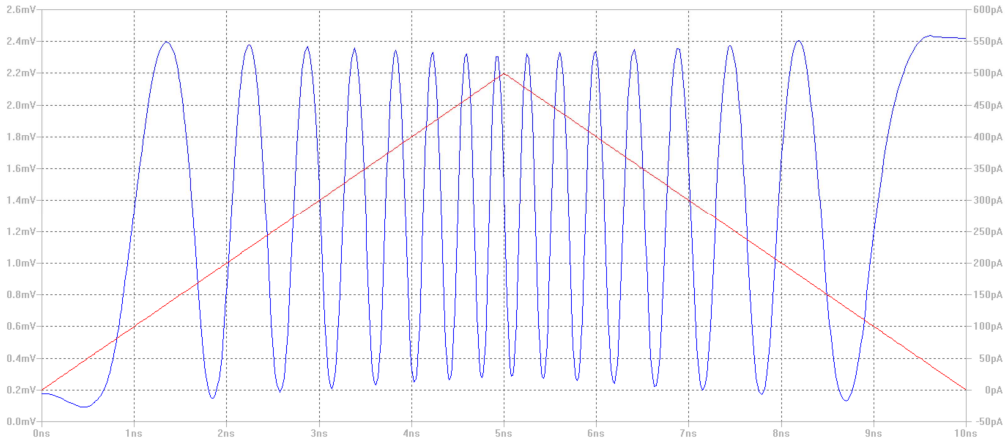

Figura 2.19: Relação entre a frequência da saída e a amplitude da entrada. Em vermelho, a entrada triangular e, em azul, os pulsos exibidos na saída.

A diferença principal, e mais relevante para a realização deste trabalho, entre os modelos de neurônio nanoeletrônico aqui utilizado e o modelo CMOS, utilizado por Yellamraju *et. al*, consiste na defasagem dos pulsos. O modelo CMOS apresenta defasagem de 180° entre pulsos gerados por entradas inibitórias e excitatórias [57], enquanto o modelo de Wen-peng *et al.* não. A existência de defasagem de 180° entre estes pulsos é uma propriedade altamente desejável para a realização de circuitos lógicos. Duas entradas, uma excitatória e uma inibitória, com o mesmo valor absoluto, quando somadas não produzem excitação. Desse modo, a simples codificação nível 0 quando não há pulsos e nível 1 quando há, pode ser utilizada. Tal codificação é muito útil em circuitos lógicos como as portas AND e XOR, que devem fornecer saída nula quando ambas as entradas estão em nível 1. A capacidade de anular os pulsos gerados por entradas com pesos de sinais diferentes é também fundamental na implementação da porta inversora, na qual a saída deve estar em nível alto quando a entrada está em nível baixo. A Figura 2.20 ilustra a defasagem entre os pulsos.

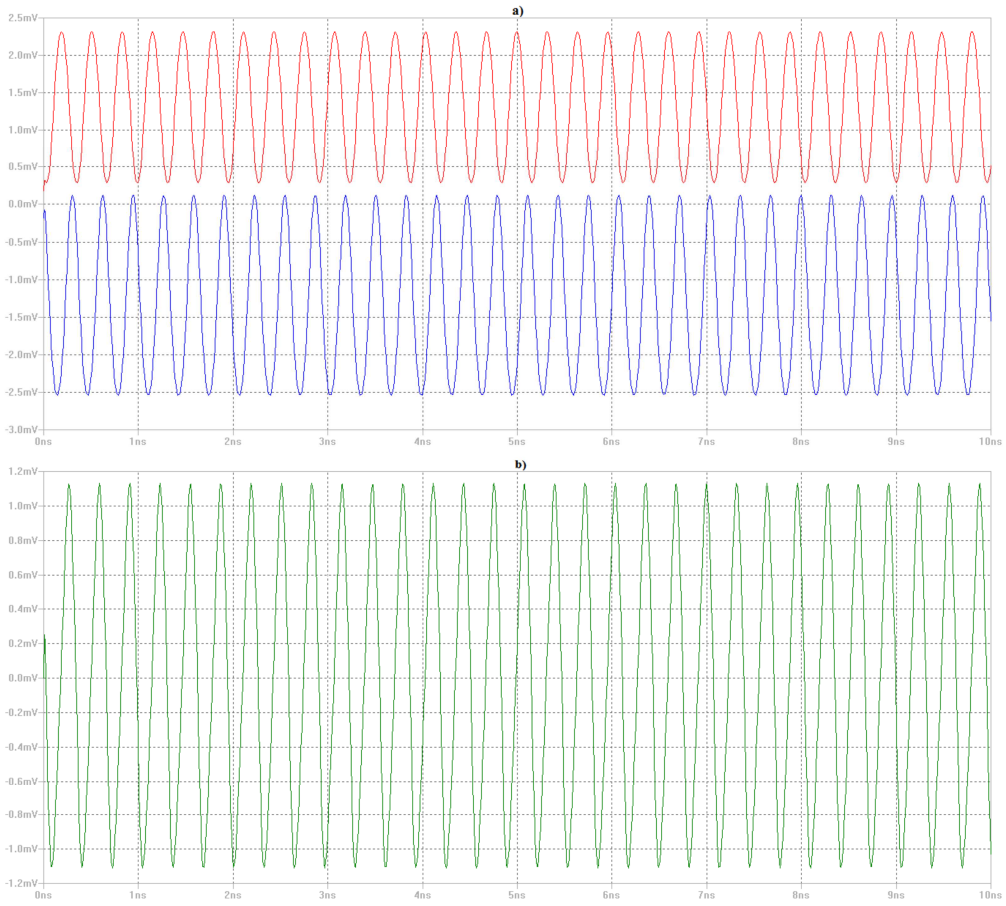

Figura 2.20: Defasagem entre os pulsos gerados por entradas inibitórias e excitatórias. a) Vermelho: saída gerada pela entrada excitatória, azul: saída gerada pela entrada inibitória. b) Soma das saídas.

Além disso, notou-se que o neurônio nanoeletrônico, quando uma sequência de pulsos é interrompida, tende a continuar a próxima sequência de pulsos exatamente do ponto no qual a sequência anterior foi interrompida. A Figura 2.21 ilustra a situação.

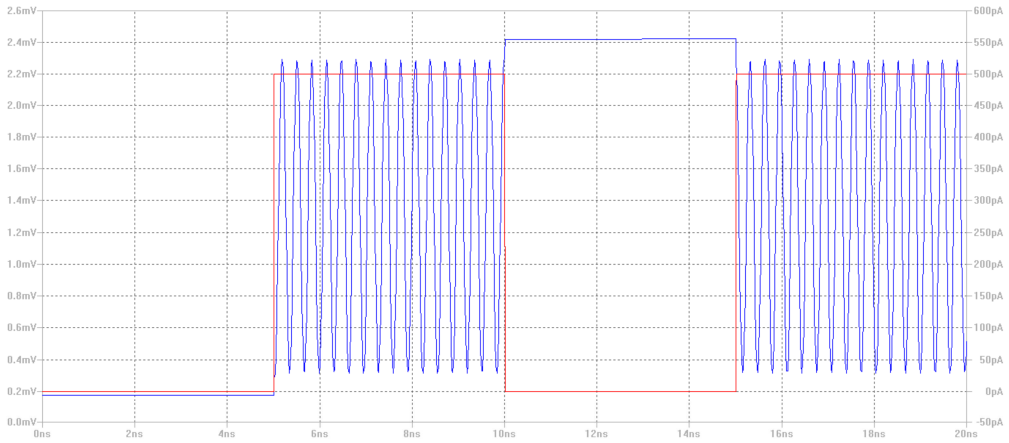

Figura 2.21: Ilustração de como ocorre a transição entre duas sequências de pulsos: a condição final da 1° sequência é mantida até o início da 2° sequência.

A situação demonstrada na Figura 2.21 faz com que a eliminação da defasagem entre os pulsos de entradas inibitórias e excitatórias se torne muito trabalhosa. Para que se efetuasse uma compensação capaz de tornar a defasagem igual a 180°, o usuário precisaria determinar, com exatidão, o ponto no qual a amplitude da entrada cai a ponto de interromper os pulsos na saída. Sendo assim, pode-se optar por modificar a codificação ou trabalhar com os pesos da rede, para que a entrada inibitória gere na saída pulsos com amplitudes que possam ser consideradas desprezíveis.

A terceira e última característica do modelo de neurônio de Wen-peng *et al.*, a ser discutida nesta seção, é o seu comportamento diante de transições. Como acontece com outros modelos de neurônios pulsantes, como o de Guimarães *et al.*, o modelo de Wen-peng *et al.* é muito sensível à transição. De forma que uma variação na entrada pode excitar o neurônio em regiões teoricamente não desejadas. A Figura 2.22 ilustra a situação.

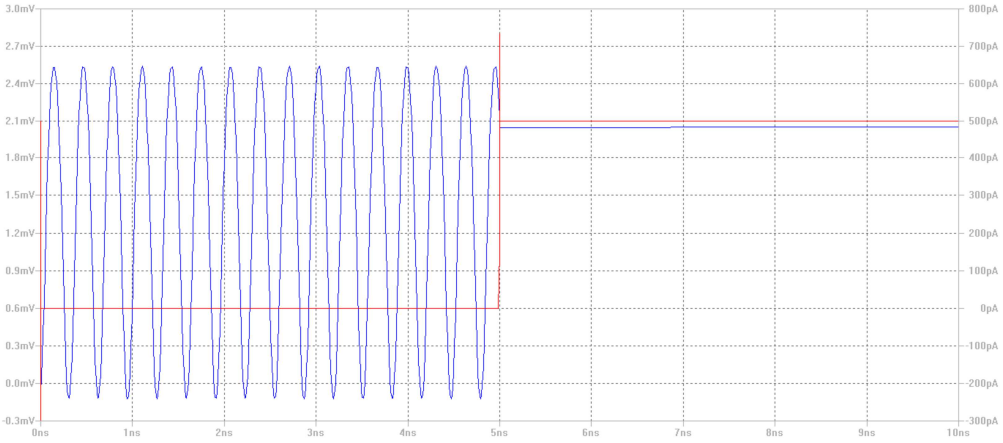

Figura 2.22: Excitação do neurônio devido à ocorrência de transição. Em vermelho, a entrada PULSE com tempos de subida e de descida iguais a 1 fs. Em azul, os pulsos fornecidos na saída.

Este comportamento, no LTSpice, é observado quando ondas de entrada do tipo PULSE são utilizadas. As ondas PULSE sempre apresentam uma transição no início da simulação. Se, como exemplifica a Figura 2.22, o usuário desejar que a entrada inicie-se em 0, o LTSpice fará uma transição, com tempo de queda definido pelo usuário, do valor de amplitude máximo até 0. A Figura 2.23 ilustra a tela do LTSpice na qual são preenchidos os parâmetros para geração de uma onda PULSE. No caso demonstrado na Figura 2.23, haverá uma transição de 500 pA para 0 A, com duração de 1 fs, no início da simulação. Caso os valores de I1[A] e I2[A] fossem trocados, essa transição seria de 0 A para 500 pA.

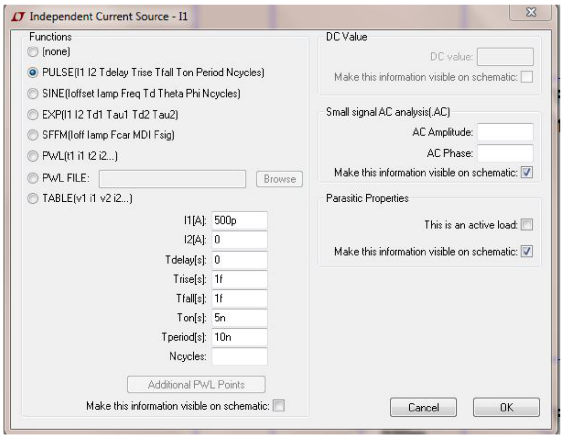

Figura 2.23: Tela do LTSpice IV, no *Windows 7*, com os parâmetros para geração de uma onda do tipo PULSE. I1[A] representa o valor máximo do sinal, I2[A] representa o valor mínimo. Tdelay[s] é o atraso aplicado às transições, geralmente nulo. Trise[s] e Tfall[s] especificam os tempos de subida e descida nas transições. Ton[s] determina o tempo pelo qual o sinal se manterá em nível alto e Tperiod[s] especifica o período do sinal.

O comportamento demonstrado na Figura 2.22 também pode ser reproduzido com a utilização de ondas quadradas, basta que o usuário force uma transição no início da simulação.

É válido ressaltar que o projetista pode utilizar esta característica do comportamento do neurônio em seu favor, como será mostrado no próximo capítulo.

3 VALIDAÇÃO DO MODELO DE NEURÔNIO DE WEN-PENG *ET AL.*

"Carry on my wayward son,
There'll be peace when you are done"
Kerry Livgren

Para que a construção do bloco proposto fosse possível, foi necessária a realização de alguns testes com o modelo de neurônio de Wen-peng *et. al.* Estes testes referem-se a dois aspectos fundamentais: o funcionamento do modelo à temperatura ambiente e o comportamento conjunto de mais de um neurônio na implementação de portas lógicas. A seguir serão descritos os procedimentos seguidos para a realização destes testes e os resultados serão apresentados.

3.1 MODELO DE WEN-PENG À TEMPERATURA AMBIENTE

Na seção 2.3.3.3, o modelo de neurônio pulsante de Wen-peng *et al.* [37], foi apresentado. Os parâmetros utilizados pelos autores no artigo original, mostrados na Tabela 2.2, são válidos para operação a $T = 0$ K. Além disso, conforme explicado na seção 2.8.3, o modelo SPICE para o SET, utilizado por Wen-peng *et al.* [37], funciona apenas para o *software* PSPICE. Sendo assim, o primeiro passo foi substituir o modelo SPICE pelo desenvolvido por Lientshnig *et al.* [55].

O segundo passo para a realização deste trabalho foi a modificação dos parâmetros da Tabela 2.2, para que o modelo de neurônio funcionasse corretamente a $T = 300$ K. Para que esse ajuste fosse feito, foram realizadas algumas simulações, alterando os valores dos parâmetros até que o funcionamento correto fosse atingido. Vale ressaltar que, conforme estudado por Pês [35] e por Nogueira [9], a redução dos módulos das capacitâncias do circuito e o aumento das fontes de tensão de entrada, em geral, resolvem o problema de funcionamento à temperatura ambiente.

Para a realização das simulações foi utilizado o símbolo do SET com 2 *gates*, ver seção 2.8.3, conforme mostra a Figura 3.1. A fonte de corrente I1 fornece na entrada uma onda quadrada, com valor máximo de 500 pA e período de 10 ns, conforme a Figura 3.2 (b).

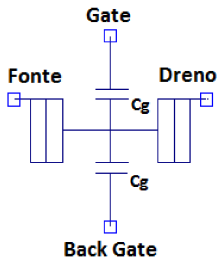

Figura 3.1: Símbolo do transistor mono-elétron de 4 terminais, SET, utilizado. C_g representa a capacitância de *gate*.

O circuito do neurônio é mostrado na Figura 3.2. Os parâmetros obtidos através desta simulação, e utilizados durante todo o trabalho, são mostrados na Tabela 3.1.

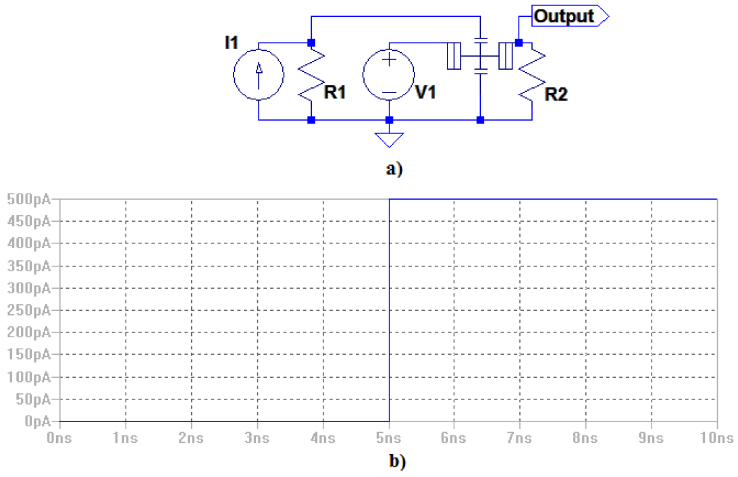

Figura 3.2: a) Circuito do neurônio de Wen-peng *et al.*, construído no LTSpice IV. b) Entrada apresentada ao circuito pela fonte I1.

Tabela 3.1: Parâmetros do Neurônio de Wen-peng *et al.* para funcionamento em $T = 300$ K.

Parâmetro	Valor
I_1	500 pA
R_1	10^{15} Ω
V_1	100 mV
R_2	10^7 Ω
C_g	1 aF

A redução do módulo da capacitância C_g e o aumento nos valores de V_1 e de I_1 foram decisivos no funcionamento do circuito à temperatura ambiente. A Figura 3.3 mostra a onda de corrente apresentada na entrada e a resposta do neurônio, pode-se notar a ativação do neurônio depois de ocorrida a transição na entrada.

