UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO, DE CIÊNCIAS EXATAS E EDUCAÇÃO
DEPARTAMENTO DE ENG. DE CONTROLE, AUTOMAÇÃO E COMPUTAÇÃO
CURSO DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Luiz Augusto Scheuermann França

Computação neuromórfica em nanoescala: desenvolvimento de arquitetura NoC baseada em tecnologia MOS para redes neurais

Blumenau
2023

Luiz Augusto Scheuermann França

Computação neuromórfica em nanoescala: desenvolvimento de arquitetura NoC baseada em tecnologia MOS para redes neurais

Trabalho de Conclusão de Curso de Graduação em Engenharia de Controle e Automação do Centro Tecnológico, de Ciências Exatas e Educação da Universidade Federal de Santa Catarina como requisito para a obtenção do título de Engenheiro de Controle e Automação.
Orientadora: Janaina Gonçalves Guimarães, Dra.
Coorientadora: Beatriz Oliveira Câmara da Fé, Ma.

Blumenau
2023

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

França, Luiz Augusto Scheuermann
Computação neuromórfica em nanoescala : desenvolvimento de arquitetura NoC baseada em tecnologia MOS para redes neurais / Luiz Augusto Scheuermann França ; orientadora, Janaina Gonçalves Guimarães, coorientadora, Beatriz Oliveira Câmara da Fé, 2023.
76 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Campus Blumenau, Graduação em Engenharia de Controle e Automação, Blumenau, 2023.

Inclui referências.

1. Engenharia de Controle e Automação. 2. Engenharia Neuromórfica. 3. Redes-em-Chip. 4. Nanoeletrônica. 5. Inteligência Artificial. I. Guimarães, Janaina Gonçalves. II. Fé, Beatriz Oliveira Câmara da. III. Universidade Federal de Santa Catarina. Graduação em Engenharia de Controle e Automação. IV. Título.

Luiz Augusto Scheuermann França

Computação neuromórfica em nanoescala: desenvolvimento de arquitetura NoC baseada em tecnologia MOS para redes neurais

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de "Engenheiro de Controle e Automação" e aprovado em sua forma final pelo Curso de Graduação em Engenharia de Controle e Automação.

Blumenau, 04 de Julho de 2023.

Banca Examinadora:

Documento assinado digitalmente
Janaina Goncalves Guimaraes
Data: 13/07/2023 09:26:09-0300
CPF: ***.475.831-**
Verifique as assinaturas em https://v.ufsc.br

Profa. Janaina Gonçalves Guimarães, Dra.
Universidade Federal de Santa Catarina

Documento assinado digitalmente
Adao Boava
Data: 13/07/2023 09:23:07-0300
CPF: ***.640.969-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Adão Boava, Dr.
Universidade Federal de Santa Catarina

Documento assinado digitalmente
Ciro Andre Pitz
Data: 13/07/2023 08:03:30-0300
CPF: ***.728.089-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Ciro André Pitz, Dr.
Universidade Federal de Santa Catarina

Dedico esta monografia aos meus avós *in memoriam*
Antônio Rogério França e Maria Imirena Waltrick França,
os quais acreditaram em mim desde o princípio.

AGRADECIMENTOS

Aos meus pais, Dorgel França e Beatriz Scheuermann, por sempre terem me incentivado e estarem ao meu lado durante toda a minha trajetória, saibam que vocês sempre foram a minha base e o meu refúgio de afeto e aconchego, sem vocês nada disso seria possível, tenho muito orgulho de ser o vosso filho.

À minha orientadora Janaína Guimarães, a qual carecem palavras para descrever o quão incrível és, mas que com toda convicção lhe afirmo, és a engenheira mais admirável que já conheci. Agradeço-lhe de coração por toda a paciência e prestatividade investidas durante a minha orientação, saiba que este trabalho é reflexo direto da sua motivação e apoio empregados na nossa busca rumo ao desconhecido.

À minha coorientadora Beatriz da Fé, por toda a sua disponibilidade e cordialidade em compartilhar seu conhecimento no decorrer deste trabalho.

A Tiago Busarello, professor pelo qual tenho grande admiração e respeito, por ter me dado o prazer de ter sido orientado por uma pessoa tão excepcional, bem como por todas as orientações e conselhos no decorrer da minha jornada.

Aos meus amigos Igor Schwartz, Eduardo Bruch, Gustavo Bruch, Lucas Gonçalves, Matheus Gabriel e Yanne Xavier, pelas memórias, diversão e momentos de descontração proporcionados a mim nesses últimos anos, vocês fazem parte disso.

À Louise Reips e Felipe Vieira, docentes excepcionais, exemplos de humildade e empatia, por sempre terem me norteado através dos desafios, meandros e prazeres da matemática, além de terem me proporcionado o prazer da vossa amizade.

À Rosilda Stürmer e Sidney Stürmer, professores admiráveis e fascinantes, os quais me recordo com especial carinho por terem me apoiado no início da minha jornada acadêmica.

Por fim, a todos aqueles que de direta ou indiretamente contribuíram para minha formação, ficam aqui declarados os meus sinceros agradecimentos.

E a estrada se dividiu em duas naquele bosque,
E eu escolhi a menos percorrida.
Isso fez toda a diferença.
Robert Frost, 1916

RESUMO

A aproximação aos limites físicos e tecnológicos resultante do processo de ultra miniaturização de transistores estabelece novos paradigmas no desenvolvimento de tecnologias de próxima geração. A engenharia neuromórfica busca implementar sistemas que aproveitem a natureza paralela e distribuída do cérebro, oferecendo soluções com poder de processamento superior aos sistemas computacionais tradicionais. A abordagem da computação neuromórfica integrada a redes-em-chip (NoCs) em tecnologia nanoeletrônica, representa novos passos na evolução tecnológica, superando os limites da Lei de Moore. Dessa maneira, a presente investigação tem como objetivo a implementação e avaliação de uma arquitetura neuromórfica de rede-em-chip em topologia *torus*, sendo baseada em tecnologia MOS de 16nm para redes neurais. O sistema foi implementado em software computacional LTspice visando a classificação do conjunto de dados Iris de Fisher. A alta taxa de acertos obtida na classificação do conjunto de dados afere a validade do uso de transistores em nanoescala para aplicações utilizando-se de inteligência artificial.

Palavras-chave: engenharia neuromórfica; redes-em-chip; nanoeletrônica; Iris de Fisher; inteligência artificial.

ABSTRACT

The approximation to the physical and technological limits resulting from the process of ultra miniaturization of transistors establishes new paradigms in the development of next generation technologies. Neuromorphic engineering seeks to implement systems that take advantage of the parallel and distributed nature of the brain, offering solutions with higher processing power than traditional computational systems. The approach of neuromorphic computing integrated to networks-on-chip (NoCs) in nanoelectronic technology, represents new steps in technological evolution, overcoming the limits of Moore's Law. Thus, the present investigation aims to implement and evaluate a neuromorphic network-on-chip architecture in *torus* topology, based on 16nm MOS technology for neural networks. The system was implemented in computational software LTspice aiming at the classification of Fisher's Iris data set. The high rate of controlled hits in the classification of the data set, validates the use of nanoscale transistors for applications using artificial intelligence.

Keywords: neuromorphic engineering; networks-on-chip; nanoelectronics; Fisher's Iris; artificial intelligence.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ANN	*Artificial Neural Network*
CMOS	*Complementary Metal-Oxide Semiconductor*
CNN	*Convolutional Neural Network*
DEMUX	Demultiplexador
DNN	*Deep Neural Network*
FFN	*Feed-Forward Network*
IP	*Elemento de Processamento*
LUT	*Look Up Table*
MOS	*Metal-Oxide Semiconductor*
MOSFET	*Metal-Oxide-Semiconductor Field Effect Transistor*
NMOS	*N-channel Metal-Oxide Semiconductor*
NoC	Rede-em-Chip
PMOS	*P-channel Metal-Oxide Semiconductor*
ReLU	*Retified Linear Unit*
RNA	Rede Neural Artificial
SoC	Sistema-em-Chip
ULSI	*Ultra Large Scale Integration*

SUMÁRIO

1 INTRODUÇÃO

Os estudos empregados para a compreensão do funcionamento do cérebro humano, têm sido há muito tempo fascinantes e desafiadores. A engenharia neuromórfica é uma área interdisciplinar que combina conceitos da neurociência e engenharia, com o intuito de projetar e desenvolver sistemas computacionais que se inspiram nos princípios e na estrutura do cérebro humano (YANG *et al.*, 2020; YOUNG *et al.*, 2019). Essa abordagem procura criar circuitos e arquiteturas de hardware especializados, imitando certos aspectos do funcionamento do cérebro, os quais ficaram conhecidos como chips neuromórficos (RAJENDRAN *et al.*, 2019).

Ao contrário dos computadores tradicionais, que são baseados em arquiteturas von Neumann, a engenharia neuromórfica busca implementar sistemas que aproveitem a natureza paralela e distribuída do cérebro (YANG *et al.*, 2020; RAJENDRAN *et al.*, 2019). Os chips neuromórficos são projetados para emular redes neurais eletroquímicas, utilizando unidades de processamento e comunicação inspiradas em neurônios e sinapses biológicas (YOUNG *et al.*, 2019; MEAD, 1990). Esses chips podem ser construídos através da aplicação de várias tecnologias, como circuitos analógicos, digitais ou híbridos. Um dos principais objetivos da engenharia neuromórfica é alcançar a eficiência energética e poder de processamento superior aos dos sistemas computacionais convencionais, o uso de novas interconexões de rede é essencial para o desenvolvimento de tal tecnologia (RAJENDRAN *et al.*, 2019).

Substituindo as tradicionais interconexões ponto a ponto que são características dos sistemas-em-chip (SoCs), as redes-em-chip (NoCs) surgiram como tecnologia de ultra integração em larga escala (ULSI - *Ultra Large Scale Integration*), com interconexão altamente distribuída e integrada (FÉ, 2017; AGARWAL; SHANKAR, 2009). Com as NoCs, os componentes de SoCs, tais como blocos de processamento e memória, são conectados por meio de uma topologia de rede, semelhante a ao que acontece em redes de computadores. Nesse modelo, cada componente é conectado a um roteador, o qual encaminha os pacotes de dados pela rede até o destino pretendido (AHMAD; SETHI, 2020).

As NoCs oferecem vantagens significativas em relação aos barramentos compartilhados, como menor consumo de energia, maior paralelismo, maior largura de banda e melhor tolerância a falhas (AHMAD; SETHI, 2020). Além disso, elas permitem a integração eficiente de componentes de IP, facilitando o diagnóstico e a depuração do sistema (SALMINEN; KULMALA; HÄMÄLÄINEN, 2009). Portanto, a abordagem distribuída de NoCs permite uma melhor escalabilidade, desempenho e flexibilidade no projeto de chips complexos (KUNDU; CHATTOPADHYAY, 2015).

A busca pela melhoria contínua da eficiência energética e da performance dos microprocessadores, demonstrou por muito tempo a validade inquestionável da Lei de Moore. Formulada em 1965 por Gordon Moore, co-fundador da Intel, a Lei de Moore é uma

observação empírica sobre o crescimento exponencial da capacidade de processamento e armazenamento dos circuitos integrados, principalmente no que tange aos microprocessadores. Sua premissa presumia que o número de transistores em um chip dobraria aproximadamente a cada 18 meses, resultando em um aumento significativo de eficiência e performance (MOORE, 2006).

Apesar da Lei de Moore ter tido um impacto profundo na revolução microeletrônica, impulsionando a inovação e o desenvolvimento de novas tecnologias, a aproximação aos limites físicos e tecnológicos estabelece novos paradigmas no desenvolvimento de dispositivos das próximas gerações (SANTOS PÊS, 2012). Consequência direta da Lei de Moore, a migração de tecnologias em microescala para nanoescala deu origem a uma nova abordagem na indústria microeletrônica: a tecnologia nanoeletrônica (FÉ, 2017).

A ultra miniaturização e a alta eficiência energética dos dispositivos nanoeletrônicos oferecem possibilidades emocionantes para avanços tecnológicos em várias áreas. A nanoeletrônica tem várias aplicações potenciais em campos como a eletrônica de consumo, medicina, setor elétrico, telecomunicações e computação (NELSON; SHIPBAUGH, 1995). A abordagem da computação neuromórfica integrada a redes-em-chip em tecnologia nanoeletrônica, representa novos passos na evolução tecnológica que vão muito além da Lei de Moore.

No entanto, é importante ressaltar que a nanoeletrônica, assim como a engenharia neuromórfica, ainda está em estágios iniciais de pesquisa e desenvolvimento. Muitos desafios técnicos e científicos precisam ser superados antes que as tecnologias nanoeletrônicas integradas em sistemas neuromórficos possam ser amplamente adotadas. Entretanto, iniciativas científicas e tecnológicas, tais como a abordada na presente investigação, buscam tornar a nanoeletrônica e a sua integração com sistemas neuromórficos uma realidade tangível no desenvolvimento de novas tecnologias.

1.1 OBJETIVOS

No que segue são apresentados os objetivos geral e específicos deste estudo, os quais nortearam o desenvolvimento das atividades aqui propostas.

1.1.1 Objetivo geral

A presente investigação tem por objetivo a implementação e avaliação de uma arquitetura neuromórfica de NoC em topologia *torus* baseada em tecnologia de semicondutores de metal-óxido (MOS), tendo como referência os modelos propostos nas investigações de Binas *et al.* (2016) e Binas *et al.* (2020). Para tal serão utilizados modelos preditivos disponíveis em tecnologia MOS de 16nm (atual), para avaliação de desempenho futuro.

1.1.2 Objetivos específicos

Visando-se a obtenção de resultados concretos para obtenção do objetivo enunciado anteriormente, esta monografia tem como objetivos específicos:

1. Implementar em software computacional via uso de metodologia hierárquica, o modelo neural;
2. Validar por simulação o funcionamento dos neurônios e a arquitetura de rede implementada;
3. Efetuar a avaliação de desempenho da rede neuromórfica no processamento de um problema clássico de classificação utilizando de redes neurais;

1.2 ORGANIZAÇÃO DO TRABALHO

O conteúdo da presente monografia encontra-se estruturado em 5 capítulos. No que segue, tem-se uma breve descrição dos mesmos:

1. **Introdução:** presente capítulo em que são apresentados o tema e a contextualização da investigação, seguidos dos objetivos e contribuições da proposta;
2. **Revisão Bibliográfica:** são dadas as bases conceituais para o entendimento desta investigação, sendo estruturada mediante a exploração dos seguintes conceitos: nanoeletrônica, redes-em-chip, redes neurais e tecnologia MOS.
3. **Metodologia:** são expostas as metodologias implementadas para o desenvolvimento do trabalho, bem como o método de avaliação dos circuitos desenvolvidos, acompanhado das considerações referentes a simulação computacional do sistema;
4. **Implementação, Análise e Discussão dos Resultados:** capítulo em que são abordados e discutidos os resultados obtidos no decorrer do desenvolvimento deste trabalho;
5. **Considerações Finais:** é apresentada a síntese final dos pontos levantados no presente estudo, sendo acompanhada das perspectivas voltadas a realização de estudos futuros.

2 REVISÃO BIBLIOGRÁFICA

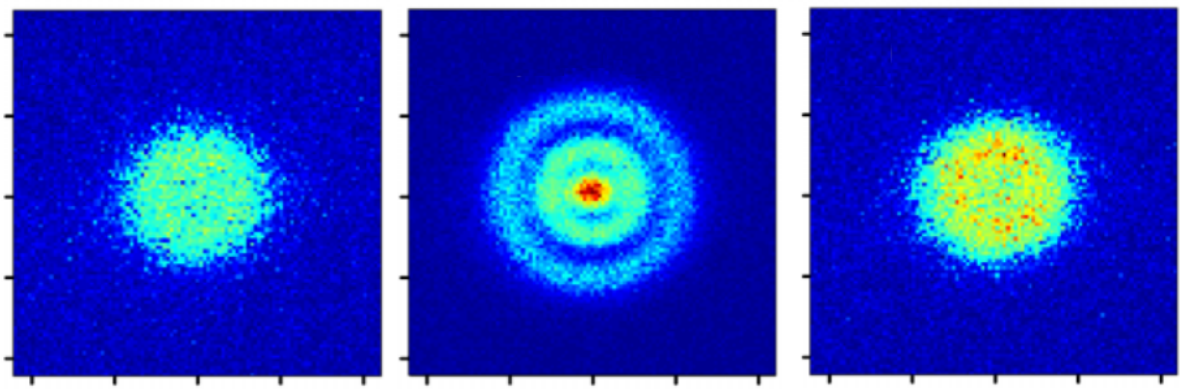
Sistemas neuromórficos são projetados com arquiteturas inspiradas nas redes neurais biológicas. Tais sistemas são compostos por diversas unidades de processamento de dados, os chamados neurônios artificiais, os quais se comunicam entre si por meio de conexões conhecidas como sinapses. Essas arquiteturas são altamente paralelas e distribuídas, permitindo um processamento eficiente de informações em tempo real (YANG *et al.*, 2020; YOUNG *et al.*, 2019).

A engenharia neuromórfica em nanoescala é uma área em constante evolução, impulsionada pela ultra miniaturização de componentes eletrônicos e pelos avanços na capacidade computacional. A compreensão da engenharia neuromórfica em dimensões tão diminutas requer o entendimento de diversos conceitos interdisciplinares. Portanto, este capítulo aborda uma breve introdução dos conceitos de nanoeletrônica, redes-em-chip, redes neurais e tecnologia MOS.