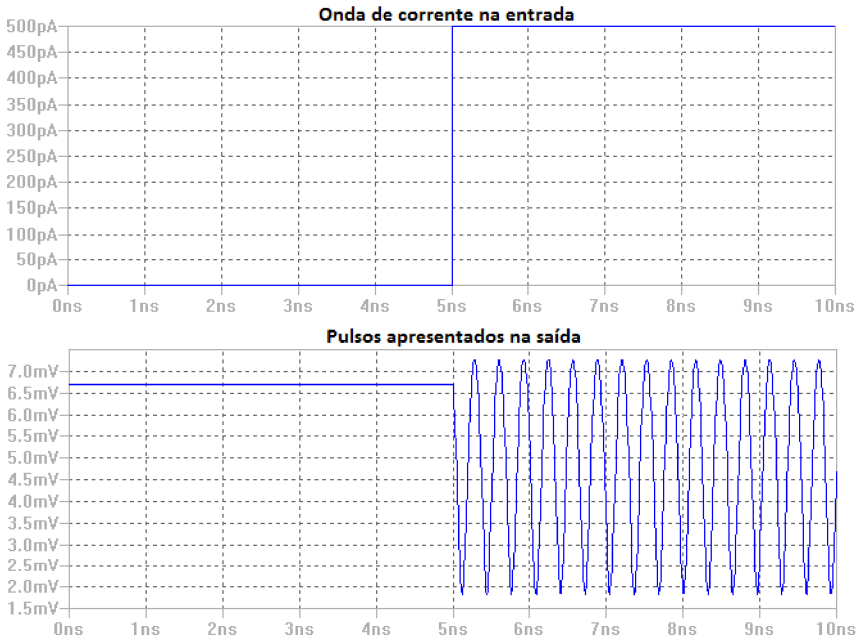

Figura 3.3: Resultado da simulação após o redimensionamento do modelo de Wen-peng.

3.2 CODIFICAÇÃO DA INFORMAÇÃO

Quanto à codificação utilizada, levando em consideração o problema de defasagem diferente de 180° entre os pulsos gerados por entradas inibitórias e excitatórias, a princípio uma codificação por amplitude foi cogitada. Desse modo, uma redução da amplitude dos pulsos na saída representaria o nível lógico 0. Após alguns testes, contudo, constatou-se que esta codificação tampouco funcionaria.

Devido ao comportamento do SET, descrito na seção 2.9, Figura 2.21, não há como garantir que haverá redução na amplitude dos pulsos de saída quando duas excitações de sinais diferentes forem somadas. Isto ocorre, pois nem sempre os vales e os picos dos pulsos coincidem, dependendo do ponto no qual a sequência de pulsos foi interrompida, a próxima sequência pode ser iniciada em um vale ou em um pico.

A Figura 3.4 demonstra a tentativa de utilização da metodologia proposta em [57] para realização da porta inversora. Esta metodologia será detalhada na próxima seção. Ela consiste, basicamente, da realização de uma soma entre o sinal a ser invertido e outro sinal, chamado de calibrador. O sinal a ser invertido é apresentado como entrada inibitória, o sinal

calibrador é a entrada excitatória. Sendo assim, nos pontos em que se espera saída nula, os pulsos gerados pelas duas entradas anulariam-se mutuamente. Esta metodologia não funciona para o modelo de Wen-peng *et. al.* Nota-se, na Figura 3.4, que as amplitudes dos pulsos são diferentes e não há defasagem de 180° entre eles.

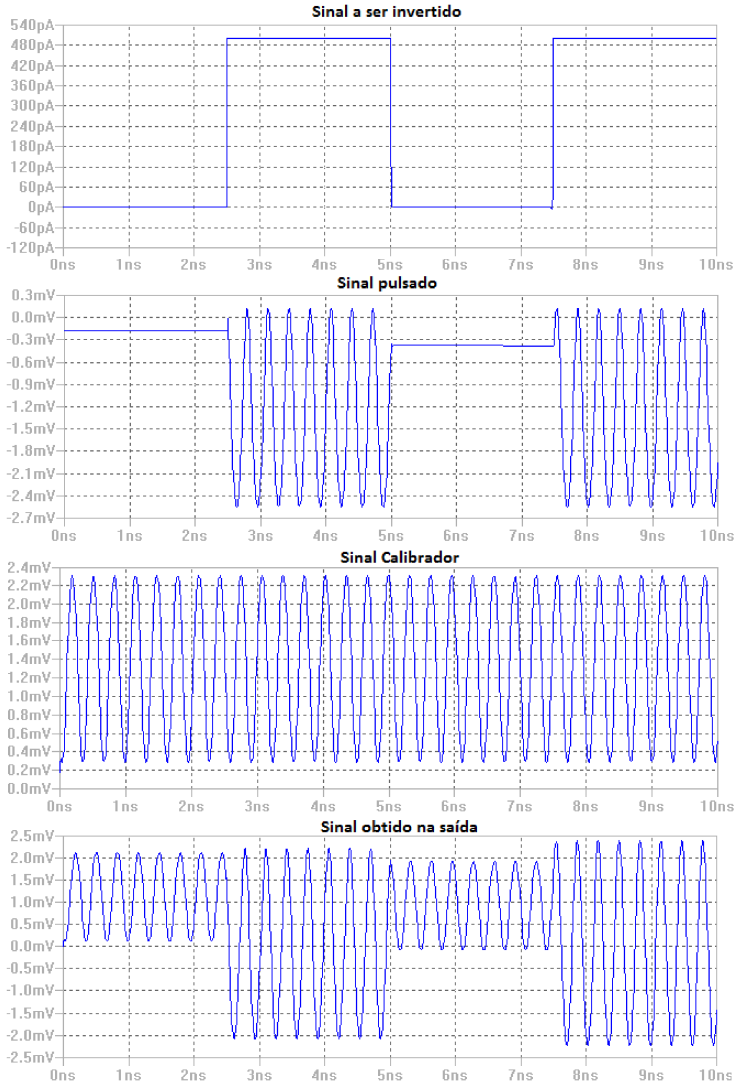

Figura 3.4: Tentativa de inversão de sinal.

Embora a Figura 3.4 possa sugerir que a codificação aumento da amplitude de pulso corresponde a 0 e redução da amplitude corresponde a 1 é viável, outras simulações revelaram que esta característica não se repete quando a frequência da entrada é alterada. Além disso, este padrão não foi observado na implementação de portas lógicas, que necessitam da interconexão de mais de um neurônio.

Assim, tornou-se necessário trabalhar com os pesos da rede, na tentativa de garantir que os pulsos gerados por entradas inibitórias teriam amplitudes reduzidas, podendo ser dife-

renciados daqueles gerados por entradas excitatórias. Após várias tentativas, chegou-se aos valores de 300 mV para entradas excitatórias e -100 mV para entradas inibitórias. O resultado foi bastante satisfatório permitindo, inclusive, a utilização da codificação: saída pulsada corresponde a nível 1 e não pulsada, a nível 0. Para melhor visualização, esta codificação foi exposta na Tabela 3.2.

Tabela 3.2: Codificação utilizada no trabalho

Saída	Nível Lógico representado
Pulsada, amplitude de pulsos da ordem de mV	1
Não pulsada, amplitude de pulsos da ordem de μV	0

É válido, contudo, ressaltar que, em todos os gráficos aqui apresentados, há pulsos nas regiões que aparentam ser contínuas. Porém, a amplitude desses pulsos, da ordem de alguns μV, é desprezível quando comparada com a amplitude dos pulsos em regiões que representam nível lógico igual a 1, da ordem de mV.

3.3 PROJETO DE PORTAS LÓGICAS

No artigo em que Wen-peng *et al.* propõem seu modelo de neurônio pulsante, o neurônio é mostrado em funcionamento isolado, respondendo corretamente aos estímulos apresentados. Contudo, para que tal modelo possa ser utilizado na construção de uma SNN, a funcionalidade dele quando em conjunto com mais neurônios precisa ser testada. Desta forma, prosseguiu-se à construção de portas lógicas através deste modelo. Foram construídas as seguintes portas lógicas: NOT/inversora, NAND/NÃO-E, OR/OU e, através da combinação destas, AND/E e, finalmente, a XOR/Ou-EXCLUSIVO. Todos os circuitos são derivados da elaboração de portas lógicas com neurônios pulsantes em tecnologia CMOS. A seguir, o modo como cada uma das portas foi desenvolvida é apresentado.

3.3.1 NOT - inversora

Para a construção de uma porta inversora utilizando um neurônio pulsante, Yellamraju *et al.*, [57], fizeram uso de uma propriedade bastante interessante, presente no modelo de neurônio de Liu [40]: quando uma entrada inibitória e uma excitatória possuem o mesmo peso, as respostas que o neurônio fornece para cada uma delas aniquilam-se mutuamente. Desse modo, a saída de um neurônio com uma entrada inibitória e uma excitatória de mesmo peso é nula. Isto ocorre, pois o valor de amplitude dos pulsos gerados por entradas inibitórias e excitatórias, para o modelo CMOS, é exatamente igual, e a defasagem entre estes pulsos é

de 180°. Por exemplo, considere que um peso de 100 mV gere pulsos com amplitude variando de 0 a 2,5 mV na saída do neurônio. Um peso de -100 mV, ou seja, uma entrada inibitória, produz na saída uma variação de 0 a -2,5 mV. Quando estas entradas são combinadas a saída será nula, desde que a defasagem seja igual a 180°. No modelo de Liu, esta defasagem existe, portanto, as saídas aniquilam-se mutuamente. Desse modo, para construir uma inversora, Liu sugere que um sinal que esteja sempre em nível lógico 1 seja conectado como uma entrada excitatória, este sinal recebe o nome de calibrador. O sinal a ser invertido será a entrada inibitória. Assim, os pulsos com amplitude negativa, gerados pelo sinal a ser invertido, serão anulados pelos pulsos gerados pelo sinal excitatório. O neurônio só pulsará quando o sinal a ser invertido estiver em nível 0 e a inversão será obtida.

O modelo de Wen-peng, contudo, apresenta defasagem entre os pulsos diferente de 180° e a amplitude dos pulsos gerados por entradas com sinais diferentes não é idêntica. Dessa forma, a aniquilação entre as entradas excitatórias e inibitórias não ocorre. Assim, para que a realização da porta inversora fosse possível, foram feitas algumas simulações a fim de se observar o comportamento do neurônio. A alteração dos pesos, conforme explicado na seção anterior, apresentou resultados satisfatórios. Dessa forma, aplicando pesos maiores, em módulo, às entradas excitatórias, nas regiões em que a saída deve estar em nível alto, o neurônio apresentará pulsos de amplitude consideravelmente maior (mil vezes maior que os pulsos nas regiões de saída em nível baixo). A Tabela 3.3 mostra os valores de pesos utilizados. Nesta tabela, V_- indica um peso negativo, ou uma entrada inibitória, e V_+ indica peso positivo, ou entrada excitatória. Estes valores foram utilizados em todos os circuitos desenvolvidos neste trabalho.

Tabela 3.3: Pesos utilizados.

Peso	Valor
V_-	300 mV
V_+	-100 mV

Assim, para implementar uma porta inversora, basta conectar o sinal que se deseja inverter como uma entrada inibitória e um sinal contrário ao que se deseja inverter como entrada excitatória, a este sinal é dado o nome de calibrador. A Figura 3.5 ilustra a ideia.

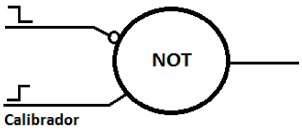

Figura 3.5: Esquema da realização de uma porta inversora com o modelo de neurônio de Wen-peng. O círculo na entrada indica uma entrada inibitória.

O circuito, construído no LTSpice, é mostrado na Figura 3.6. Da mesma forma que na simulação anterior, a entrada é uma fonte de corrente do tipo onda quadrada cujo valor máximo é de 500 pA e o período é de 10 ns.

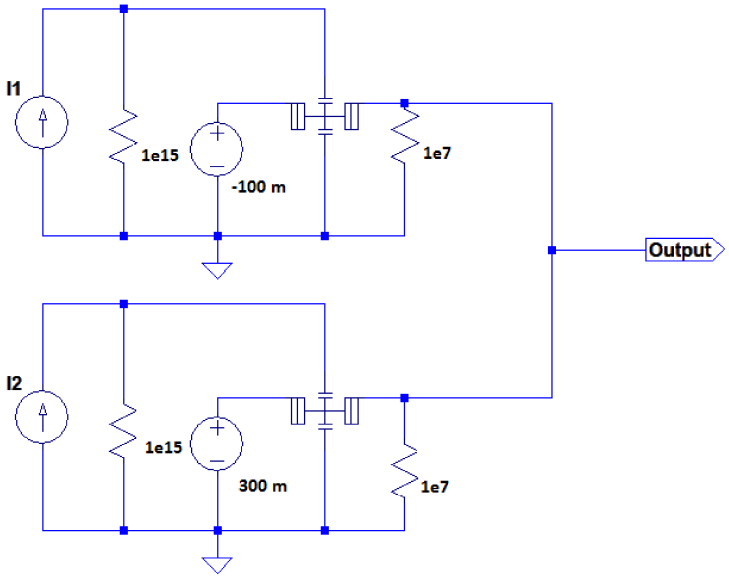

Figura 3.6: Circuito que implementa a porta inversora com dois neurônios.

A configuração dos pesos na Figura 3.6 demonstra a entrada que se deseja inverter, no caso, I_1, que é a entrada inibitória.

Há uma maneira mais simples de inverter o sinal de entrada, utilizando o neurônio nanoeletrônico. Conforme mostrado na seção 2.9, o neurônio é sensível a transições no sinal de entrada. A Figura 2.22 mostra, na realidade, a inversão do sinal de entrada devido à ocorrência de transição no início da simulação. Sendo assim, utilizando um único neurônio e forçando a ocorrência de transição no começo da simulação, a saída fornecerá o sinal invertido. Vale ressaltar que esta abordagem mostrou-se funcional para a inversão de um único sinal, ou seja, apresentando para o neurônio uma entrada A, obter-se-á na saída $\bar{A}$. Con-

tudo, caso se deseje a inversão de um sinal composto, por exemplo, X = A + B, a primeira abordagem, utilizando um sinal calibrador deve ser utilizada para obtenção de $\bar{X}$.

A Figura 3.7 apresenta a resposta do circuito que implementa a porta inversora. Os pulsos só ocorrem quando o sinal a ser invertido é nulo, o que configura a inversão. Esta figura foi obtida utilizando-se o processo de inversão por ocorrência de transição no sinal de entrada. O processo de inversão através de sinal calibrador será ilustrado na seção 3.3.4.

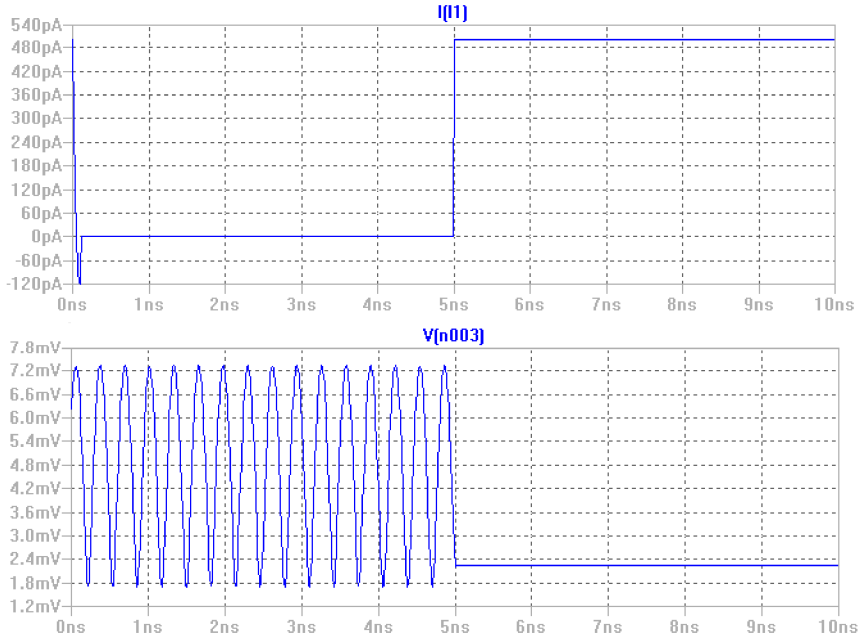

Figura 3.7: Resposta da porta NOT, com um neurônio, através da ocorrência de transição. I1 representa a entrada, do tipo onda PULSE. V(n003) é a saída fornecida pelo neurônio.

3.3.2 OR - OU

A porta OU é, dentre todas, a de mais fácil implementação. Basta conectar ao neurônio duas entradas com pesos positivos, excitatórias, e ele pulsará quando qualquer uma delas estiver em nível 1. Yellamraju *et. al* [57] alertam para a importância de realizar uma realimentação inibitória no neurônio. Esta ligação faz com que o neurônio volte ao estado relaxado quando as entradas voltam a zero. A Figura 3.8 mostra o esquema para a realização da porta OU.

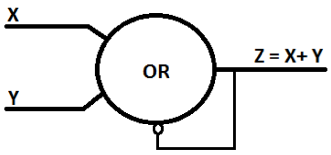

Figura 3.8: Esquema da realização de uma porta OU com o modelo de neurônio de Wenpeng. O círculo na entrada indica uma entrada inibitória [57] - modificada.

O circuito, construído no LTSpice, é mostrado na Figura 3.9. Nesta simulação, para que todas as combinações de entrada fossem obtidas, $00, 01, 10$ e 11, foram utilizadas duas fontes de corrente do tipo onda quadrada. Uma delas com período de 10 ns e outra com período de 5 ns. Ambas com valor máximo de 500 pA.