2.1 NANOELETRÔNICA

A nanotecnologia é o campo científico-tecnológico voltado à compreensão da matéria e objetos em nanoescala (PINA *et al.*, 2006). O termo nanotecnologia foi introduzido pela investigação pioneira de Taniguchi (TANIGUCHI, 1974), o qual visava o entendimento de objetos com tamanho abaixo da microescala (NELSON; SHIPBAUGH, 1995). Nanômetro é a unidade de medida base da nanotecnologia e corresponde à bilionésima parte do metro ($1\text{nm} = 10^{-9}\text{m}$). Os átomos (ver Figura 1), que são os blocos fundamentais da matéria, medem apenas um décimo de nanômetro (0,1 nm) (HANSON, 2008; PINA *et al.*, 2006). Em um domínio com dimensões tão pequenas, há o prevalecimento da física quântica sobre a física clássica (FÉ, 2017).

Figura 1 – Observação experimental do primeiro registro fotográfico de um átomo de hidrogênio via observação direta.



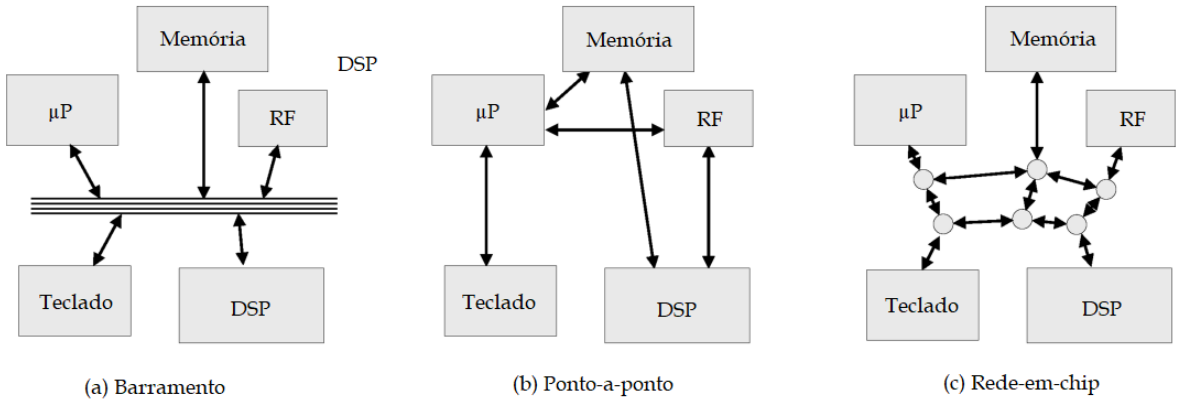
Fonte: Adaptado de Stodolna *et al.* (2013).

A nanoeletrônica consiste na aplicação de conceitos de nanotecnologia à eletrônica (FÉ, 2017). A nanoeletrônica surgiu da necessidade da miniaturização dos dispositivos eletrônicos além do horizonte da microeletrônica. Portanto, a nanoeletrônica surge como uma alternativa para superar os limites físicos já atingidos pelos dispositivos microeletrônicos, decorrente do processo de ultraminiaturização dos circuitos integrados (SANTOS PÊS, 2012). A transição da microeletrônica para a nanoeletrônica requer o entendimento do funcionamento de dispositivos novos e já existentes, como os dispositivos MOS, em escala nanométrica.

2.2 REDES-EM-CHIP

Os sistemas-em-chip, *Systems on Chip* (SoCs), referem-se a integração de diversos núcleos de processamento (IPs) em um único chip de silício. A tecnologia de SoCs é predominante em diversos dispositivos eletrônicos atuais, tais como sistemas embarcados, computação *mobile* e computadores pessoais. Tais sistemas são resultado do processo de ultra integração em larga escala (ULSI - *Ultra Large Scale Integration*), consequência direta da busca pela miniaturização e integração de componentes eletrônicos. Apesar dos contínuos avanços na tecnologia, as formas de conexão tradicionais aplicadas a SoCs, tais como ponto-a-ponto e barramento, não conseguem acompanhar a eficiência e escabilidade dos atuais dispositivos eletrônicos (AHMAD; SETHI, 2020).

Figura 2 – Representação de diferentes estruturas de comunicação em SoCs. (a) Barramento de interconexão. (b) Conexão via links ponto-a-ponto. (c) Estrutura de rede-em-chip.



Fonte: Adaptado de Kundu e Chattopadhyay (2015).

Nesse contexto, as NoCs se apresentam como uma solução aprimorada para a implementação de SoCs. As NoCs são redes de comunicação estabelecidas entre múltiplos dispositivos eletrônicos. A implementação de NoCs busca solucionar os desafios na comunicação de dispositivos em sistemas ultra integrados, otimizando variáveis como área de

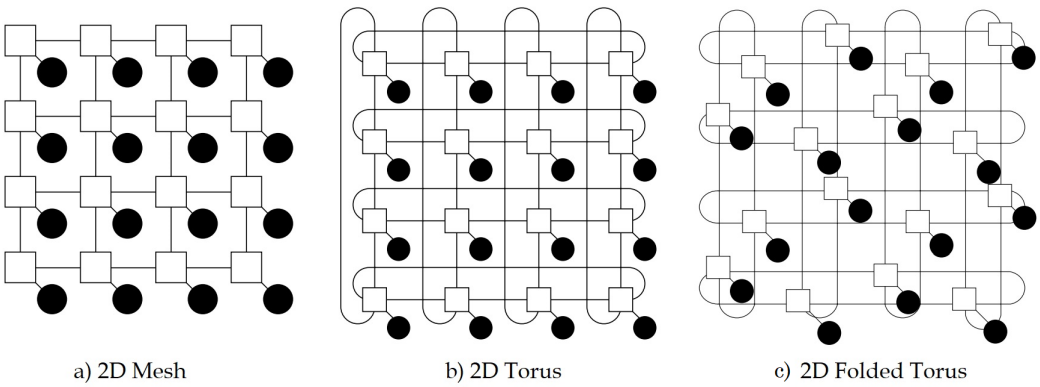
utilização, largura de banda, *delay* de propagação, latência, vazão e consumo de potência (KUNDU; CHATTOPADHYAY, 2015). A Figura 2, representa um comparativo entre NoCs e formas tradicionais de conexão entre dispositivos.

Apesar da promessa em revolucionar a comunicação em SoCs, o desenvolvimento de redes-em-chip representa um desafio em escala multidimensional. Nesse sentido, por serem inspiradas em redes de computadores, o design de NoCs envolve tanto componentes de hardware, quanto de elementos de redes de comunicação, tais como topologias e interfaces de rede, roteadores e algoritmos de roteamento (FÉ, 2017).

2.2.1 Topologias de Rede

A topologia de rede diz respeito ao modo pelo qual os elementos (nós ou nodos) de rede estão interconectados. Essencialmente, a topologia refere-se à forma física (ou geométrica) da rede, definindo a organização dos elementos e a interligação entre os nós via enlaces (FÉ, 2017). Diversas são as topologias de rede que podem ser aplicadas em NoCs, diferindo-se de acordo com sua arquitetura de interconexão. A Figura 3 demonstra três dessas topologias: *Mesh*; *Torus* e *Folded Torus*; na qual os quadrados brancos representam os elementos roteadores, enquanto as circunferências pretas referem-se aos núcleos IP de processamento.

Figura 3 – Topologias de rede para modelos de NoCs: *Mesh*; *Torus* e *Folded Torus*.



Fonte: Adaptado de Kundu e Chattopadhyay (2015).

A escolha adequada da topologia de rede para o design de NoCs deve levar em conta as diversas variáveis intrínsecas da arquitetura do sistema. Nesse sentido, conforme destacado pela investigação de Dally e Towles (2004) (DALLY; TOWLES, 2004), as arquiteturas de interconexão que apresentam menor diâmetro, menor distância nodal, maior número de *links* e uma maior largura de banda de bisseção, devem ser priorizadas na concepção de sistemas que fazem uso de NoCs. Em face disso, na Tabela 1 é apresentada uma comparação entre as arquiteturas de interconexão genéricas ($n \times n$) do tipo *Mesh*, *Torus* e *Folded Torus*.

Tabela 1 – Comparação entre parâmetros para diferentes topologias de rede.

Parâmetros	*Mesh*	*Torus*	*Folded Torus*
Diâmetro	$2n - 2$	$n/2$	$n/2$
Largura de Biseção	n	$2n$	$2n$
Número de Links	$2n^2 - 2n$	$2n^2$	$2n^2$
Número de Roteadores	n^2	n^2	n^2
Grau do roteador	3 a 5	5	5

Fonte: Do autor.

Das variáveis presentes na Tabela 1, o diâmetro de rede refere-se a maior distância ou número de saltos necessários para conectar qualquer par de nós da rede. Em NoCs, o diâmetro de rede é um fator importante a ser avaliado, visto que afeta o desempenho geral do sistema (KUNDU; CHATTOPADHYAY, 2015). Um diâmetro de rede menor implica em menor latência de comunicação e tempos de resposta mais rápidos entre os nós (DALLY; TOWLES, 2004).

Em NoCs, a largura de biseção da rede é determinada pela topologia escolhida e pela quantidade de *links* disponíveis para comunicação. Uma largura de biseção alta indica uma rede robusta, capaz de suportar um maior volume de tráfego simultâneo entre diferentes regiões da rede. Por outro lado, uma largura de biseção baixa pode resultar em gargalos de comunicação e congestionamento. O número de *links* é a quantidade total de conexões físicas existentes em uma rede. Os *links* são os meios pelos quais os diferentes nós na rede podem trocar informações e se comunicar uns com os outros (DALLY; TOWLES, 2004).