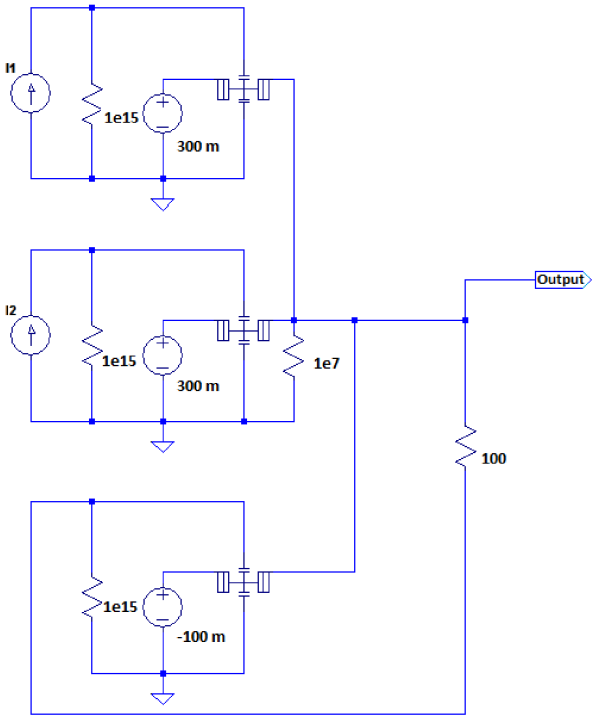

Figura 3.9: Circuito que implementa a porta OU com três neurônios.

A Figura 3.10 apresenta a resposta do circuito que implementa a porta OU. Os pulsos ocorrem quando qualquer uma das entradas está em nível 1 e quando ambas estão em nível 1.

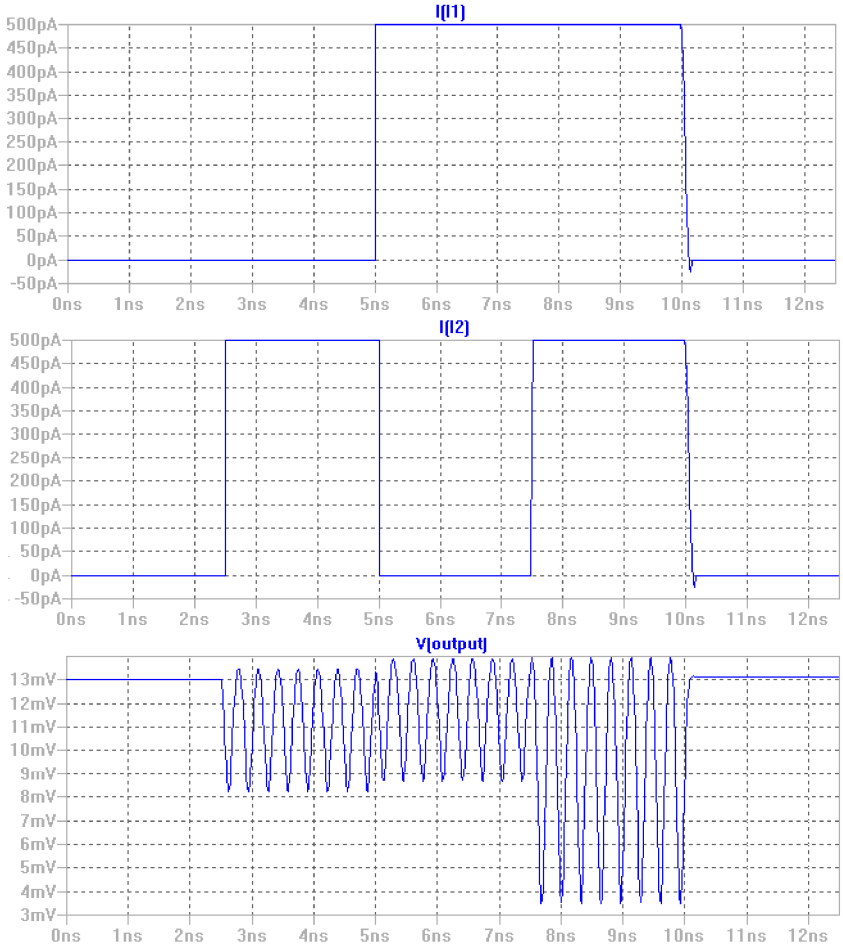

Figura 3.10: Resposta da porta OU implementada com 3 neurônios. As entradas, I1 e I2, são ondas quadradas com frequências de 100 MHz e 200 MHz, respectivamente.

3.3.3 NAND - NÃO-E

Para realizar uma porta NÃO-E, são necessárias duas entradas inibitórias, incluindo a realimentação, e uma excitatória. Os pesos das entradas inibitórias devem ser maiores, em módulo, do que o peso da entrada excitatória. Neste trabalho, utilizaram-se os valores demonstrados na Tabela 3.3. A Figura 3.11 ilustra o esquema para a implementação de uma porta NÃO-E.

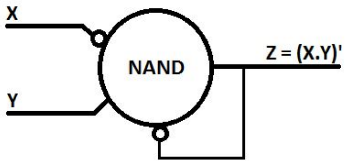

Figura 3.11: Esquema da realização de uma porta NÃO-E com o modelo de neurônio de Wen-peng. O círculo na entrada indica uma entrada inibitória.

O circuito, construído no LTSpice, é mostrado na Figura 3.12. Da mesma forma que para a porta OU, para que todas as combinações de entrada fossem obtidas, $00, 01, 10$ e 11, foram utilizadas duas fontes de corrente do tipo PULSE. Uma delas com período de 10 ns e outra com período de 5 ns. Ambas com valor máximo de 500 pA. O motivo da utilização da onda PULSE é o fato de que, uma porta NÃO-E precisa pulsar para a combinação de entradas 00. A única maneira de o SET ser excitado para entradas nulas é a ocorrência de uma transição.

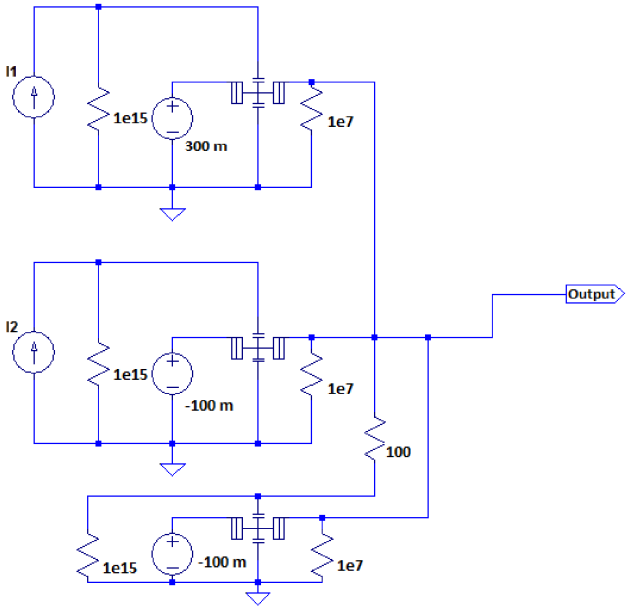

Figura 3.12: Circuito que implementa a porta NÃO-E com três neurônios.

A Figura 3.13 apresenta a resposta do circuito que implementa a porta NÃO-E. Os pulsos ocorrem para as combinações $00, 01$ e 10.

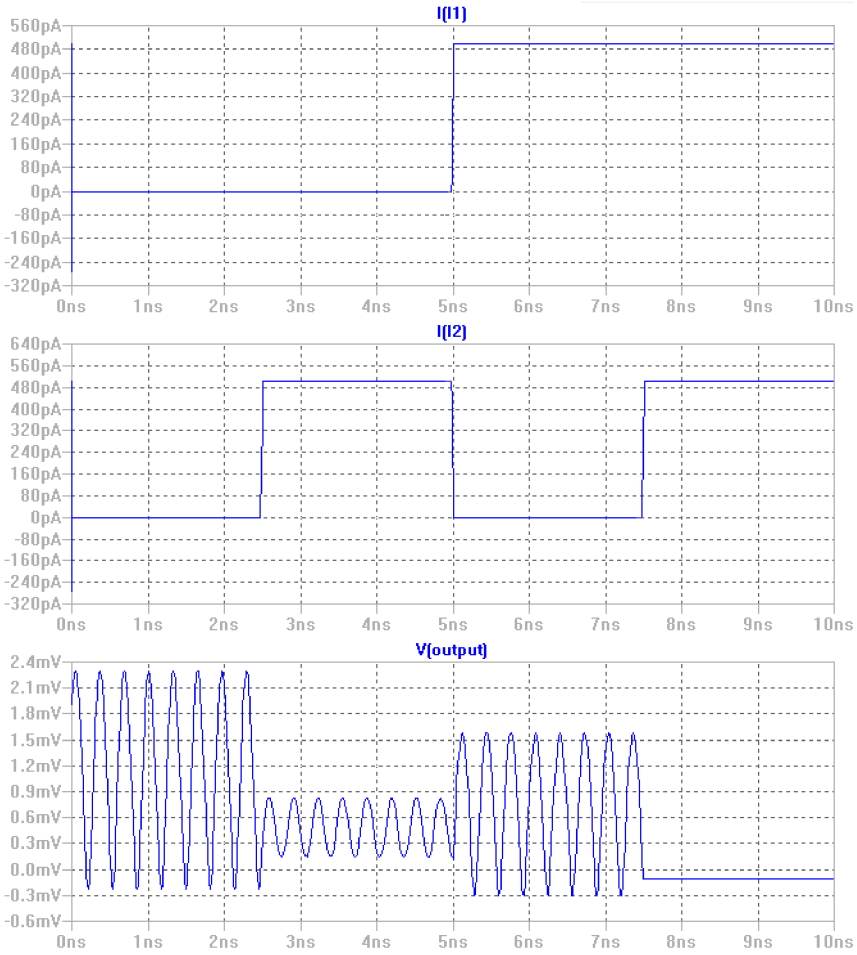

Figura 3.13: Resposta da porta NAND implementada com 3 neurônios. As entradas, I1 e I2, são ondas PULSE com frequências de 100 e 200 MHz, respectivamente.

3.3.4 AND - E

A realização da porta E foi possível através da combinação das portas NÃO-E e inversora. Basta ligar a inversora na saída da NÃO-E para se obter a E. Aqui se deve tomar o cuidado de escolher o sinal calibrador adequado para a inversora. Como se deseja inverter uma porta NÃO-E, cuja saída só é nula para a combinação de entradas 11, o calibrador deve possuir nível 1 para esta combinação e 0 para todas as outras. A Figura 3.16 ilustra o processo de inversão, e a Figura 3.14 ilustra o circuito final.

A utilização de uma onda PULSE para realizar a inversão da porta NÃO-E não é possível, pois, conforme explicado na seção 3.3.1, o sinal será uma composição de duas entradas: $X = \overline{A \cdot B}$, e quer-se obter $\bar{X}$.

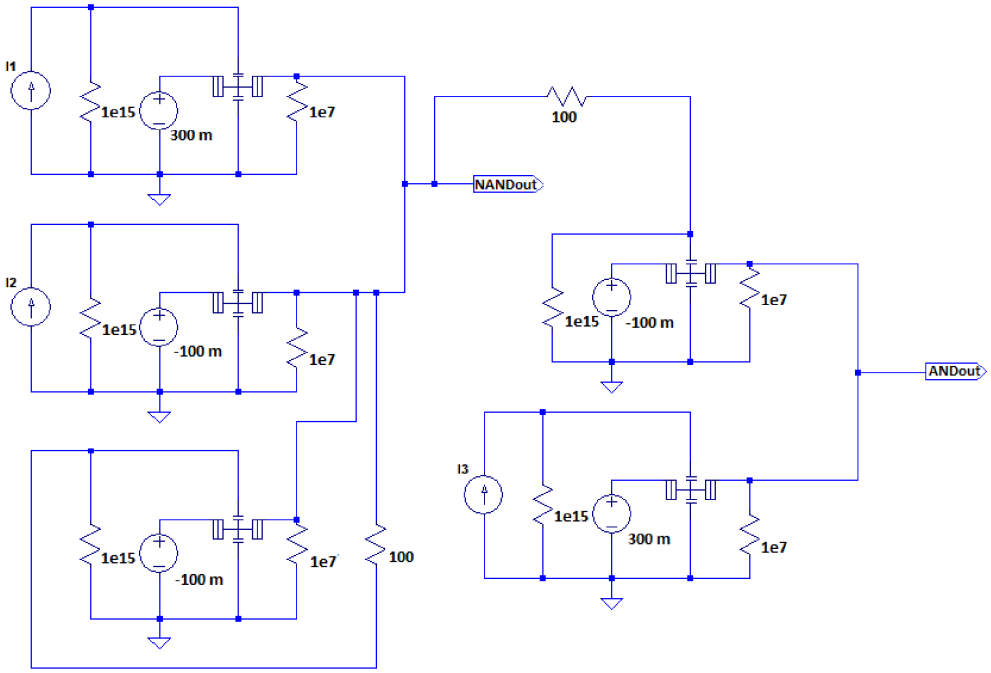

Figura 3.14: Circuito que implementa a porta E com cinco neurônios.

A Figura 3.15 apresenta a resposta do circuito que implementa a porta E. Os pulsos ocorrem quando ambas as entradas estão em nível 1.

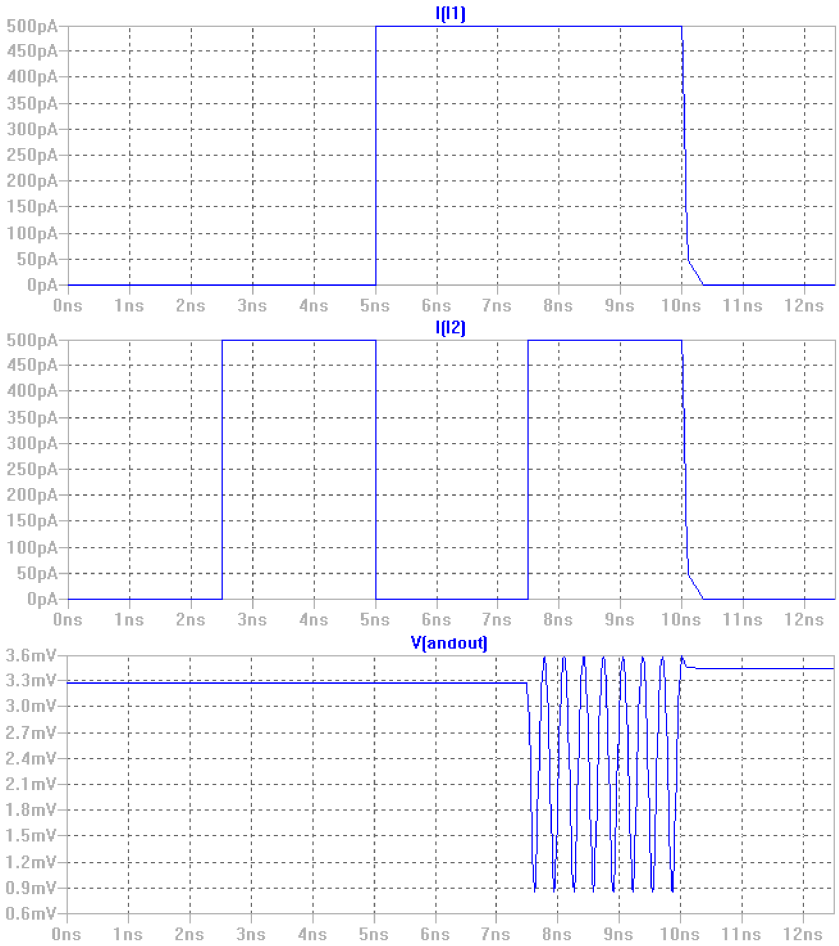

Figura 3.15: Resposta da porta AND implementada com 5 neurônios. As entradas, I1 e I2, são ondas quadradas com frequências de 100 e 200 MHz, respectivamente.

Esta porta foi construída através da aplicação de uma porta inversora na saída da porta NÃO-E, conforme a seção 3.3.4. Para ilustrar este processo, a Figura 3.16 apresenta a saída da porta NÃO-E, o sinal calibrador e o resultado da inversão.

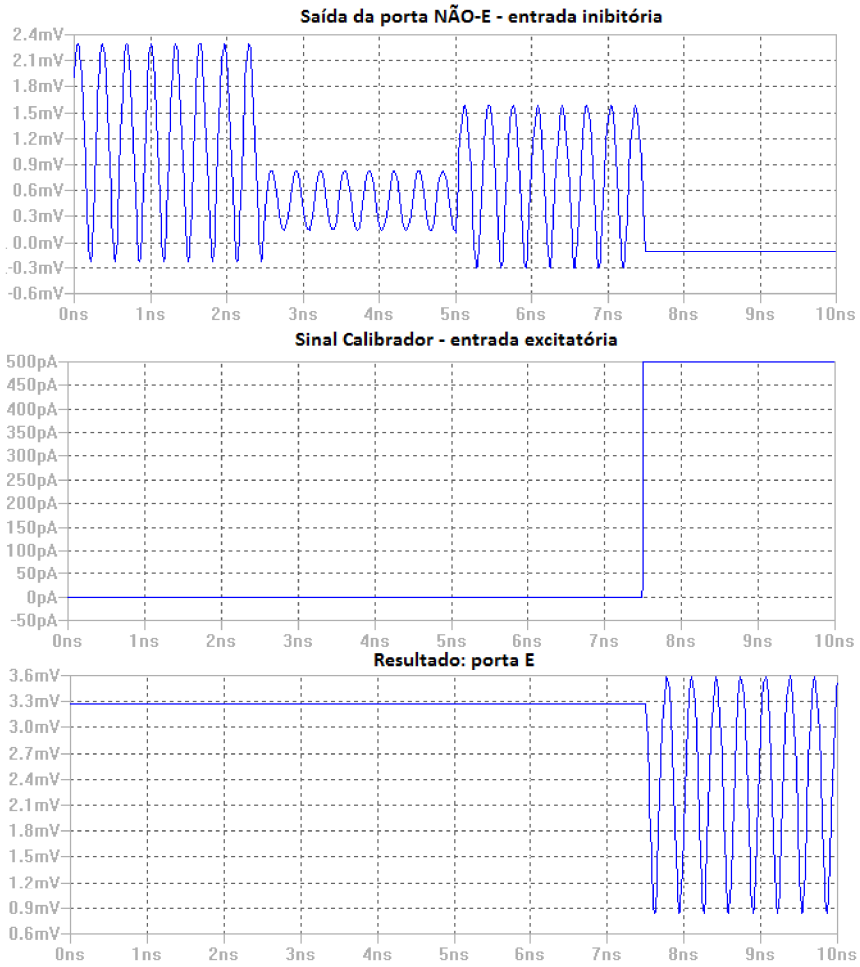

Figura 3.16: Inversão com uso de sinal calibrador: porta E.

A aproximação utilizada neste trabalho foi escolhida após uma série de tentativas de implementação da porta E de maneira direta. Todas estas tentativas mostraram-se infrutíferas. Quando o número de neurônios utilizados na tentativa de realização desta porta tornou-se muito alto (mais de 8 neurônios) a inversão da saída da porta NÃO-E configurou a opção mais enxuta. A Figura 3.17 ilustra a proposta de Yellamraju *et al.* para realização da porta E com o neurônio CMOS [57]. Analisando criticamente a proposta, percebe-se que não há duas entradas apresentadas para o neurônio, na verdade, o esquemático sugere que o neurônio efetua a operação E entre uma entrada e a própria saída. A tentativa de reprodução desta topologia não forneceu resultados adequados.

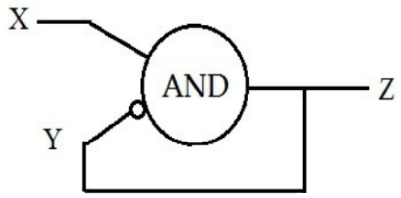

Figura 3.17: Esquema para realização da porta E proposto por Yellamraju *et. al* [57].

3.3.5 XOR - Ou exclusivo

A proposta de implementação da porta XOR, utilizando um modelo de neurônio pulsante CMOS, feita por Liu [40], é mostrada na Figura 3.18. Uma vez que $X \oplus Y = \bar{X} \cdot Y + X \cdot \bar{Y}$, na proposta de Liu, N1 e N2 são responsáveis pela função E e o neurônio N0 pela OU entre as saídas destas.

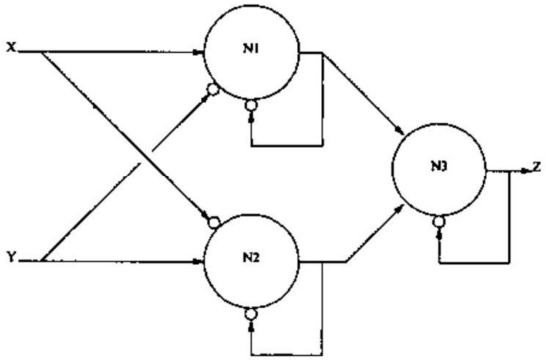

Figura 3.18: Esquema proposto por Liu *et al.* para implementação de uma porta XOR [40].

Há duas razões pelas quais este esquema não funciona para o modelo de neurônio de Wen-peng: o primeiro deles é a impossibilidade de se realizar a porta E de forma direta e o segundo é a ausência de aniquilação das entradas inibitórias e excitatórias, devido à defasagem entre os pulsos ser diferente de 180°, conforme discutido nas seções 2.9 e 3.3.1. Desta forma, uma vez que a melhor configuração para a realização da porta E possui 5 neurônios, foi efetuado um cálculo para verificar qual seria a melhor alternativa: implementar a XOR com a porta E ou fazer a conversão para a porta OU, através do teorema de DeMorgan, conforme as equações 3.1 e 3.2. Aqui, será considerada a melhor alternativa aquela que fizer uso do menor número de neurônios.

$$\overline{X \cdot Y} = \bar{X} + \bar{Y} \tag{3.1}$$

$$\overline{X + Y} = \bar{X} \cdot \bar{Y} \tag{3.2}$$

Dessa forma, utilizando DeMorgan, a função XOR é igual a:

$$\begin{aligned} X \oplus Y &= \bar{X} \cdot Y + X \cdot \bar{Y}, \\ \overline{X \oplus Y} &= \overline{\bar{X} \cdot Y + X \cdot \bar{Y}} \\ &= \overline{(\bar{X} \cdot Y)} \cdot \overline{(X \cdot \bar{Y})} \\ &= (X + \bar{Y}) \cdot (\bar{X} + Y) \end{aligned}$$

Contudo, este resultado é equivalente a $\overline{X \oplus Y}$, de forma que deve-se negar a expressão mais uma vez e aplicar de novo o teorema de DeMorgan para que se obtenha a expressão desejada:

$$\begin{aligned} X \oplus Y &= \overline{(X + \bar{Y}) \cdot (\bar{X} + Y)} \\ &= \overline{(X + \bar{Y})} + \overline{(\bar{X} + Y)} \end{aligned}$$

Agora tem-se a função XOR implementada através da utilização das funções inversora e OU. A Tabela 3.4 mostra o número de neurônios necessário para resolver o problema da XOR de maneira direta, ou seja, utilizando a função E. A Tabela 3.5 mostra a quantidade de neurônios para a solução utilizando DeMorgan.

Tabela 3.4: Número de neurônios necessário para resolver o problema da XOR de forma direta.

Entrada	Inversora	Função E	Saída (função OU)	Total
2 neurônios	4 neurônios	10 neurônios	1 neurônio	17 neurônios

Há, ainda, duas opções para a realização da inversão dos sinais: a abordagem através de um sinal calibrador ou a utilização de inversão pela ocorrência de transição. A segunda opção é, claramente, a mais enxuta, uma vez que utiliza apenas um neurônio, enquanto a primeira opção utiliza dois. Para ilustrar a diferença gerada no número de neurônios final da rede, pela utilização de uma ou outra abordagem, o circuito que realiza a XOR para 2 entradas fará uso da inversão através de sinal calibrador, e os circuitos que realizam a XOR para 3 e 5 entradas, usarão a inversão por ocorrência de transição no sinal de entrada.

Na Tabela 3.4, a camada de entrada é responsável por apresentar as duas entradas, X e Y, à SNN. A camada inversora obtém os sinais $\bar{X}$ e $\bar{Y}$, a camada da função E fornece $\bar{X} \cdot Y$ e $X \cdot \bar{Y}$. Por fim, a camada de saída, que é uma função OU, faz $\bar{X} \cdot Y + X \cdot \bar{Y}$. Esta rede necessita de um total de 17 neurônios, sendo que mais da metade destes neurônios, 10, são responsáveis por efetuar a função E.

Tabela 3.5: Número de neurônios necessário para resolver o problema da XOR utilizando DeMorgan.

Entrada	Inversora	Função OU	2° inversora	Saída (OU)	Total
2 neurônios	4 neurônios	2 neurônios	4 neurônios	1 neurônio	13 neurônios

Na Tabela 3.5, a camada de entrada é responsável por apresentar as duas entradas, X e Y, à SNN. A primeira camada inversora obtém os sinais $\bar{X}$ e $\bar{Y}$. A camada da função OU fornece $\bar{X} + Y$ e $X + \bar{Y}$. A segunda camada inversora gera os sinais $\overline{\bar{X} + Y}$ e $\overline{X + \bar{Y}}$. Por fim, a camada de saída, que é uma função OU, faz $\overline{(\bar{X} + Y)} + \overline{(X + \bar{Y})}$. Esta rede necessita de um total de 13 neurônios, sendo, portanto, uma rede mais enxuta que a demonstrada pela Tabela 3.4. Dessa forma, a XOR será feita utilizando o teorema de DeMorgan.

A Figura 3.19 mostra o circuito final da rede que realiza a função XOR com duas entradas, no caso, I_1 e I_2. As entradas foram implementadas com fontes de corrente do tipo onda quadrada, uma com período de 10 ns e outra de 5 ns. O valor máximo é de 500 pA. A Figura 3.20 apresenta a resposta do circuito que implementa a porta XOR. Os pulsos ocorrem nas combinações de entrada 01 e 10.

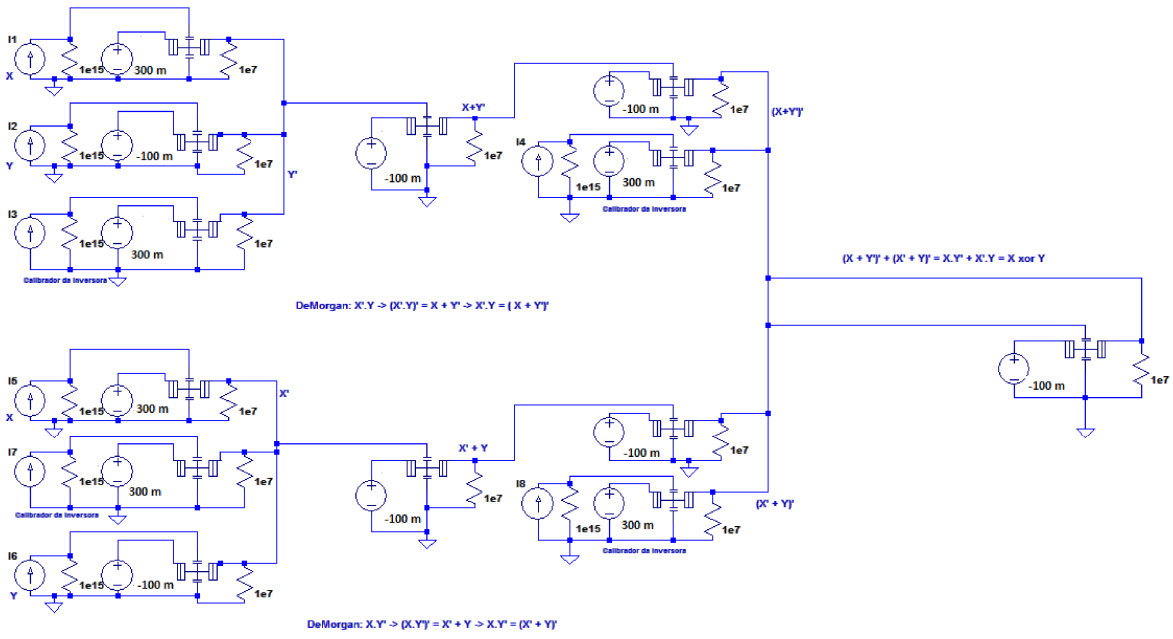

Figura 3.19: Circuito que implementa a porta XOR com 13 neurônios.

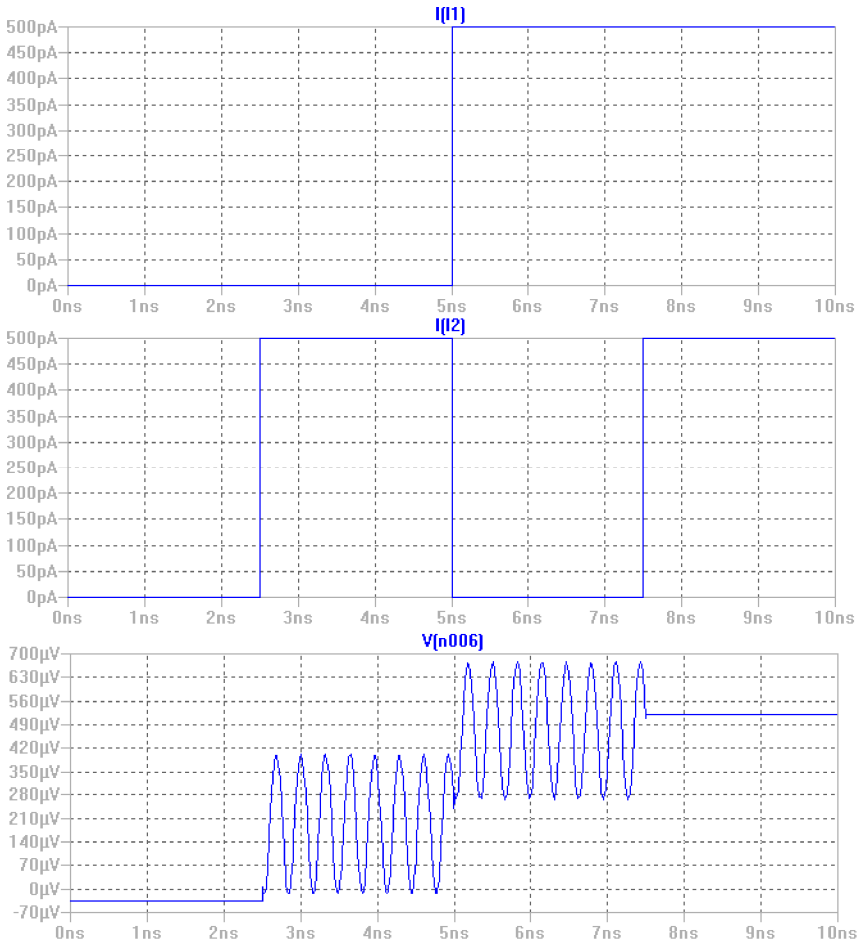

Figura 3.20: Resposta da porta XOR implementada com 13 neurônios. As entradas, I1 e I2, são ondas quadradas com frequências de 100 e 200 MHz, respectivamente. V(n006) é a saída do circuito.

4 REDES NANOELETRÔNICAS PULSADAS BASEADAS EM NOCS

"So wake me up when it's all over
When I'm wiser and I'm older
All this time I was finding myself
And I didn't know I was lost"
AVICII

Neste capítulo serão apresentados os procedimentos e resultados relativos às simulações que fazem uso dos blocos de construção aqui propostos.

4.1 ROTEADOR SIMPLES

O próximo passo deste trabalho consiste na implementação de um roteador simples. Este elemento é mostrado na Figura 4.1. Ele é chamado de roteador simples por dois motivos: primeiro, o roteador não conecta neurônios em todas as 4 direções (norte, sul, leste e oeste) e, segundo, não há um algoritmo de roteamento que maneje as entradas e saídas do roteador. Na verdade, este roteador apenas faz com que as entradas passem diretamente por ele, conforme a Tabela 4.1.

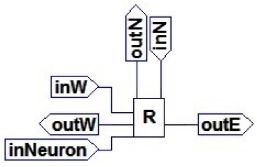

Figura 4.1: Elemento de roteamento simples com 3 entradas e 3 saídas. Os *labels* W, E e N identificam as direções Oeste, Leste e Norte, respectivamente. A entrada $inNeuron$ recebe os pulsos do neurônio.

Tabela 4.1: *Look-Up Table* de roteamento para o elemento da Figura 4.1.

Via de Entrada	Via de Saída
inW	$outW$
$inNeuron$	$outE$
inN	$outN$

A LUT que implementa esta lógica, no LTspice, está demonstrada na Figura 4.2.

```
.subckt Router_table inN inL inNeuron outN outL outW
B1 outN 0 V=(V(inN))
B2 outL 0 V=(V(inL))
B3 outW 0 V=(V(inNeuron))
```

Figura 4.2: Tabela de roteamento implementada no LTSpice IV.

4.2 XOR DE DUAS ENTRADAS COM ROTEADOR SIMPLES

Uma vez pronto o elemento roteador simples, prosseguiu-se ao teste do primeiro Bloco Neural desenvolvido. Conforme explicitado na seção 2.3.2, a implementação de SNNs com NoCs pressupõe um bloco que se repete em todos os nós da NoC. Este bloco é formado por um elemento processador e um elemento roteador. Neste trabalho, o elemento processador é o neurônio pulsante de Wen-peng, redimensionado para funcionar a $T = 300$ K, e o elemento roteador é o mostrado na Figura 4.1. A conexão entre esses dois elementos é mostrada na Figura 4.3.

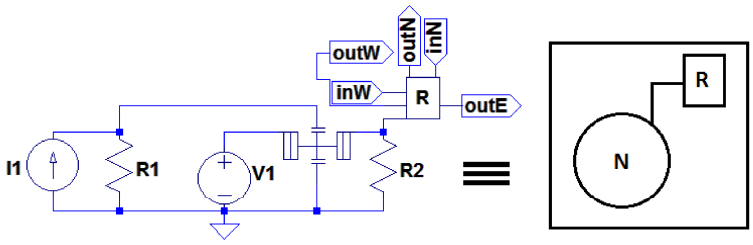

Figura 4.3: Bloco de construção da SNN.

Para realizar o teste deste bloco, ele foi utilizado na rede que realiza a porta XOR com 2 entradas, seção 3.3.5. Ou seja, foi acrescentado um elemento roteador a cada neurônio da rede de 13 neurônios mostrada na Figura 3.19. A Figura 4.4 ilustra o esquema de ligação e a Figura 4.5 mostra o circuito implementado no LTSpice IV.

A vantagem de utilização de uma NoC para interconexão da SNN fica clara quando se analisa a Figura 4.4. Em vez de se ter cada neurônio de uma camada conectado a todos os neurônios da camada seguinte, apenas uma interconexão comunica um neurônio com a próxima camada. Com o uso do roteador, esta rede continua sendo totalmente interconectada. O uso da NoC melhora muito o desempenho do circuito no que concerne à área ocupada e à potência dissipada [10]. Além disso, o uso do neurônio nanoeletrônico provê uma melhora ainda maior nestes dois parâmetros. Os neurônios que não se comunicam com as camadas subsequentes representam os sinais calibradores, necessários às portas inversoras.