O número de roteadores refere-se à quantidade de nós de roteamento presentes em uma NoC (DALLY; TOWLES, 2004), já o grau do roteador é a quantidade de caminhos de comunicação direta que ele pode estabelecer com outros roteadores. A quantidade de roteadores em uma NoC depende da escala e complexidade do sistema em que ela está sendo aplicada. Quanto maior o número de nós ou elementos de processamento que precisam se comunicar na NoC, maior geralmente será o número de roteadores necessários para fornecer uma rota eficiente e rápida entre eles (KUNDU; CHATTOPADHYAY, 2015).

2.2.1.1 Arquitetura Mesh

No paradigma de NoCs, a topologia de rede em configuração *Mesh* é a mais implementada. Segundo Fé (FÉ, 2017), a popularização do uso de redes em arquitetura de interconexão *Mesh* se dá em vista do seu *layout* eficiente. Na topologia *Mesh* cada *core* IP é conectado diretamente a um único roteador, em que cada roteador, com exceção dos roteadores presentes nos cantos e bordas da rede, é conectado a quatro outros roteadores imediatamente dispostos na vizinhança (KUNDU; CHATTOPADHYAY, 2015).

Em uma rede *Mesh*, cada dispositivo pode atuar como um roteador para enviar e receber dados. Essa característica proporciona maior robustez e tolerância a falhas, uma

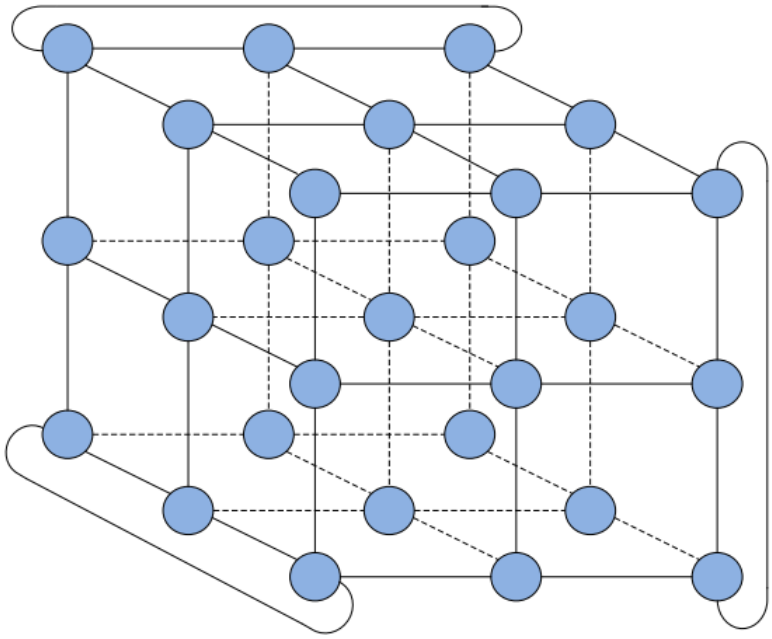
vez que, se um dispositivo falhar, os outros dispositivos ainda podem se comunicar por meio de rotas alternativas. Além disso, a capacidade de comunicação é distribuída entre todos os dispositivos da rede, evitando gargalos em um único ponto central (KUNDU; CHATTOPADHYAY, 2015).

2.2.1.2 Arquitetura Torus

A arquitetura *Torus* (ou rede toroidal) é uma topologia de rede em que os nós estão organizados em forma de grade bidimensional com conexões toroidais, formando um anel fechado. Essa topologia é chamada de "toroidal" porque pode ser visualizada como um toro estendido no espaço. Uma vantagem da arquitetura *Torus* é que ela oferece baixa latência e alta largura de banda de comunicação entre os nós. Além disso, é uma topologia escalável, pois o número de nós pode ser facilmente aumentado sem afetar significativamente o desempenho geral da rede. Isso a torna adequada para sistemas distribuídos e computação de alto desempenho, demandados por NoCs e demais sistemas neuromórficos.

Na arquitetura *Torus*, cada nó é conectado a seus vizinhos adjacentes em todas as direções possíveis. Por exemplo, em uma rede toroidal de dimensão 3 (ou 3D *Torus*), cada nó teria conexões com seus vizinhos acima, abaixo, à esquerda, à direita, à frente e atrás. Isso cria uma estrutura altamente interconectada e permite que os dados sejam transmitidos de forma eficiente em várias direções. A visualização de uma arquitetura *Torus* em escala tridimensional pode ser conferida na Figura 4.

Figura 4 – Arquitetura *Torus* em escala tridimensional.



Fonte: Wang *et al.* (2015).

Apesar de apresentar maior robustez quando comparada à redes em arquitetura *Mesh*, a implementação da arquitetura *Torus* pode ser complexa devido às conexões cruzadas necessárias para criar a topologia toroidal. Nesse caso, é necessário considerar as rotas e o desenvolvimento de algoritmos de roteamento eficientes para evitar congestionamentos e garantir uma comunicação eficaz entre os nós (WANG *et al.*, 2015).

2.2.1.3 Arquitetura Folded Torus

A arquitetura "Torus Dobrada" (*Folded Torus*) é uma variação da arquitetura toroidal, que busca reduzir a latência e melhorar a escalabilidade. Na arquitetura *Folded Torus*, a rede é organizada em forma de grade tridimensional, onde os nós são dispostos em uma estrutura toroidal que é dobrada em si mesma. Essa dobra cria uma sobreposição de caminhos, permitindo maior paralelismo e redundância nos fluxos de dados (KUNDU; CHATTOPADHYAY, 2015).

A principal vantagem da arquitetura *Folded Torus* é a redução da latência de comunicação entre os nós da rede. Ao dobrar a estrutura toroidal, os caminhos entre os nós são encurtados, resultando em tempos de comunicação mais rápidos. Além disso, a sobreposição de caminhos ajuda a distribuir a carga de tráfego de maneira mais eficiente, evitando gargalos e aumentando a escalabilidade do sistema.

Da maneira semelhante à observada na topologia *Torus*, a arquitetura *Folded Torus* também oferece melhorias na tolerância a falhas de transmissão de dados através da rede. Como existem múltiplos caminhos entre os nós, a falha de um caminho ou de um elemento roteador específico não causa interrupção completa na comunicação, já que os dados podem ser roteados por outros caminhos disponíveis (KUNDU; CHATTOPADHYAY, 2015).

No entanto, é importante notar que a implementação da arquitetura *Folded Torus* requer uma topologia física específica e um protocolo de roteamento adequado. Além disso, a complexidade do *design* e a escalabilidade podem ser desafiadoras, especialmente em grandes sistemas.

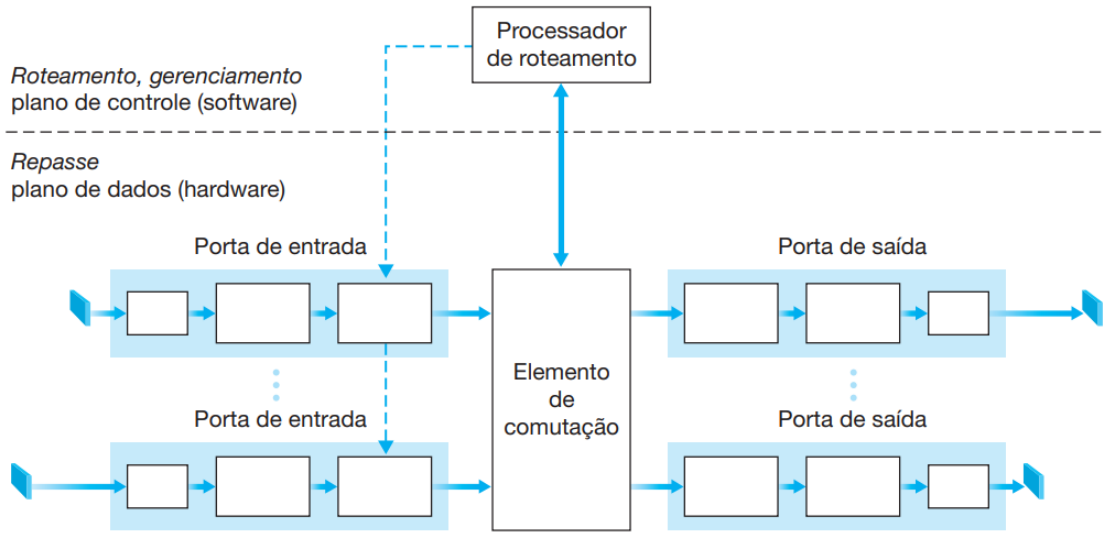
2.2.2 Elemento Roteador

Em aplicações de rede, sistemas finais trocam mensagens fragmentadas em pequenas porções de dados, denominadas pacotes. Entre origem e destino, cada pacote percorre enlaces de comunicação e comutadores de pacotes. Comutadores de pacotes têm por função encaminhar pacotes que chegam em um dos seus enlaces de comunicação de entrada para um dos seus enlaces de comunicação de saída.