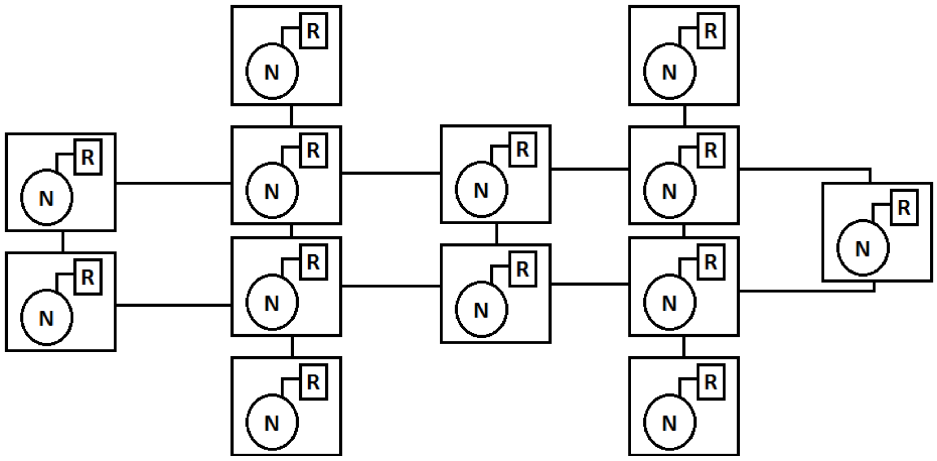

Figura 4.4: SNN nanoeletrônica construída através de uma NoC com arquitetura 2D-*mesh*.

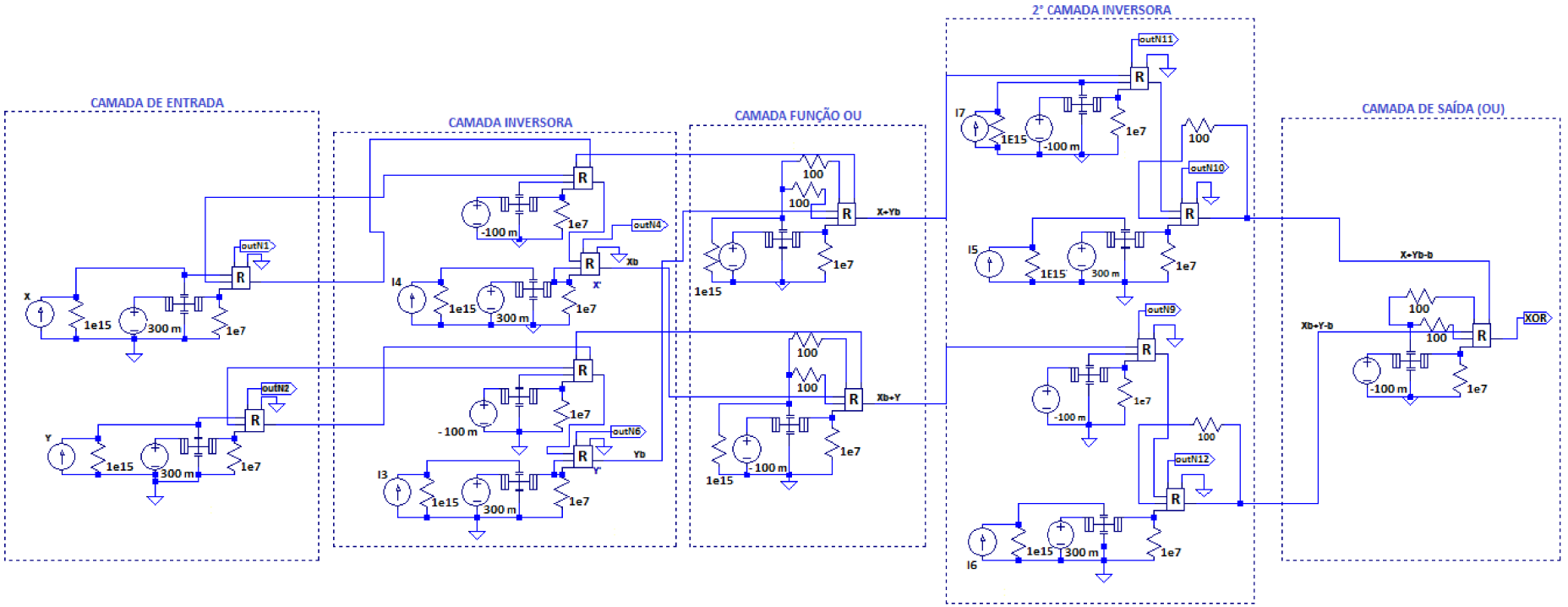

Figura 4.5: Circuito utilizado para testar o primeiro Bloco Neural, implementação da porta XOR com duas entradas no LTSpice IV.

A Figura 4.5 deixa claro o porquê deste roteador ser considerado um protótipo simples: embora haja alguns elementos no circuito que façam uso de todas suas entradas e saídas, a grande maioria (8 dos 13 roteadores) fica com suas entrada e saída na direção Norte ociosas. Isto ocorre devido ao tamanho da rede. Uma vez que ela pode ser considerada pequena, a interconexão se dá quase que exclusivamente na direção horizontal, Leste e Oeste. Neste circuito, as entradas foram apresentadas como duas ondas quadradas, com períodos de 5 ns e 10 ns. Os valores máximos são de 500 pA.

A Figura 4.6 apresenta a resposta do circuito que implementa a porta XOR com o uso do bloco de construção com roteador simples, Figura 4.3. Os pulsos ocorrem nas combinações de entrada 01 e 10.

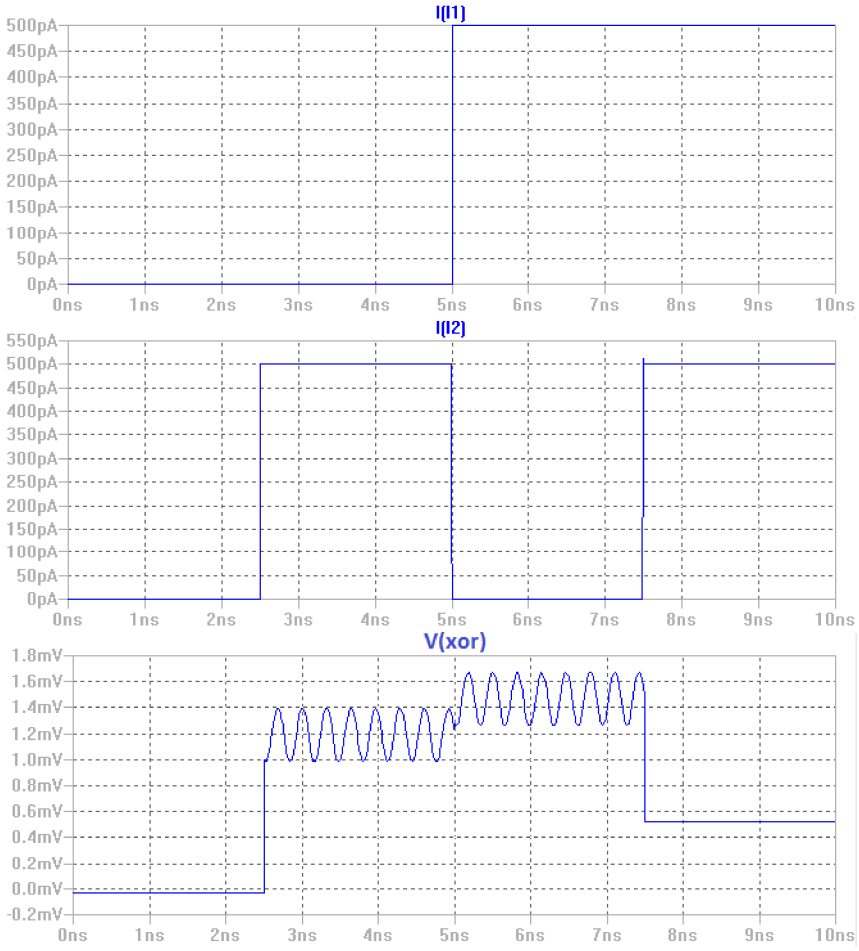

Figura 4.6: Resposta da porta XOR implementada com o primeiro bloco de construção proposto.

4.3 XOR COM 3 ENTRADAS

Para a realização da porta XOR com 3 entradas, fez-se uso dos teoremas de DeMorgan, da mesma forma que na seção 3.3.5. Vale lembrar que a porta XOR pode ser vista como um detector de paridade ímpar: toda vez que o número de entradas em nível alto for ímpar, a saída estará em 1. Após a aplicação dos teoremas de DeMorgan, Equações 3.1 e 3.2, chegou-se à seguinte expressão:

$$\overline{A \oplus B \oplus C} = \overline{(A + B + C)} + \overline{(A + \bar{B})} + \bar{C} + \overline{(\bar{A} + B + \bar{C})} + \overline{(\bar{A} + \bar{B} + C)}$$

Esta expressão foi obtida a partir da aplicação de lógica negativa diretamente na tabela verdade da função $A \oplus B \oplus C$. Assim sendo, os termos que produzem saída igual a 0 foram agrupados, e um produto de somas foi realizado com as entradas que geraram estes termos. Da mesma forma que para a XOR de 2 entradas, esta expressão precisa ser invertida novamente, para que se obtenha $A \oplus B \oplus C$. Por questões de simplicidade, e a fim de reduzir o tamanho do circuito utilizado, optou-se por inserir uma porta inversora na camada de saída da rede neural.

A Tabela 4.2 demonstra as camadas da SNN.

Tabela 4.2: Número de neurônios por camada para a XOR com 3 entradas.

Entrada	Função OU	Inversora	2° OU	Saída (inversora)	Total
6 neurônios	4 neurônios	4 neurônios	1 neurônio	1 neurônio	16 neurônios

Neste circuito foi utilizada a abordagem de inversão do sinal por ocorrência de transição. Os sinais, já invertidos, foram apresentados como entradas para a rede, por isso esta SNN possui 6 entradas: $A, B, C, \bar{A}, \bar{B}$ e $\bar{C}$. Nota-se que esta é uma solução mais enxuta, uma vez que, se fosse utilizada a inversão por sinal calibrador, a rede teria 19 neurônios. O circuito que implementa a XOR com 3 entradas é mostrado na Figura 4.7.

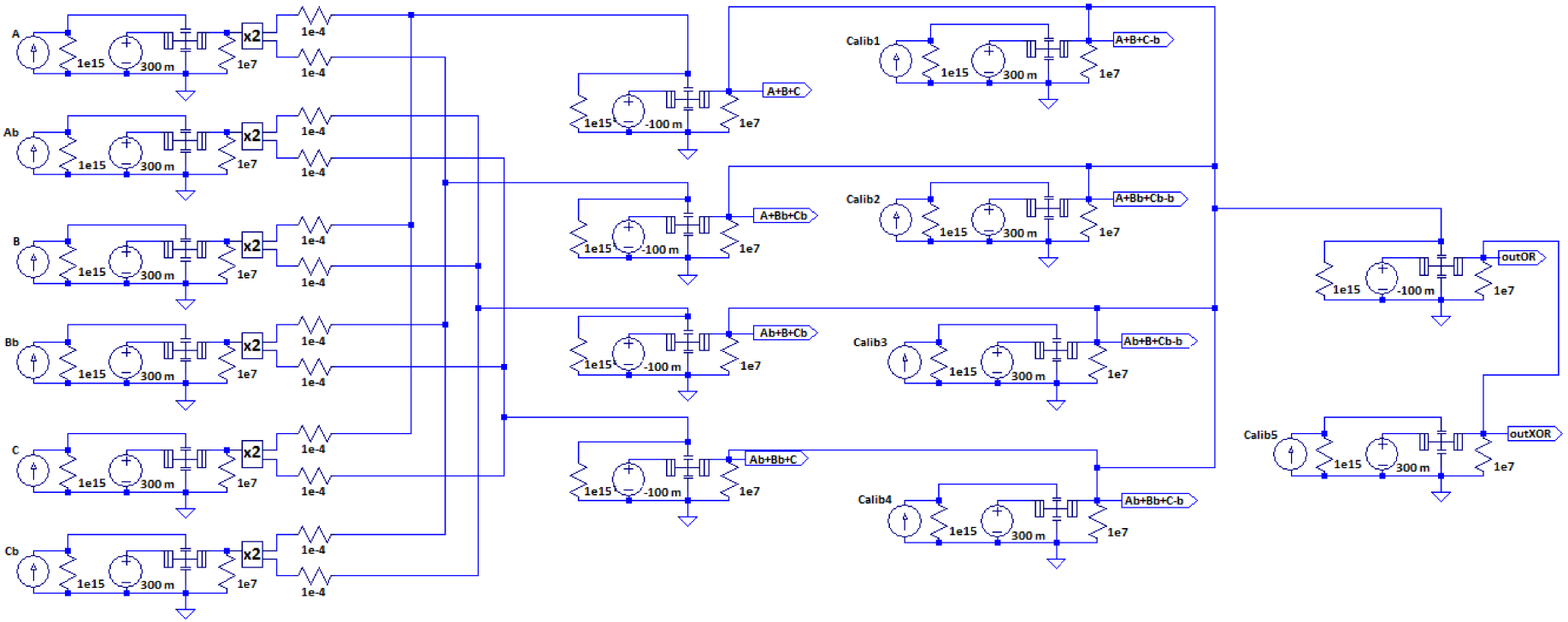

Figura 4.7: Circuito que implementa a XOR com 3 entradas no LTSpice IV.

Nesta figura, o componente marcado com "x2" apenas fornece dois ramos de saída. Foi necessária a inclusão deste componente, pois cada saída é necessária em dois neurônios da próxima camada. Por simplicidade, uma vez que os gráficos obtidos foram exatamente iguais, a resposta da XOR com 3 entradas está retratada na Figura 4.14 juntamente com a resposta obtida quando esta porta foi implementada com o roteador.

4.4 XOR COM 5 ENTRADAS

Os passos seguidos para a realização da porta XOR com 5 entradas são os mesmos descritos na seção anterior, para a XOR com 3 entradas. Desse modo, a expressão resultante da aplicação dos teoremas de DeMorgan é mostrada a seguir:

$$\begin{aligned}\overline{A \oplus B \oplus C \oplus D \oplus E} = \; & \overline{(A+B+C+D+E)} \quad +\overline{(A+B+C+\bar{D}+\bar{E})}+ \\ & +\overline{(A+B+\bar{C}+D+\bar{E})} \quad +\overline{(A+B+\bar{C}+\bar{D}+E)}+ \\ & +\overline{(A+\bar{B}+C+D+\bar{E})} \quad +\overline{(A+\bar{B}+C+\bar{D}+E)}+ \\ & +\overline{(A+\bar{B}+\bar{C}+D+E)} \quad +\overline{(A+\bar{B}+\bar{C}+\bar{D}+\bar{E})}+ \\ & +\overline{(\bar{A}+B+C+D+\bar{E})} \quad +\overline{(\bar{A}+B+C+\bar{D}+E)}+ \\ & +\overline{(\bar{A}+B+\bar{C}+D+E)} \quad +\overline{(\bar{A}+B+\bar{C}+\bar{D}+\bar{E})}+ \\ & +\overline{(\bar{A}+\bar{B}+C+D+E)} \quad +\overline{(\bar{A}+\bar{B}+C+\bar{D}+\bar{E})}+ \\ & +\overline{(\bar{A}+\bar{B}+\bar{C}+D+\bar{E})} \quad +\overline{(\bar{A}+\bar{B}+\bar{C}+\bar{D}+E)}\end{aligned}$$

A Tabela 4.3 demonstra o número de neurônios necessário para realizar a XOR com 5 entradas.

Tabela 4.3: Número de neurônios por camada para a XOR com 5 entradas.

Entrada	Função OU	Inversora	2° OU	Saída (inversora)	Total
10 neurônios	16 neurônios	16 neurônios	1 neurônio	1 neurônio	44 neurônios

A fim de facilitar a visualização do circuito que implementa a XOR com 5 entradas, ele foi aqui representado através de um esquemático simplificado, que pode ser visto na Figura 4.8.

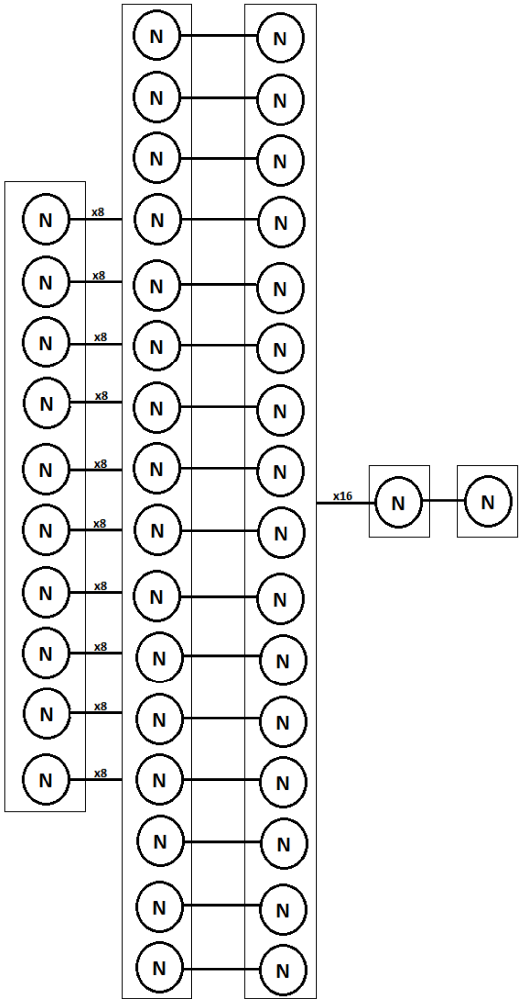

Figura 4.8: Esquemático do circuito que realiza a XOR com 5 entradas. As conexões marcadas com x8 repetem-se 8 vezes, e as com x16 repetem-se 16 vezes. As conexões sem marcação ocorrem apenas uma vez.