Dentre os principais tipos de comutadores de pacotes, destacam-se os roteadores e os comutadores de camada de enlace. Enquanto os comutadores de camada de enlace são aplicados na comutação de pacotes em redes de acesso, os roteadores são utilizados principalmente no *core* das redes (KUROSE; ROSS, 2014). De maneira genérica, a arquitetura de um roteador é composta por quatro componentes, sendo eles: as portas de

entrada, o elemento de comutação, as portas de saída e o processador de roteamento. Uma perspectiva de alto nível da arquitetura genérica de um roteador pode ser visualizada na Figura 5.

Figura 5 – Arquitetura de um elemento roteador genérico



Fonte: Kurose e Ross (2014).

No núcleo das NoCs, os roteadores são fundamentais para a transferência de dados entre emissor e destinatário. Através da conexão de n entradas à n saídas, roteadores permitem a implementação das camadas físicas de transporte de uma NoC. Tendo em vista que a transferência de dados via comutação de pacotes é fundamental em topologias de rede, os elementos roteadores em NoCs influenciam diretamente na velocidade, vazão, área e potência do sistema (FÉ, 2017).

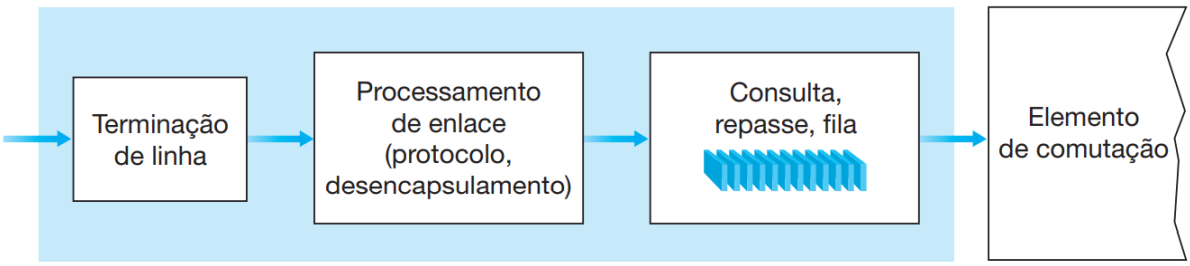
No que se refere à funcionalidade dos elementos de roteamento, as portas de entrada, portas de saída e o elemento de comutação de um roteador executam a função de repasse, sendo em geral implementadas em hardware. A função de repasse, é a transferência dos pacotes dos enlaces de entrada até os enlaces de saída adequados de um roteador. Os elementos de repasse são agrupados pelo plano de repasse do roteador, estrutura conhecida pelo termo clássico *look-up table* (LUT).

Por sua vez, as funções de controle de um roteador (execução de protocolos, funções de gerenciamento e ativação/desativação de enlaces) são realizadas pelo elemento processador de roteamento. Dessa maneira, diferentemente dos demais elementos que compõem o roteador, o processador de roteamento pertence ao plano de controle do roteador.

2.2.2.1 Processamento de entrada

De maneira genérica, as portas de entrada são responsáveis pela extração dos endereços dos dados de entrada e armazenagem temporária dos mesmos. Dessa maneira, o processamento de entrada pertence a camada física, visto que atua na interface responsável pela finalização de um enlace físico à entrada do roteador. Além do mais, as portas de entrada pertencem às camadas de enlace, exercendo a interoperação com as funções de enlace presentes na outra face da entrada. Na Figura 6 é apresentada uma visão geral da lógica de entrada em elementos roteadores.

Figura 6 – Esquematização do processamento de entrada em roteadores.



Fonte: Kurose e Ross (2014).

Além de atuarem como elementos centrais de pré-processamento de pacotes, as portas de entrada têm por característica a implementação da função de exame. A função de exame refere-se ao processo de consulta à tabelas de repasse para determinar para qual porta de saída deve ser repassado um dado pacote que se encontre presente na porta de entrada. Para certificar que os pacotes não serão perdidos ao longo do processo de roteamento, as portas de entrada devem possuir *buffers* de armazenamento temporário de pacotes.

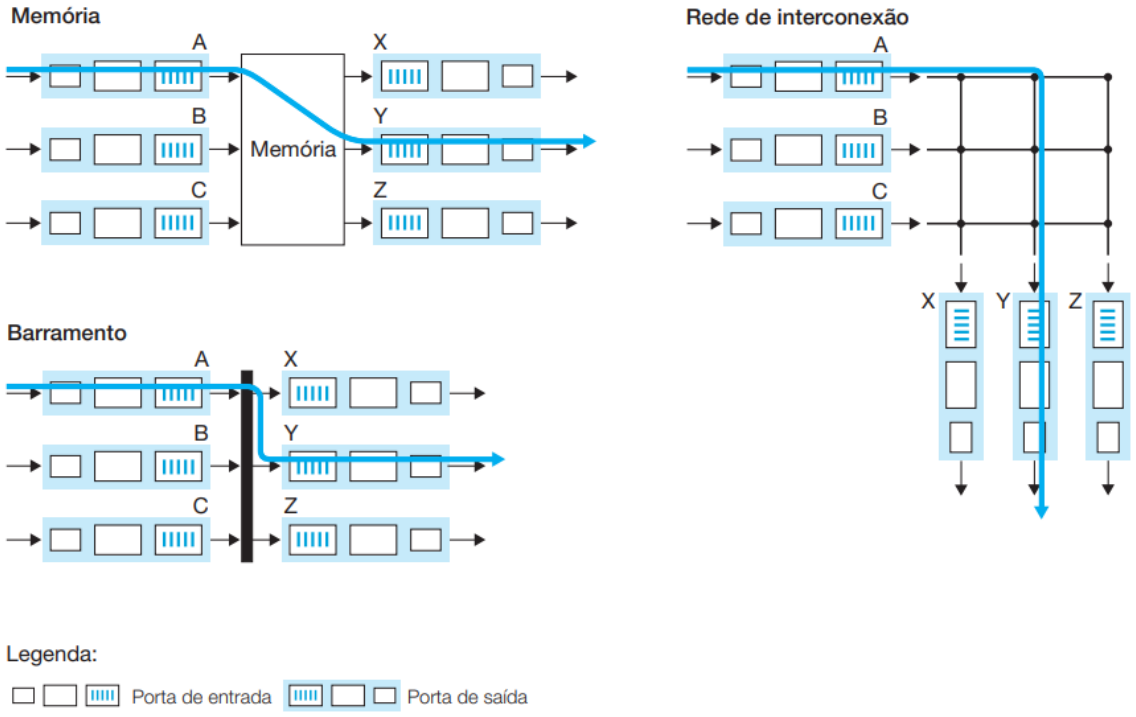
2.2.2.2 Elemento de comutação

O elemento de comutação desempenha um papel central no roteador, pois é responsável por realizar a troca de pacotes de uma porta de entrada para uma porta de saída específica. Diversas são as técnicas de comutação de pacotes aplicáveis ao elemento de comutação, destacando-se a comutação por memória, a comutação por barramento e a comutação por uma rede de interconexão. Na Figura 7 são ilustradas as principais técnicas de comutação enunciadas.

Dentre as principais técnicas de comutação presentes na Figura 7, a comutação por uma rede de interconexão (*crossbar*) é de especial interesse em roteadores aplicados em redes NoCs. Conforme destacado por Kurose e Ross (2014), a técnica de comutação *crossbar* permite superar obstáculos presentes nas outras técnicas de comutação, como é o caso da limitação da largura de banda resultante da comutação por barramento

(KUROSE; ROSS, 2014). Além do mais, as redes do tipo *crossbar* permitem a comutação de diversos pacotes em paralelo, o que justifica o seu uso em aplicações de NoCs integradas ao processamento paralelo proporcionado por redes neurais.

Figura 7 – Técnicas de comutação de pacotes em roteadores.



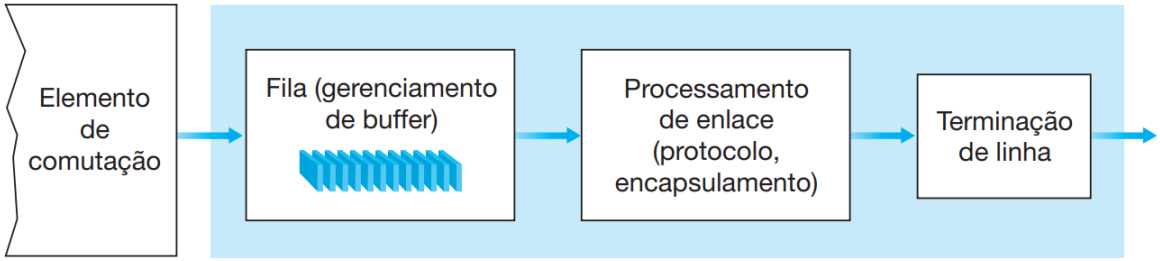
Fonte: Kurose e Ross (2014).