A resposta da XOR com 5 entradas pode ser vista na Figura 4.17, uma vez que é idêntica à resposta desta mesma porta quando implementada com o roteador.

4.5 ROTEADOR *FULL*

Conforme descrito na seção 2.6, o primeiro passo para qualquer algoritmo de roteamento consiste na atribuição de endereços para os pacotes. Dessa forma, o primeiro passo para a realização de um roteador capaz de interconectar neurônios nas quatro direções, e que possa

ser usado em redes maiores, é a definição de um esquema de endereçamento.

Neste trabalho, os endereços serão dados aos neurônios. Primeiramente, considerou-se atribuir um endereço binário a cada neurônio da rede, o roteador, posteriormente, compararia estes endereços e decidiria a via de saída mais adequada à sequência de pulsos recebida. Contudo, como será visto na seção 4.7, a rede que implementa a função lógica XOR com 5 entradas, possui 44 neurônios. Para que se possa gerar 44 endereços binários diferentes, seriam necessários 6 bits. A fim de realizar um circuito mais enxuto, optou-se por utilizar endereços em números decimais, que representam níveis de tensão.

O esquema de endereçamento funciona da seguinte maneira: cada rede possui 5 camadas, de modo que um intervalo foi atribuído para cada camada da rede. Por exemplo, considere que todos os endereços seriam representados por níveis de tensão entre 0 e 5 V. Assim, a primeira camada da rede seria representada pelo intervalo de 0 a 1 V. A segunda, de 1 a 2 V, a terceira de 2 a 3 V, a quarta de 3 a 4 V e, finalmente, a quinta de 4 a 5 V.

Considere a XOR com 3 entradas, a camada de entrada, segundo a Tabela 4.2, possui 6 neurônios. Para atribuir um endereço a cada neurônio, um intervalo de 1 V deve ser dividido por 6. Resultando nos seguintes endereços: 0,17 V; 0,34 V; 0,51 V; 0,68 V; 0,85 V e 1 V. Esta lógica foi seguida para todas as camadas das redes. Todos os endereços foram gerados com divisores de tensão. A Figura 4.9 ilustra a ideia.

Figura 4.9: Divisor de tensão que gera os endereços dos neurônios da primeira camada da rede que implementa a XOR com 3 entradas.

A Figura 4.10 mostra o símbolo, no LTSpice IV, utilizado para representar o roteador *full*. A Tabela 4.4 lista suas entradas e saídas.

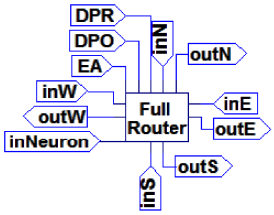

Figura 4.10: Roteador *full* desenvolvido no LTSpice IV.

Tabela 4.4: Entradas e saídas do Roteador *full* .

Via	Função
EA Endereço Atual	Entrada. Recebe o endereço atual do neurônio.
DPO Destino Pulso Originado	Entrada. Recebe o endereço do neurônio para o qual o pulso gerado se destina.
DPR Destino Pulso Recebido	Entrada. Recebe o endereço do neurônio para o qual o pulso recebido se destina.
inN	Entrada. Pode receber sinais de neurônios da mesma camada ou de outra camada.
outN	Saída. Pode enviar sinais para a mesma camada ou para camadas diferentes.
inE	Entrada. Pode ser usada quando um neurônio recebe mais de um sinal. Também pode ser usada quando há retroalimentação.
outE	Saída. Envia sinais para a próxima camada.
inS	Entrada. Pode receber sinais de neurônios da mesma camada ou de outra camada.
outS	Saída. Pode enviar sinais para a mesma camada ou para camadas diferentes.
inNeuron	Entrada. Recebe os pulsos gerados pelo neurônio no mesmo nó do roteador.
inW	Entrada. Recebe os sinais que devem excitar ou não o neurônio.
outW	Saída. Envia os sinais recebidos em inW para o neurônio.

Percebe-se, através da análise da Tabela 4.4, que apenas as entradas de endereçamento e a que recebe os pulsos do neurônio possuem funções fixas. Dessa forma, as outras entradas e saídas (N, S, E e W) ficam à disposição do usuário, permitindo que ele preencha a LUT do roteador de acordo com o circuito no qual o elemento será utilizado.

Os algoritmos de roteamento utilizados na implementação da XOR com 3 e com 5 entradas baseiam-se na comparação entre as entradas EA, DPO e DPR. O roteador *full* é capaz de realizar o encaminhamento de dois sinais: o recebido (DPR) e o gerado pelo neurônio ao qual ele se conecta (DPO). Um exemplo de como o roteador poderia encaminhar um pulso recebido são redes nas quais há comunicação entre camadas não adjacentes. Suponha uma rede na qual um pulso originado na primeira camada destina-se a um neurônio pertencente à terceira camada. Se a rede for interconectada através de uma NoC, não haverá ligação direta entre a primeira e a terceira camadas. Assim, os pulsos devem passar por um roteador na segunda camada, o qual compara os endereços e encaminha o pulso para a próxima camada.

Para decidir se o pulso recebido e o originado devem ser enviados para a próxima camada, para outro neurônio da mesma camada ou se se destinam ao próprio neurônio ao qual o roteador está ligado, o algoritmo de roteamento analisa o resultado de duas operações:
DPR - EA e DPO - EA. A seguir ilustram-se as possibilidades.

$$DPR - EA = \begin{cases} < 0 \\ 0 \\ > 0 \end{cases}$$

$$DPO - EA = \begin{cases} < 0 \\ 0 \\ > 0 \end{cases}$$

O caso em que a diferença entre DPR e EA, ou entre DPO e EA, é igual a zero é o mais fácil de tratar. Significa que o pulso encontra-se no neurônio de destino e assim, *outW* será igual à entrada que está recebendo o pulso (definida pelo usuário). Os outros dois casos, em que a diferença é maior ou menor do que zero, precisam que intervalos mais específicos sejam definidos, a fim de determinar se o pulso deve ser enviado para outra camada ou para outro neurônio, na mesma camada. Estes intervalos dependem dos valores de endereço determinados pelo usuário. Há, ainda, a possibilidade de não ocorrência de alguma situação. Por exemplo, em uma rede na qual não há retroalimentação, DPR - EA < 0 não ocorre.

Pode-se dizer que os algoritmos aqui gerados são fixos, ou seja, o melhor caminho não muda. Sempre que uma dada condição é satisfeita, a via de saída será a mesma. O encaminhamento se dá através da LUT, a qual não é atualizada, pois o caminho de conexão não é modificado.

A Figura 4.11 ilustra o bloco de construção composto pelo neurônio e pelo roteador *full*, bem como o símbolo que representará este bloco nos circuitos apresentados a partir desta

seção. A seguir serão explicitados os métodos de construção para as SNNs que resolvem a XOR de 3 e de 5 entradas, cada um destes circuitos usa uma LUT diferente para o roteador.
.

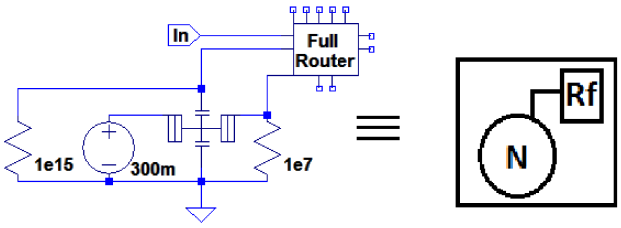

Figura 4.11: Bloco de construção para SNNs: neurônio como elemento processador conectado ao roteador *full*.

4.6 XOR DE 3 ENTRADAS COM ROTEADOR *FULL*

O esquema de endereçamento para a SNN que realiza a XOR com 3 entradas é mostrado na Tabela 4.5. Todos os endereços foram obtidos com divisores de tensão iguais ao da Figura 4.9. Para tanto, foram utilizados 5 divisores de tensão, um para cada camada da rede.

Uma vez determinados os endereços, pode-se simplificar as hipóteses que devem ser analisadas, com base na topologia da rede. Desse modo, foram utilizadas as seguintes características da rede para simplificar a LUT:

- Não há retroalimentação na rede;
- Não há comunicação entre camadas não adjacentes;

O algoritmo de roteamento precisa decidir em que situações cada via de saída será utilizada. Para o roteador *full*, existem 4 vias de saída que podem receber os pulsos: *outN, outS, outE* e *outW*. A Tabela 4.6 demonstra em que situações cada uma delas será utilizada.

A saída *outN* pode ser usada em duas situações, neste caso específico: quando o pulso recebido através de *inNeuron* destina-se a uma camada anterior àquela na qual o pulso se originou, ou quando o pulso recebido por *inW* está no nó de destino, mas o roteador recebe outro pulso, através de *inS*. A primeira situação é descrita por $DPO - EA < 0$. Pela topologia da rede, sabe-se que ela não possui retroalimentação, de modo que esta condição será sempre falsa. A segunda hipótese, na verdade, é a mais utilizada neste tipo de rede. Conforme pode ser visto na seção 4.3, cada porta OU, na segunda camada da rede, precisa receber 3 entradas. Estas funções OU são desempenhadas por neurônios conectados ao roteador *full*. A via de entrada principal para o sinal que deve excitar o neurônio é *inW*, mas, caso

Tabela 4.5: Endereçamento para a XOR com 3 entradas. Os N_i representam os neurônios de cada camada, i varia de 1 a 6.

Camada	Endereços [V]
Camada de entrada	0 a 1
N_1	1
N_2	0,85
N_3	0,68
N_4	0,51
N_5	0,34
N_6	0,17
Função OU	1 a 2
N_1	2
N_2	1,75
N_3	1,5
N_4	1,25
Inversora	2 a 3
N_1	3
N_2	2,75
N_3	2,5
N_4	2,25
2° OU	3 a 4
N_1	4
Saída	4 a 5
N_1	5

o neurônio precise de mais de um sinal, o roteador ainda pode receber estes sinais através de *inS*, *inE* e *inN*. Assim sendo, toda vez que o pulso se encontrar em seu local de destino, $DPR = EA$, o roteador disponibilizará, em *outN*, o sinal recebido através de *inS*. Assim, caso seja necessário enviar este pulso para outro neurônio, basta que o usuário faça a conexão de *outN* com uma via de entrada no roteador de destino.

A primeira condição que ativa a saída *outS*, $DPR - EA \leq 0$, também foi imposta para ilustrar um caso de retroalimentação. A diferença aqui é que, quando se usa DPR, há a sugestão de que o pulso precisa voltar mais de uma camada. Por exemplo, ele destina-se à camada 1 e encontra-se na camada 4. Novamente, esta é uma condição que, para esta rede, será sempre falsa. Assim, *outS* sempre será usada para passar o pulso recebido em *inW* para outros neurônios da mesma camada, $DPR - EA \leq 1$, que também precisem desta entrada.

A saída *outE* sempre envia os pulsos gerados pelo neurônio para a próxima camada. Devido à topologia da rede, sabe-se que os pulsos sempre se destinarão à próxima camada.

Tabela 4.6: Utilização das vias de saída do roteador *full* para a XOR com 3 entradas.

Via de saída	Condição	Pulsos transmitidos
outN	$DPO - EA < 0$ ou $DPR = EA$	*inS*
outS	$DPR - EA \leq 0$ ou $DPR - EA \leq 1$	*inW*
outE	Sempre verdade.	*inNeuron*
outW	$DPR = EA$	$inN + inS + inE + inW$

Finalmente, a saída *outW* está sempre conectada à entrada do neurônio, ver Figura 4.11. Embora a via de entrada principal seja *inW*, o neurônio pode receber mais de uma entrada. Daí o motivo pelo qual *outW* envia ao neurônio a soma de todos os sinais presentes nas vias de entrada do roteador. A Figura 4.12 ilustra a LUT implementada no LTSpice IV.

```
.subckt FullRouter2 EA DPO DPR inN inE inS inNeuron inW outN outE outS outW

B1 outN 0 V=IF(((V(DPO)-V(EA))<0)|(V(DPR)==V(EA)),V(inS),0)
B2 outE 0 V=V(inNeuron)
B3 outS 0 V=IF(((V(DPR)-V(EA))<=0)|((V(DPR)-V(EA))<1),V(inW),0)
B4 outW 0 V=IF((V(DPR)==V(EA)),V(inW)+V(inN)+V(inS)+V(inE),0)

.ends
```

Figura 4.12: LUT para a XOR com 3 entradas implementada no LTSpice IV.

O circuito que realiza a XOR com 3 entradas é mostrado, através de um esquemático simplificado, na Figura 4.13.

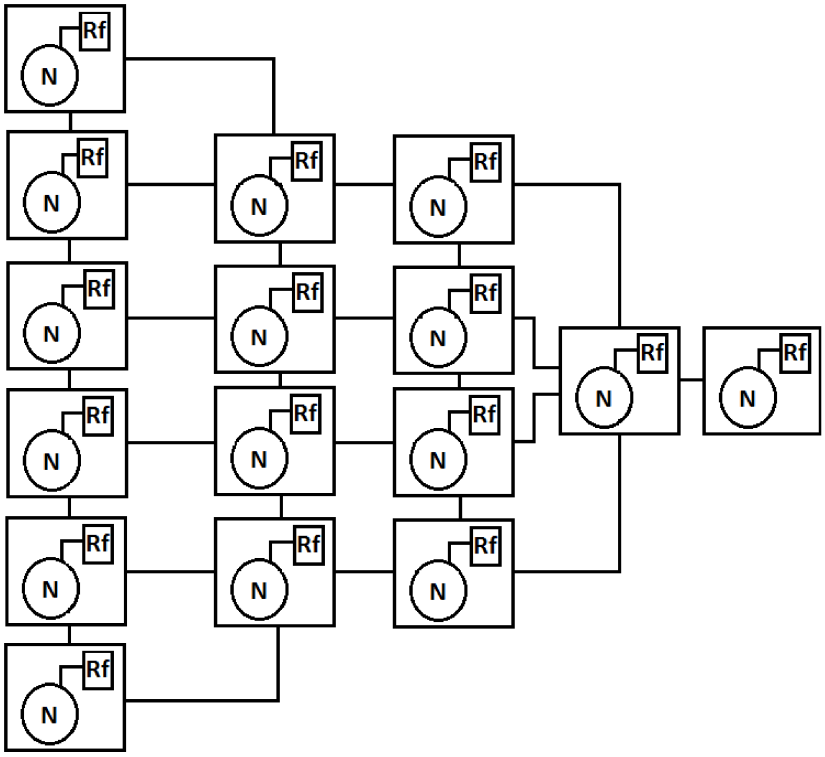

Figura 4.13: Esquema que representa o circuito para a XOR de 3 entradas com roteador *full*.

A Figura 4.14 apresenta a resposta do circuito que implementa a porta XOR com 3 entradas, através do uso do bloco de construção com roteador *full*, Figura 4.11. Os pulsos ocorrem nas combinações de entrada 001, 010, 100 e 111.

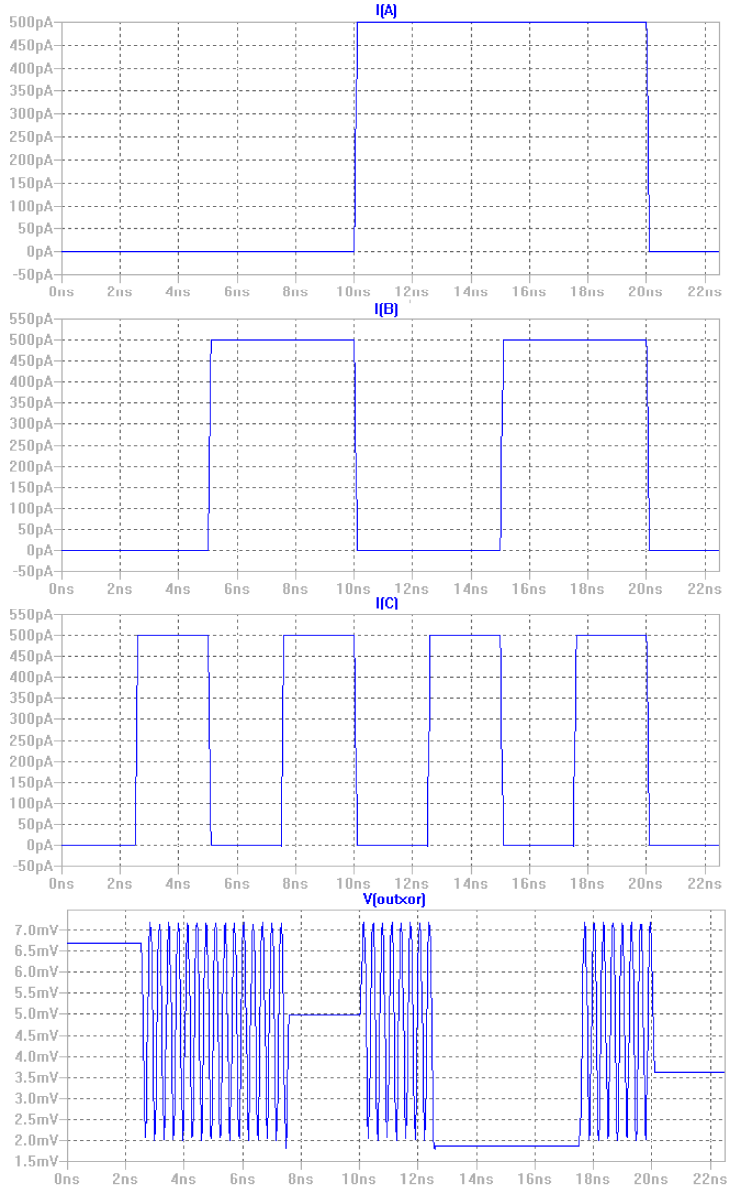

Figura 4.14: Resposta da porta XOR com 3 entradas, implementada com o segundo bloco de construção proposto. As entradas I(A), I(B) e I(C) são fontes de corrente do tipo onda quadrada com frequências de 50, 100 e 200 MHz, respectivamente

4.7 XOR DE 5 ENTRADAS COM ROTEADOR *FULL*

O esquema de endereçamento para a SNN que realiza a XOR com 5 entradas é mostrado na Tabela 4.7. Todos os endereços foram obtidos com divisores de tensão iguais aos da Figura 4.9. Para tanto, foram utilizados 5 divisores de tensão, um para cada camada da rede.