Do ponto de vista lógico, comutadores do tipo *crossbar* consistem em uma rede de $2n$ barramentos conectados a n portas de entrada com n portas de saída. Para viabilizar a correta conexão na relação entrada-saída, comutadores *crossbar* costumam apresentar um elemento demultiplexador (DEMUX), o qual permite a distribuição da entrada de pacotes à saída desejada (FÉ, 2017).

2.2.2.3 Processamento de saída

Após o estágio de comutação, as portas de saída tomam os pacotes armazenados na memória da porta de saída e os transmitem pelo enlace de saída. Portanto, no processamento de saída há a seleção e retirada dos pacotes da fila de dados para transmissão (aplicação das camadas de enlace e física). No processamento de saída, a seleção de pacotes é em geral realizada mediante a presença de um árbitro, o qual é responsável pelo processo de alocação do canal de saída de acordo com as solicitações das várias entradas associadas a este canal (FÉ, 2017). A Figura 8 demonstra a lógica de saída em elementos roteadores.

Figura 8 – Esquematização do processamento de saída em roteadores.



Fonte: Kurose e Ross (2014).

2.3 REDES NEURAIS

As Redes Neurais Artificiais (RNAs), comumente designadas em língua inglesa *Artificial Neural Networks* (ANNs), são algoritmos computacionais fundamentados em estudos lógico-matemáticos sobre a natureza da estrutura neural de organismos inteligentes, sendo caracterizadas por exibir tarefas cognitivas e de aprendizado.

Em vista da aproximação das RNAs com estruturas neurais biológicas, tais algoritmos apresentam grande capacidade de adaptação aos estímulos externos (entradas), resultando em uma aprendizagem gradativa na resolução de uma determinada situação a qual estão submetidas.

2.3.1 Neurônios Biológicos

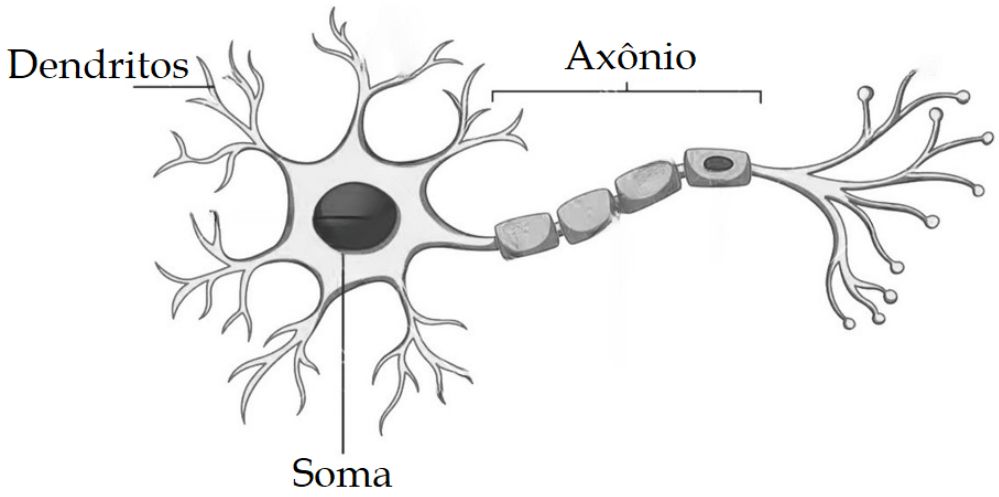
Compondo as bases estruturais e funcionais do sistema nervoso, os neurônios são células altamente especializadas, sendo capazes de receber, transmitir, processar e integrar estímulos. A anatomia de um neurônio típico é composta por três estruturas fundamentais: os dendritos, o soma (corpo celular) e o axônio (YANG *et al.*, 2020). A Figura 9 representa de maneira simplificada a estrutura anatômica de um neurônio biológico.

A capacidade dos neurônios em detectar e responder à estímulos é conhecida como excitabilidade. Estímulos são alterações detectáveis do meio interno ou externo da célula nervosa. Neurônios respondem à estímulos mediante a diferença de potencial elétrico existente entre a superfície interna e externa da membrana celular. Diferenças de potencial elétrico resultam na liberação de neurotransmissores químicos ou impulsos elétricos nas células nervosas, os quais podem restringir-se à célula afetada ou serem propagados entre células nervosas.

A propagação de impulsos nervosos nos neurônios é realizada de maneira unidirecional, no sentido dendrito, soma e axônio, respectivamente. Dessa maneira, os dendritos são responsáveis pela recepção e transmissão de informações em direção ao corpo celular do neurônio (SCHUMAN *et al.*, 2017). O processamento dos neurotransmissores e sinais elétricos provindos dos dendritos acarreta na tomada de decisão por parte do soma, o qual

pode ou não gerar um impulso nervoso (YANG *et al.*, 2020). Impulsos nervosos resultantes da tomada de ação por parte do soma são transmitidos através do axônio em direção a outros neurônios (SCHUMAN *et al.*, 2017).

Figura 9 – Anatomia simplificada de um neurônio biológico.



Fonte: Adaptado de Bond (2022).

A transmissão dos impulsos nervosos entre neurônios é a forma pela qual a célula transmite informações acerca do estímulo recebido. A propagação de informações via neurotransmissores ou impulsos elétricos na relação neurônio-neurônio é regida por sinapses. Sinapses referem-se às junções entre as terminações nervosas do axônio de um neurônio e a membrana dos dendritos de outro neurônio, o que possibilita a propagação física de sinais entre dois neurônios. De acordo com Yang *et al.* (2020), o conceito de sinapse foi introduzido pioneiramente nos estudos de Sherrington (1897), sendo utilizado para descrever a conexão funcional entre células nervosas pertencentes ao sistema nervoso central.

2.3.2 Redes Neurais Artificiais

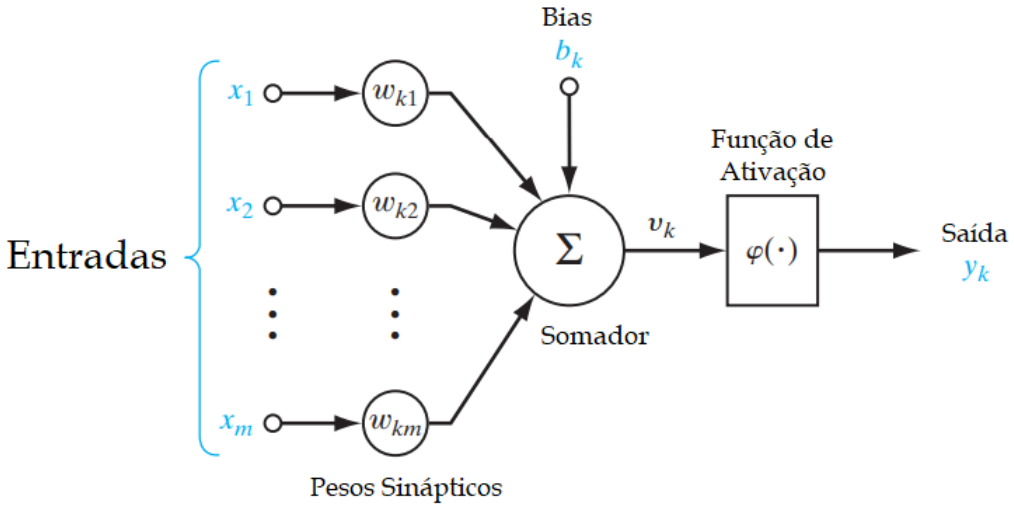
Nas redes neurais artificiais, os neurônios computacionais são as unidades responsáveis pelo processamento das informações. RNAs podem ser compostas por centenas ou até milhares de neurônios interconectados. De acordo com as investigações de Segundo (2015), Wuerges e Borba (2010), o trabalho pioneiro relacionado à concepção de neurônios computacionais foi introduzido por Warren McCulloch e Walter Pitts, sob o título *A Logical Calculus of the Ideas Immanent in Nervous Activity*, sendo publicado em 1943. A abordagem de McCulloch e Pitts teve como inspiração a fisiologia dos neurônios cerebrais humanos, sendo fundamentada pela lógica proposicional de Russell e Whitehead e pela teoria computacional de Turing (PACHECO; PEREIRA, 2018).

Haja vista que a lógica proposicional baseia-se na avaliação e classificação de afirmações entre sentenças verdadeiras ou falsas, os neurônios artificiais de McCulloch e Pitts assumem apenas estados lógicos booleanos. A comutação do estado falso (0) para o estado verdadeiro (1) de um dado neurônio se dá em resposta à estimulação por um número suficiente de neurônios vizinhos (RUSSELL; NORVIG, 2013).

Nesse sentido, a excitabilidade neuronal é modelada através da atribuição de pesos sinápticos, na qual valores maiores referem-se à maior excitação do neurônio em face ao estímulo recebido. Apesar da relativa simplicidade na lógica proposicional dos estados neuronais, o estudo de McCulloch e Pitts demonstrou ser possível o cálculo de funções computáveis através de conectivos lógicos em estrutura de rede, fundamentando-se as bases para a modelagem de neurônios artificiais.

Segundo Haykin (2009), um neurônio artificial pode ser modelado a partir de três elementos básicos: um conjunto de sinapses ou elos de conexão, um somador e, uma função de ativação. Na Figura 10 é apresentado um neurônio computacional com os seus elementos fundamentais em destaque.

Figura 10 – Modelo matemático não linear de um neurônio computacional.



Fonte: Adaptado de Haykin (2009).

No conjunto de sinapses do modelo não linear de um neurônio, conforme apresentado na Figura 10, um sinal de x_j na entrada da sinapse j é conectado ao neurônio k, sendo multiplicado pelo peso sináptico w_{kj}. Na representação do peso sináptico, o primeiro índice refere-se ao neurônio em questão, enquanto o segundo trata-se do terminal de entrada a qual o peso se refere. Dentre os vários estímulos recebidos a partir das entradas, considera-se que certos estímulos proverão uma excitação maior ou menor do neurônio receptor, na qual a atribuição dos pesos sinápticos consiste em ponderar o grau de excitação neuronal.

Por sua vez, o elemento somador (combinador linear) tem por função realizar a somatória dos sinais de entrada, os quais foram ponderados pelas sinapses no estágio anterior. Com o intuito de limitar o intervalo de amplitude do sinal de saída do neurônio para convergência a um valor finito, utiliza-se uma função de ativação, também designada por função restritiva, a qual tipicamente limita o sinal de saída ao intervalo unitário fechado $[0, 1]$.

Dessa maneira, em face dos argumentos supra definidos, para um neurônio k a saída do combinador linear u_k devido aos sinais de entrada, é fornecida pela seguinte expressão:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j, \tag{1}$$

na qual, $x_1, x_2, \ldots, x_m$ são os sinais de entrada e $w_{k1}, w_{k2}, \ldots, w_{im}$ são os pesos sinápticos referentes ao neurônio k. Com o intuito de manipular o aumento ou diminuição da entrada líquida da função de ativação, aplica-se um parâmetro bias b_k. A presença do bias implementa uma transformação afim à saída u_k do combinador linear. Dessa maneira, considerando o efeito do bias, bem como representando a função de ativação por $\varphi(\cdot)$, o sinal de saída y_k do neurônio é dado por:

$$y_k = \varphi(u_k + b_k). \tag{2}$$

Portanto, as equações (1) e (2) definem o modelo matemático de um neurônio k. Tendo em vista que o parâmetro de bias b_k é externo ao neurônio, é possível reformular o modelo concebido considerando-se uma nova sinapse referente a b_k. Portanto, ao definir uma nova entrada referente ao bias b_k, como $x_0 = +1$, com peso sináptico w_{k0}, torna-se possível reescrever as equações (1) e (2), respectivamente, como:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{3}$$

$$y_k = \varphi(v_k) \tag{4}$$

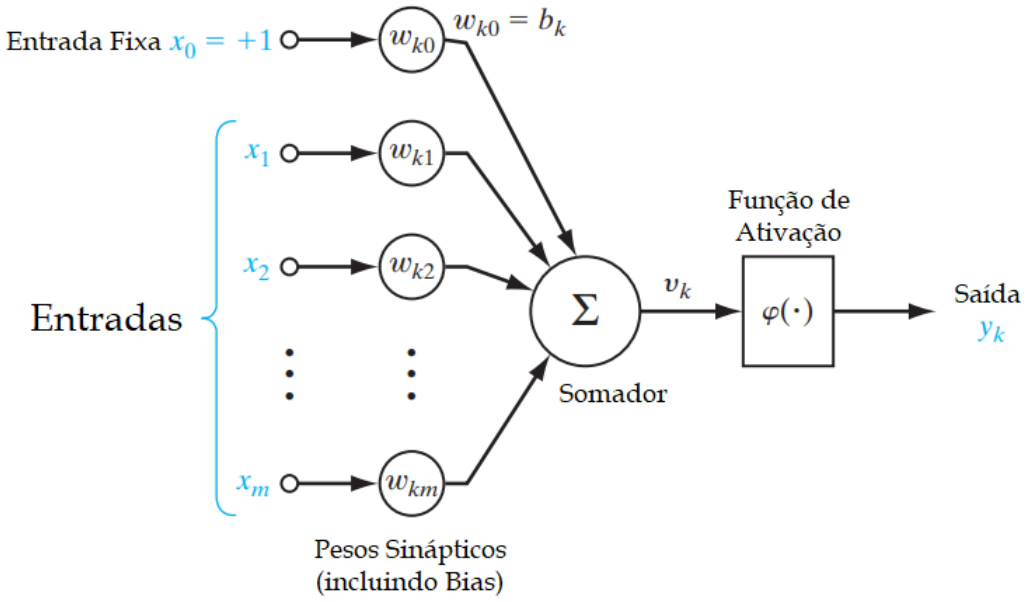
em que,

$$v_k = u_k + b_k. \tag{5}$$

A concepção das equações (3), (4) e (5) afere uma nova definição matemática para o neurônio k, entretanto tal concepção é análoga ao modelo definido pelas equações (1)

e (2). Entretanto, no modelo definido pelas equações (3), (4) e (5), o efeito do bias deve ter como relevância dois fatores: a adição de um novo sinal de entrada fixo definido por $x_0 = +1$, bem como a inserção de um novo peso sináptico w_{ko}, o qual deve ser por regra igual ao bias b_k. Para fins de melhor visualização do novo modelo, a Figura 11 demonstra a estrutura lógica do neurônio k apresentando o bias como entrada x_0 e com peso sináptico w_{ko}.

Figura 11 – Modelo matemático não linear alternativo de um neurônio computacional.



Fonte: Adaptado de Haykin (2009).

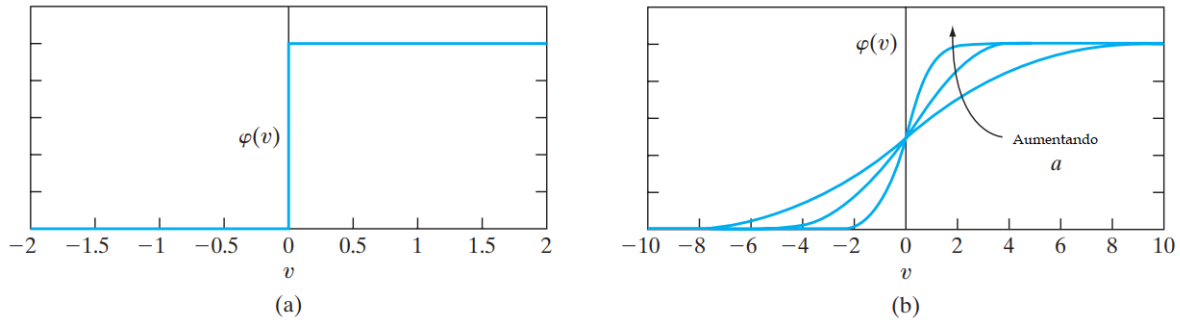
2.3.2.1 Funções de Ativação

Dentre os elementos básicos que constituem o modelo do neurônio computacional, a função de ativação $\varphi(v)$ é o elemento chave para a definição da saída a partir das informações apresentadas na entrada. Dessa maneira, conforme demonstrado por Haykin (2009), em aplicações que utilizam redes neurais artificiais, são tipicamente implementados uma das seguintes funções de ativação: Função de Limiar; Função Sigmoide e; Função ReLU.

2.3.2.1.1 Função limiar

A função limiar, também normalmente referida por função de Heaviside ou simplesmente função degrau unitário, é uma função singular que assume valor nulo quando seu argumento é negativo e valor unitário para argumentos positivos. No contexto das redes neurais, a saída y_k do neurônio k que emprega a função limiar é definida por:

Figura 12 – Dinâmica de funções de ativação para redes neurais: (a) Função Limiar; (b) Função Sigmoide.



Fonte: Adaptado de Haykin (2009).

$$y_k = \begin{cases} 1, & \text{se } v_k \geq 0 \\ 0, & \text{se } v_k < 0 \end{cases} \tag{6}$$

em que v_k é o campo local induzido do neurônio, ou seja, o potencial de ativação v_k do neurônio k o qual pode ser expresso por:

$$v_k = \sum_{j=1}^{k} w_{kj} x_j + b_k. \tag{7}$$

Note que, em face do comportamento exibido pela saída definida pela eq. (6), a função de ativação liminar faz com que o neurônio k comporte-se exatamente conforme foi proposto por McCulloch e Pitts (1943). Portanto, como reconhecimento ao trabalho pioneiro em redes neurais, a literatura se refere ao par de equações (6) e (7) como *modelo de McCulloch-Pitts*, com a saída y_k exibindo a propriedade de *tudo-ou-nada*, a qual é típica do modelo.

2.3.2.1.2 Função sigmoide

Apesar da facilidade de implementação de redes neurais utilizando-se da função de ativação limiar, a sua limitação de saída entre valores booleanos motiva a busca por funções de ativação alternativas. A função sigmoide exibe um comportamento em S, apresentando características desejáveis de crescimento e de balanceamento dos comportamentos linear e não linear. Além do mais, por dispor de parâmetros que variam entre o intervalo unitário fechado $[0, 1]$, é facilmente implementada no *design* de RNAs.