Para a XOR com 5 entradas, cabem as mesmas características da XOR com 3 entradas: não há retroalimentação e não há comunicação entre neurônios de camadas não adjacentes. A Tabela 4.8 demonstra as vias de saída do roteador, qual sinal elas disponibilizam e em que condições são usadas.

Tabela 4.7: Endereçamento para a XOR com 5 entradas. Os N_i representam os neurônios de cada camada, i varia de 1 a 16.

Camada	Endereços [V]
Camada de entrada	0 a 1
N_1	1
N_2	0,9
N_3	0,8
N_4	0,7
N_5	0,6
N_6	0,5
N_7	0,4
N_8	0,3
N_9	0,2
N_{10}	0,1
Função OU	2 a 3
N_1	3
N_2	2,938
N_3	2,875
N_4	2,813
N_5	2,75
N_6	2,688
N_7	2,625
N_8	2,563
N_9	2,5

Camada	Endereços [V]
Função OU	2 a 3
N_{10}	2,438
N_{11}	2,375
N_{12}	2,313
N_{13}	2,25
N_{14}	2,188
N_{15}	2,125
N_{16}	2,063
Inversora	4 a 5
N_1	5
N_2	4,938
N_3	4,875
N_4	4,813
N_5	4,75
N_6	4,688
N_7	4,625
N_8	4,563
N_9	4,5
N_{10}	4,438
N_{11}	4,375
N_{12}	4,313
N_{13}	4,25
N_{14}	4,188
N_{15}	4,125
N_{16}	4,063
2° OU	5 a 6
N_1	6
Saída	6 a 7
N_1	7

Analisando os endereços na Tabela 4.7, nota-se que a comunicação entre neurônios de camadas diferentes se dá quando $1 < DPO - EA < 3$. Para neurônios na mesma camada, a condição é $-1 < DPO - EA < 1$, e, para retroalimentação, $DPO - EA \leq -1$. Nota-se que foi estabelecida uma separação de 1 V entre os níveis de endereços de cada camada. Esta separação facilita consideravelmente a atribuição das condições do algoritmo de roteamento.

A saída *outN* possui funções semelhantes às descritas na seção anterior. No primeiro caso, em que $DPO - EA < 1$, o pulso recebido através de *inNeuron* deve ser enviado a um neurônio na mesma camada. Quando $DPR = EA$, a saída servirá para disponibilizar a

Tabela 4.8: Utilização das vias de saída do roteador *full* para a XOR com 5 entradas.

Via de saída	Condição	Pulsos transmitidos
outN	$DPO - EA < 1$ ou $DPR = EA$	*inS*
outS	$DPR = EA$	*inW*
outE	$DPO - EA < 0$ e $DPO - EA < 3$	*inNeuron*
outW	$DPR = EA$	$inN + inS + inE + inW$

entrada recebida em *inS*. Cabe ressaltar que, no caso da XOR com 5 entradas, esta funcionalidade é extremamente importante. Embora o roteador usado seja o mesmo, com o mesmo número de vias de entrada e saída, os neurônios da segunda camada da rede agora devem receber 5 entradas. Desse modo, uma vez que o roteador possui apenas 4 vias de entrada, não há alternativa a não ser conectar duas entradas juntas em uma mesma via.

A via de saída *outS* disponibiliza a entrada recebida em *inW*, da mesma forma que na rede com 3 entradas, há vários neurônios que necessitam do mesmo sinal de entrada. Dessa forma, a entrada recebida pela via principal, *inW*, deve estar disponível para ser enviada a outros nós da rede.

A saída *outE* envia os pulsos gerados pelo neurônio para a próxima camada. Aqui, fez-se uma análise dos endereços dos neurônios e percebeu-se que, na comunicação entre camadas, a diferença entre DPO e EA se encontra no intervalo entre 1 e 3 V.

Finalmente, da mesma forma que para a XOR de 3 entradas, a saída *outW* está sempre conectada à entrada do neurônio, ver Figura 4.11. Embora a via de entrada principal seja *inW*, o neurônio pode receber mais de uma entrada. Esta é a razão pela qual *outW* envia ao neurônio a soma de todos os sinais presentes nas vias de entrada do roteador. A Figura 4.15 ilustra a LUT implementada em LTSpice IV.

```
.subckt FullRouter2 EA DPO DPR inN inE inS inNeuron inW outN outE outS outW

B1 outN 0 V=IF(((V(DPO)-V(EA))<1)|((V(DPR)==V(EA))),V(inS),0)
B2 outE 0 V=IF((V(DPO)-V(EA)>=0)&((V(DPO)-V(EA))<=3),V(inNeuron),0)
B3 outS 0 V=IF((V(DPR)==V(EA)),V(inW),0)
B4 outW 0 V=IF((V(DPR)==V(EA)),V(inW)+V(inN)+V(inS)+V(inE),0)

.ends
```

Figura 4.15: LUT para a XOR com 5 entradas implementada no LTSpice IV.

O circuito que realiza a XOR com 5 entradas é mostrado, através de um esquemático simplificado, na Figura 4.16. Nesta figura, as conexões marcadas com "x2"ocorrem duas vezes, as sem marcação ocorrem apenas uma vez. A Figura 4.17 apresenta a resposta do circuito que implementa a porta XOR com 5 entradas, através do uso do bloco de construção com roteador *full*, Figura 4.11. Os pulsos ocorrem nas combinações de entrada 00001, 00010, 00100, 00111, 01000, 01011, 01101, 01110, 10000, 10011, 10101, 10110, 11001, 11010, 11100 e 11111.

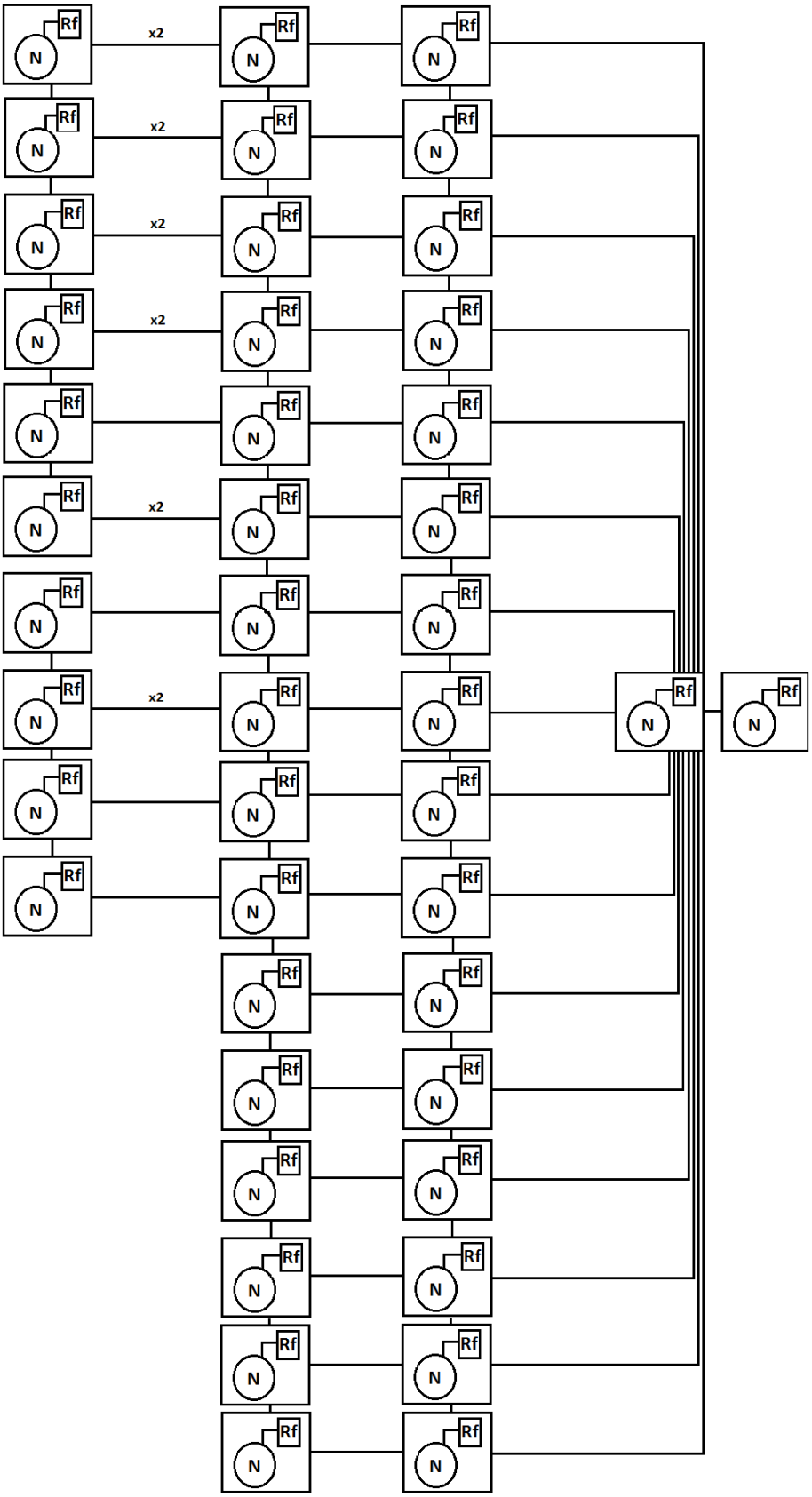

Figura 4.16: Esquema que representa o circuito para a XOR de 5 entradas com roteador *full*.

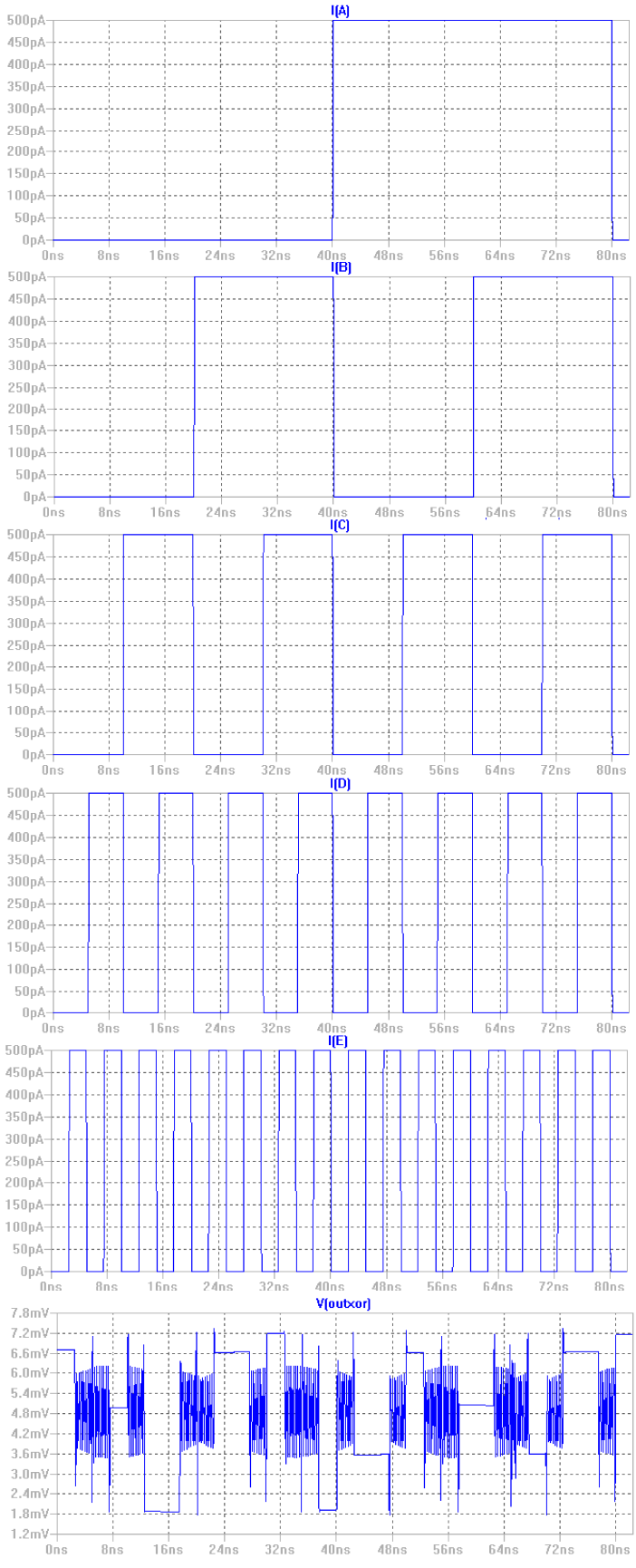

Figura 4.17: Resposta da porta XOR com 5 entradas, implementada com o segundo bloco de construção proposto. As entradas I(A), I(B), I(C), I(D) e I(E) são fontes de corrente do tipo onda quadrada com frequências de 12,5; 25, 50, 100 e 200 MHz, respectivamente

5 DISCUSSÃO DE RESULTADOS

"Minha força está na solidão.
Não tenho medo nem de chuvas tempestivas,
nem de grandes ventanias soltas.
Pois eu também sou o escuro da noite."
Clarice Lispector

Este capítulo dedica-se à discussão dos principais resultados obtidos durante as simulações realizadas neste trabalho.

5.1 VALIDAÇÃO DO MODELO DE NEURÔNIO DE WEN-PENG *ET. AL*

Para que o modelo de Wen-peng pudesse ser utilizado foi necessária a validação de dois principais comportamentos: funcionamento à temperatura ambiente e em conjunto com outros neurônios, constituindo uma rede neural. Considera-se que ambas as simulações foram bem sucedidas. Quanto ao funcionamento à temperatura ambiente, pode-se dizer que o circuito funcionou conforme o esperado uma vez que as adequações necessárias foram feitas (redução do módulo das capacitâncias e aumento dos valores das fontes de excitação). Contudo, no que concerne ao funcionamento em uma rede neural artificial, alguns aspectos observados merecem ser discutidos, são eles:

- Quanto à codificação da informação, o não cancelamento das saídas geradas por entradas inibitórias e excitatórias é fonte de algumas dificuldades. Primeiramente, a busca por uma nova codificação, talvez não tão intuitiva, poderia acabar por impossibilitar o uso deste modelo de neurônio pulsado. Neste trabalho, conseguiu-se fazer uso dos pesos da rede para que se chegasse a uma codificação tão intuitiva quanto à codificação de McCulloch. Contudo, ao se considerar grandes SNNs, como a de Theorachides *et. al* que possui mais de 100.000 neurônios, torna-se inviável trabalhar empiricamente com os pesos. Dessa forma, na construção de redes maiores, a necessidade de troca de codificação ou mesmo de utilização de outro modelo de neurônio pulsante é uma realidade.

- Quanto à construção de portas lógicas, a maior dificuldade ocorreu devido à sensibilidade à transição que o neurônio apresenta. Esta característica fez com que algumas portas lógicas pudessem ser implementadas através de excitações representadas por ondas quadradas convencionais, enquanto outras necessitavam da ocorrência de uma transição de estado no início da simulação. Tal diferenciação na implementação não

é desejável. Espera-se que a lógica de construção seja a mesma para todas as portas lógicas. Este problema é, contudo, mais difícil de ser solucionado. Uma vez que a sensibilidade à transição não foi observada somente no modelo de neurônio de Wenpeng, pode-se concluir que esta seja uma característica de dispositivos mono-elétron. Sendo assim, é aconselhável que o projetista desenvolva maneiras de utilizar este aspecto em seu favor, conforme foi aqui demonstrado na implementação de uma porta NOT através da ocorrência de transição.

5.2 REDES NANOELETRÔNICAS PULSADAS BASEADAS EM NOCS

No que concerne à construção das SNNs nanoeletrônicas, pode-se dizer que ambos os blocos de construção propostos tiveram sua funcionalidade comprovada. Os pontos que merecem destaque são: o roteamento via LUT, o endereçamento analógico e a utilização dos teoremas de DeMorgan para realizar a porta XOR.

Primeiramente, quanto ao roteamento através de LUT, pode-se concluir que esta representação é, devido à sua simplicidade, uma das melhores soluções para os algoritmos de roteamento. Propostas diferentes, como a de Harkin que utiliza *Round Robin* e a de Carrillo que apresenta um roteador adaptável ao fluxo, fazem uso da representação através de LUT. Dessa forma, mesmo que haja mudanças no algoritmo de roteamento, a LUT continua sendo uma das melhores maneiras de implementação. Neste trabalho, esta flexibilidade característica da LUT foi demonstrada através da implementação dos dois elementos roteadores: o simples e o *full*.

Quanto ao endereçamento analógico utilizado, podem ser levantadas questões relativas à robustez ao ruído desta solução. Assim, pode-se dizer que esta aproximação apresenta robustez razoável ao ruído. Esta afirmação pode ser feita já que, apesar de os endereços serem fixos, no roteamento feito pela LUT o algoritmo não precisa chegar ao valor exato de endereço. Ou seja, o roteamento se baseia em intervalos de tensão da ordem de alguns volts, sendo, portanto, robusto quanto a niveis normais de ruído.

A utilização dos teoremas de DeMorgan foi necessária uma vez que o circuito que implementa a porta AND precisou de 5 neurônios. Este número foi considerado alto para a realização de uma porta lógica com apenas duas entradas. Assim, a solução mais enxuta se deu através de portas NOT e OR, ou seja, os teoremas de DeMorgan. Contudo, ressalta-se que um equacionamento mais preciso para o SET, uma vez que o modelo de SET aqui utilizado não considera a natureza estocástica do tunelamento, pode resolver o problema.

6 CONCLUSÕES E PERSPECTIVAS FUTURAS

"Time...
Don't let it slip away.
Raise yo' drinkin' glass: here's to yesterday!
In Time we're all gonna trip away.
Don't piss Heaven off
We got Hell to pay."
Aerosmith

O objetivo principal deste trabalho foi a implementação de um bloco de construção nanoeletrônico para redes neurais pulsantes conectadas através de NoCs. Para que este objetivo pudesse ser atingido, foram dados vários passos intermediários que consistiram, basicamente, em testes com o modelo de neurônio nanoeletrônico de Wen-peng *et al.* Primeiro, foi necessário realizar a mudança de parâmetros, para que o funcionamento em temperatura ambiente fosse atingido. Posteriormente, portas lógicas foram implementadas a fim de testar o comportamento conjunto de mais de um neurônio. Pode-se dizer que estes passos foram completados com êxito, de modo que a funcionalidade do primeiro elemento do bloco de construção, o elemento processador, foi comprovada.

Algumas características de operação do neurônio devem ser ressaltadas, pois podem ser melhoradas. Dentre estas, a de maior importância é o fato de a defasagem entre os pulsos gerados por entradas inibitórias e excitatórias ser diferente de 180°. Possuir esta defasagem igual a 180° torna a construção de blocos lógicos intuitiva e simples. De fato, esta foi a maior dificuldade encontrada durante a implementação dos circuitos que realizam as portas lógicas. O ajuste no modelo de SET pode dar bons resultados quanto a este problema, uma vez que o valor de capacitância do *gate* do transistor possui grande influência no formato e na definição dos pulsos. Outra característica importante é o comportamento observado diante de transições. Sabe-se que a transição em um sinal de entrada gera perturbação suficiente para que o SET comece a apresentar as oscilações de Coulomb. Dessa forma, o neurônio "liga e desliga"conforme ocorrem as transições. Este comportamento permitiu que portas lógicas que exigem saída igual a 1 quando as entradas estão zeradas, como a NÃO-E, fossem implementadas. Porém, gerou uma diferença na lógica de construção: a porta NÃO-E não opera quando em sua entrada são colocadas ondas quadradas que não apresentem transição no começo. Esta limitação pode ser contornada pela utilização de ondas PULSE como excitação de entrada. Novamente, este problema existe devido à defasagem ser diferente de 180°. Um modelo de SET com equacionamento mais detalhado e com parâmetros bem ajustados pode ser decisivo na implementação de uma SNN nanoeletrônica.

O segundo elemento do bloco de construção proposto consiste em um roteador. Primei-

ramente, foi construído um roteador simples, cuja função se limita a encaminhar os pulsos recebidos de maneira fixa e incondicional: pulsos que cheguem pela direção norte, são encaminhados pela direção norte. Apesar da simplicidade deste primeiro elemento roteador, as simulações feitas com ele ajudaram a identificar problemas de simulação e convergência no LTSpice IV. Isto apontou para a necessidade de se ter uma ferramenta de simulação que forneça maiores recursos ao usuário. Dentre os *softwares* conhecidos de simulação de circuitos nanoeletrônicos, como o SIMON e o SECS, o LTSpice é o que apresenta maiores vantagens, como a facilidade na construção de circuitos densos e a quantidade de informação disponível *online*. Contudo, quando comparado com ferramentas completas, como o Matlab, é visível a menor quantidade de recursos que ele disponibiliza ao usuário.

Para que o bloco de construção proposto pudesse ser utilizado em redes neurais relativamente densas, a implementação de um roteador mais completo foi necessária. Dessa forma, foi construído o roteador *full*, o qual apresenta um lógica simples de comparação de endereços e é capaz de conectar neurônios nas quatro direções. A princípio, o objetivo foi desenvolver um bloco para ser utilizado em NoCs de arquitetura 2D-mesh. Contudo, através de uma conexão entre a saída norte e a entrada sul (ou vice-versa) o roteador é capaz de interconectar, também, redes em uma arquitetura 2D-torus. Além disso, através da implementação utilizando LUT, o mesmo símbolo de roteador pode ser usado em vários tipos de circuitos, basta que o usuário adapte a LUT à sua necessidade. Com o roteador *full* funcionando adequadamente, bastou conectá-lo à saída do neurônio para que o bloco de construção fosse implementado. A funcionalidade do bloco foi comprovada através da implementação de portas XOR com 3 e com 5 entradas.

Como perspectiva futura apresenta-se a realização de uma SNN nanoeletrônica capaz de realizar tarefas mais complexas, como a identificação de padrões. Para que este objetivo seja atingido, algumas melhorias precisam ser efetuadas, dentre elas;

- Nota-se a necessidade de uma ferramenta de simulação que forneça maiores recursos ao usuário. Desta forma, a implementação de um modelo de SET para funcionamento em Matlab apresenta-se como uma boa alternativa;

- Devido às limitações encontradas neste trabalho quanto ao modelo de neurônio utilizado, considera-se a troca de modelo. Por exemplo, o modelo proposto por Guimarães *et. al* considera o funcionamento do neurônio em pequenas SNNs, podendo constituir uma boa alternativa para o modelo de Wen-peng;

- Quanto ao elemento roteador, já foi proposto um roteador nanoeletrônico. Assim, pode-se considerar uma implementação totalmente nanoeletrônica;

- A multiplexação no tempo das vias de entrada e saída do roteador também é interessante. Esta divisão permite reduzir pela metade o número de vias, reduzindo também o número de interconexões;

- A proposta de Carrillo *et. al* considera um roteador capaz de selecionar o algoritmo de roteamento conforme o fluxo da rede. Considerar o uso de um roteador adaptável em grandes SNNs pode gerar ganhos no tempo de processamento;
- Por fim, ressalta-se a necessidade de implementação de um algoritmo de treinamento para a SNN. Em grandes redes, determinar os pesos empiricamente torna-se impraticável.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] B. M. S. M. Alencar, “Estudo sobre o desempenho de blocos básicos para o desenvolvimento de uma memória associativa nanoeletrônica,” Mestrado, Universidade de Brasília - UnB, 2012.

[2] A. Bindal e S. H. Hagh, “The impact of silicon nano-wire technology on the design of single-work-function cmos transistors and circuits,” Nanotechnology, vol. 17, pp. 4340 – 4351, 2006.

[3] M. Lundstrom, “Is nanoelectronics the future of microelectronics?” Proceedings of the 2002 International Symposium on Low Power Electronics and Design, ISLPED ’02., pp. 172 – 177, 2002.

[4] K. F. Goser, C. Pacha, A. Kanstein e M. L. Rossmann, “Aspects of systems and circuits for nanoelectronics,” Proceedings of the IEEE, vol. 85, pp. 558 – 573, 1997.

[5] J. G. Guimarães, “Arquiteturas de redes neurais nanoeletrônicas para processadores em escala giga ou tera,” Ph.D. dissertation, Universidade de Brasília - UnB, 2005.

[6] C. Wasshuber, “Single-electronics - how it works. how it’s used. how it’s simulated,” International Symposium on Quality Electronic Design, 2002. Proceedings., pp. 502 – 507, 2002.

[7] D. J. Paul, “Nanoeletronics,” Encyclopedia of physical science and technology, vol. 10, pp. 285 – 301, 2002.

[8] M. Akazawa, E. Tokuda, N. Asashi e Y. Amemiya, “Quantum hopfield network using single-electron circuits - a novel hopfield network free from local-minimum difficulty,” Analog Integrated Circuits and Signal Processing, vol. 24, pp. 51 – 57, 2000.

[9] C. P. S. M. Nogueira, “Rede de hopfield mono-elétron,” Bacharelado em Engenharia Elétrica, 2010.

[10] S. Carrillo, J. Harkin, L. McDaid, S. Pande, S. Cawley, B. McGinley e F. Morgan, “Advancing interconnect density for spiking neural network hardware implementations using traffic-aware adaptive network-on-chip routers,” Neural Networks, vol. 33, pp. 42 – 57, 2012.

[11] Z. L. Kovács, Redes Neurais Artificiais - Fundamentos e Aplicações. Livraria da Física, 2006.

[12] E. Oroski, "Identificação de falhas em espaçadores de linhas de transmissão utilizando visão estéreo e redes neurais artificais," Mestrado, Universidade de Brasília - UnB, 2011.

[13] J. Jiang, P. Trundle e J. Ren, "Medical image analysis with artificial neural networks," Computerized Medical Imaging and Graphics, vol. 34, pp. 617 – 631, 2010.

[14] J. Harkin, F. Morgan, L. McDaid, S. Hall, B. McGinley e S. Cawley, "A reconfigurable biologically inspired paradigm for computation using net work-on-chip and spiking neural networks," International Journal of Reconfigurable Computing, vol. 2009, pp. 1 – 13, 2009.

[15] J. G. Guimarães e A. R. S. Romariz, "Bio-inspired oscillators with single-electron transistors: Circuit simulation and input encoding example," Journal of Computational and Theoretical Nanoscience, vol. 10, pp. 1 – 5, 2013.

[16] S. Cawley, F. Morgan, B. Mcginley, S. Pande, L. Mcdaid, S. Carrillo, J. Harkin, "Hardware spiking neural network prototyping and application," Genetic Programming and Evolvable Machines, vol. 12, pp. 257 – 280, 2011.

[17] V. A. Pedroni, Circuit Design with VHDL. MIT Press, 2004.

[18] R. J. Tocci, N. S. Widmer e G. L. Moss, Sistemas Digitais: princípios e aplicações.Pearson Prentice Hall, 2007.

[19] L.P. Maguire, T.M. McGinnity, B. Glackin, A. Ghani, A. Belatrech e J. Harkin, "Challenges for large scale implementations of spiking neural networks on fpgas," Neurocomputing, vol. 71, pp. 13 – 29, 2007.

[20] A. DeHon e R. Rubin, "Design of fpga interconnect for multilevel metallization," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 12, pp. 1038 – 1050, 2004.

[21] L. Benini e G. DeMicheli, "Networks on chips: a new soc paradigm," Computer, vol. 35, pp. 70 – 78, 2004.

[22] B. Liu, "Architecture exploration of crossbar-based nanoscale reconfigurable computing platforms," Nano Communication Networks, vol. 1, pp. 232 – 241, 2010.

[23] S. Furber, S. Temple e A. Brown, "High performance computing for systems of spiking neurons," AISB06 workshop on GC5: architecture of brain and mind, vol. 2, pp. 29 – 36, 2006.

[24] J. Schemmel, J. Fieres e K. Meier, "Wafer-scale integration of analog neural networks," IEEE international joint conference on computational intelligence, pp. 431 – 438, 2008.

[25] T. Theocharides, G. Link, N. Vijaykrishnan, M. Invin e V. Srikantan, "A generic reconfigurable neural networks architecture as a network-on-chip," Proceedings of IEEE International SoC Conference, pp. 191 – 194, 2004.

[26] G. Tegart, "Nanotechnology: The technology for the 21° century," The second international conference on Technology foresight, vol. 2, pp. 1 – 12, 2003.

[27] B. S. Pês e J. G. Guimarães "A performance comparison of hamming and Hopfield single-electron neural networks," Proceedings of the 13th Microelectronics Students Forum, 2013.

[28] P. Narayanan, M. Leuchtenburg, T. Wang e C. A. Moritz, "Cmos control enabled single-type fet nasic," Symposium on VLSI, 2008. ISVLSI '08. IEEE Computer Society Annual, pp. 191 – 196, 2008.

[29] B. S. Pês e J. G. Guimarães, "A nanoelectronic building block for spiking neural networks," Proceedings of IEEE Nano 2014, 2014.

[30] C. P. S. M. Nogueira e J. G. Guimarães, "Pattern recognition based on autoassociative single-electron neural network," J. Comput. Theor. Nanosci, vol. 9, pp. 974 – 979, 2012.

[31] R. H. Chen, A. N. Korotkov e K. K. Likharev, "Single-electron transistor logic," Applied Physics Letters, vol. 68, pp. 1954 – 1956, 1996.

[32] J. P. Sun, G. I. Haddad, P. Mazumder e J. N. Schulman, "Resonant tunneling diodes: models and properties," Proceedings of the IEEE, vol. 86, pp. 641 – 660, 1998.

[33] G. L. Snider, A. O. Orlov e I. Amlani, "Quantum-dot cellular automata," J. Vac. Sci. Technol., vol. 4, pp. 1394 – 1398, 1999.

[34] G. T. Zardalidis e I. Karafyllidis, "Secs: A new single-electron-circuit simulator," IEEE Trans. on Circuits and Systems, vol. 55-I, no. 9, pp. 2774 – 2784, 2008.

[35] B. S. Pês, "Estudo sobre o desempenho de redes neurais nanoeletrônicas," Bacharelado em Engenharia Elétrica, 2012.

[36] C. Gerousis e D. Ball, "Single-electron tunneling circuits for image processing applications," Proceedings of CDES, pp. 139 – 144, 2008.

[37] L. Weng-peng, C. Xu e L. Hua-xiang, "A new hardware-oriented spiking neuron model based on set and its properties," ScienceDirect Physics Procedia, vol. 22, pp. 170 – 176, 2011.

[38] P. Beckett e A. Jennings, "Towards nanocomputer architecture," Australian Computer Science Communications, vol. 24, pp. 141 – 150, 2002.

[39] S. A. Kalogirou, “Applications of artificial neural-networks for energy systems,” Applied Energy, vol. 67, pp. 17 – 35, 2000.

[40] B. Liu, “A cmos neuron for vlsi circuit implementation of pulsed neural networks,” IECON 02 [IEEE 2002 28th Annual Conference of the Industrial Electronics Society], vol. 4, pp. 3182 – 3185, 2002.

[41] E. M. Izhikevich, “Simple model of spiking neurons,” IEEE transactions on Neural Networks, vol. 14, pp. 1569 – 1572, 2003.

[42] W. Gerstner e W. M. Kistler, Spiking Neuron Models: Single Neurons, Populations, Plasticity. Cambridge University Press, 2002.

[43] H. Paugam-Moisy and S. Bohte, “Computing with spiking neuron networks,” Handbook of natural computing, pp. 335 – 376, 2012.

[44] H. Shayani, P. Bentley e A. Tyrrell, “A cellular structure for online routing of digital spiking neuron axons and dendrites on fpgas,” Evolvable Systems: from biology to hardware, vol. 5216, pp. 273 – 284, 2008.

[45] T. Oya, T. Asai, R. Kagaya, T. Hirose e Y. Amemiya, “Neuronal synchrony detection on single-electron neural networks,” Chaos, solitons and fractals, vol. 27, pp. 887 – 894, 2006.

[46] Y. Kanazawa, Tetsuya Asai, M. Ikebe e Yoshihito Amemiya, “A novel cmos circuit for depressing synapse and its application to contrast-invariant pattern classification and synchrony detection,” International Journal of Robotics and Automation, vol. 19, pp. 206 – 212, 2004.

[47] E. A. Rietman, M. W. Tilden e M. Ashkenazi, Analog computation with rings of quasiperiodic oscillators: the microdynamics of cognition in living machines,” Robotics and Autonomous Systems, vol. 45, pp. 249 – 263, 2003.

[48] M. Saneei, A. Afzali-Kusha e Z. Navabi, “Low-latency multi-level mesh topology for nocs,” ICM ’06. International Conference on Microelectronics, 2006., pp. 36 – 39, 2006.

[49] P. Guerrier e A. Greiner, “A generic architecture for on-chip packet-switched interconnections,” Proceedings of the conference on Design, automation and test in Europe, pp. 250 – 256, 2000.

[50] S. Kumar, A. Jantsch, J.Soininen, M. Forsell, M.l Millberg, J. Öberg, K. Tiensyrjä e A. Hemani, “A network on chip architecture and design methodology,” Proceedings of IEEE Computer Society Annual Symposium on VLSI, pp. 105 – 112, 2002.

[51] W. J. Dally e B. Towles, “Route packets, not wires: on-chip interconnection network,” Proceedings of Design Automation Conference, pp. 684 – 689, 2001.

[52] R. Isermann e M. Münchhof, Identification of Dinamic Systems. Springer, 2011.

[53] A. S. Tanenbaum, Redes de Computadores – 4° ed. Elsevier, 2003.

[54] R. M. Kielkowski, Inside Spice – 2° ed. McGraw-Hill Professional, 1998.

[55] G. Lientschnig, I. Weymann e P. Hadley, "Simulating hybrid circuits of singleelectron transistors and field-effect transistors," Journal of Applied Physics, vol. 42, pp. 6467 –6472, 2003.

[56] R. Fahmy e K. Ismail, "Analysis of a single-electron decimal adder," Applied Physics Letters, vol. 70, pp. 2613 – 2615, 1997.

[57] S. Yellamraju, S. Kumari, S. Girolkar, S. Chourasia e A. D. Tete, "Design of various logic gates in neural networks," Proceedings of IEEE India Conference INDICON, pp. 57 – 60, 2013.