De acordo com a investigação de Oliveira Florentino, Fátima Vilela Biscaro e Souza Passos (2010), as funções sigmoidais mais comumente utilizadas são: Logística, Gompertz,

Morgan-Mercer-Flodin e Weibull (OLIVEIRA FLORENTINO; FÁTIMA VILELA BISCARO; SOUZA PASSOS, 2010). Entretanto, no contexto das redes neurais artificiais é comum a aplicação da função logística como função de ativação de neurônios. Dessa maneira, a função logística pode ser definida por:

$$\varphi(v) = \frac{1}{1 + \exp(-av)} \tag{8}$$

na qual, a é o parâmetro de inclinação da função sigmoide. Por sua vez, a derivada da função sigmoide é fornecida por:

$$\frac{\partial \varphi(v)}{\partial v} = \varphi(v) \cdot (1 - \varphi(v)) \tag{9}$$

A possibilidade de obtenção de diferentes inclinações a, resulta na obtenção de diferentes curvas para a função de ativação do neurônio k. O comportamento da função de ativação sigmoide pode ser observado no item b) da Figura 12.

2.3.2.1.3 Função ReLU

A função de ativação ReLU (*rectifed linear unit* - unidade linear retificada) é atualmente a função de ativação mais implementada em arquiteturas de redes neurais, como redes convolucionais (CNNs) e redes neurais profundas (DNNs), se mostrando eficaz em muitas aplicações de aprendizado de máquina e visão computacional. A principal vantagem da função de ativação ReLU é sua simplicidade e eficiência computacional. A característica de não linearidade da função ReLU viabiliza o seu uso em aplicações de *deep learning*, visto que a mesma possibilita o uso do algoritmo de *backpropagation*. As expressões para a função ReLU e sua derivada são dadas por:

$$ReLU(x) = \max\{0, x\} \qquad \frac{\partial ReLU(x)}{\partial x} = \begin{cases} 1, & \text{se } x \geq 0 \\ 0, & \text{se } x < 0 \end{cases} \tag{10}$$

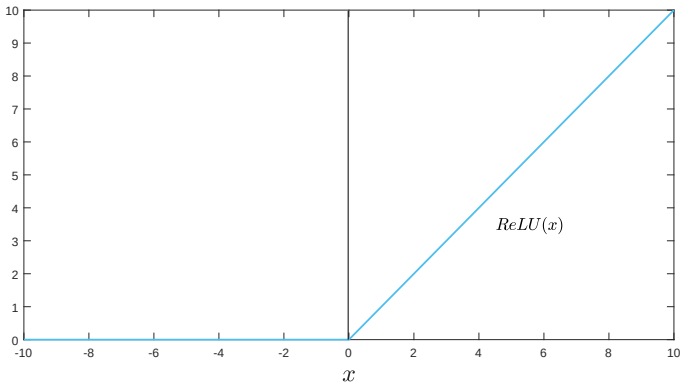
A partir das expressões em (10), observa-se que para uma determinada entrada negativa, a resposta do neurônio é nula, portanto não há ativação do neurônio, ou seja, a função ReLU retorna o valor de entrada se for maior que zero e valor nulo caso contrário. Simplificando, a função "liga" o neurônio se a entrada for positiva e o "desliga" se a entrada for negativa. Portanto, tal característica resulta na possibilidade de direcionar a ativação de neurônios na rede, tendo por consequência uma rede neural eficiente e de fácil computação. O comportamento da função ReLU pode ser observado na Figura 13.

Um dos maiores desafios no treinamento de redes neurais através de algoritmos de *deep learning* consiste na dissipação do gradiente (*vanishing gradient*). O gradiente de uma

função de ativação f, denotado por ∇f, consiste na coleção de todas as derivadas parciais em um vetor. A dissipação do gradiente refere-se ao aumento exponencial da potência do erro propagado em função da distância da camada mais distante em redes FFNs (*feed-forward networks*) (SUSSILLO; ABBOTT, 2015). A sua presença é altamente indesejada para a escalabilidade de redes neurais, pois resulta na ineficiência de performance na adição de novas k camadas de neurônios.

Dessa maneira, comparando-se a expressão para a derivada da função de ativação sigmoidal, fornecida pela eq. (9), juntamente com a derivada da função de ativação ReLU (eq. (6)), observa-se que enquanto o valor máximo da eq. (9) é obtido quando $\varphi(v) = 0,5$, resultando em $\varphi(v)' = 0,25$, o valor máximo da derivada da função de ativação ReLU resulta em $ReLU(x)' = 1$. Portanto, a aplicação da função sigmoide incorre na diminuição sucessiva do gradiente no treinamento dos neurônios através da rede, visto que o mesmo será multiplicado sucessivamente por valores iguais ou inferiores a $\varphi(v)' = 0,25$. Tal característica não acontece com a função ReLU, o que possibilita o treinamento de redes neurais em *deep learning* na ausência da dissipação do gradiente (SUSSILLO; ABBOTT, 2015).

Figura 13 – Comportamento da função de ativação ReLU.



Fonte: Do autor.

2.3.3 Aprendizado de Máquina

O Aprendizado de Máquina, também conhecido como *Machine Learning*, é um ramo da inteligência artificial que se concentra no desenvolvimento de algoritmos e técnicas que permitem que os computadores aprendam e tomem decisões baseadas em dados, sem serem explicitamente programados para realizar tarefas específicas (WUERGES; BORBA, 2010; HAYKIN, 2009).

Em vez de seguir instruções precisas, os algoritmos de Aprendizado de Máquina são projetados para aprender com exemplos e experiências passadas, identificando padrões

e relações nos dados. Esses algoritmos são alimentados com um conjunto de dados de treinamento, que consiste em entradas (dados de entrada) e saídas correspondentes (rótulos ou respostas corretas). O algoritmo analisa esses dados de treinamento para criar um modelo estatístico ou uma função matemática que relaciona as entradas às saídas desejadas (SEGUNDO, 2015).

Uma vez treinado, o modelo de *Machine Learning* pode ser usado para fazer previsões ou tomar decisões sobre novos dados de entrada. Ele é capaz de generalizar a partir dos padrões aprendidos durante o treinamento para realizar tarefas, como classificação, regressão, agrupamento, detecção de anomalias e muitas outras, dependendo do problema em questão.

Existem diferentes abordagens e técnicas de Aprendizado de Máquina, incluindo aprendizado supervisionado, aprendizado não supervisionado, aprendizado por reforço e aprendizado profundo, o qual é mais conhecido pelo termo *deep learning*. Cada abordagem tem seus próprios algoritmos e métodos específicos, mas todos compartilham o objetivo comum de permitir que os computadores aprendam a partir dos dados de forma automatizada (RAMOS, 2003).

Os estudos de Rumelhart *et al.* (1986) demostraram que é possível o treinamento de redes neurais utilizando-se de um modelo com diversas camadas de neurônios (RUMELHART *et al.*, 1986). Conforme destacado pela investigação de RAMOS (2003), atualmente as redes perceptron de múltiplas camadas (*Multilayer Perceptron Neural Network* - MLP), treinadas com o algoritmo retro propagação ou *backpropagation*, se apresentam como o principal modelo de treinamento de redes neurais (RAMOS, 2003).

2.3.4 Categorização de dados Iris de Fisher

A classificação do conjunto de dados *Iris* de Fisher é um dos problemas clássicos de inteligência artificial. Introduzido pelo estudo do estatístico e biológo britânico Ronald Aylmer Fisher (FISHER, 1936), *The use of multiple measurements in taxonomic problems* (O uso de múltiplas medições em problemas taxonômicos), o conjunto de dados *Iris* de Fisher compreende uma coleção de informações tabuladas, contendo uma série de características utilizadas na classificação taxonômica de três espécies de flores do gênero *Iris*: *I. setosa*, *I. versicolor* e, *I. virginica*. A Figura 14 representa as três espécies do subconjunto de flores do gênero *Iris*.

Distribuídas entre 150 amostras, sendo 50 amostras para cada uma das 3 espécies, o conjunto de dados *Iris* de Fisher apresenta informações multivariadas referentes às características anatômicas das flores amostradas. Cada amostra é descrita por quatro características ou atributos: comprimento da sépala, largura da sépala, comprimento da pétala e largura da pétala. O anexo B apresenta o conjunto de dados em sua íntegra.

O uso de técnicas de aprendizado de máquina via implementação de redes neurais, fornece um meio ótimo para a categorização das espécies e o agrupamento dos dados,

sobretudo em vista da simplicidade, tamanho razoável e características bem definidas do conjunto de dados. Logo, o objetivo do uso do conjunto de dados no contexto de redes neurais, consiste no desenvolvimento de algoritmos que possam classificar corretamente as amostras de flores em suas respectivas espécies com base nas características individuais de cada flor.

Figura 14 – Compilado de registros fotográficos destacando as estruturas de pétalas e sépalas do subconjunto de flores do gênero *Iris*: a) *I. setosa*, b) *I. versicolor* e, c) *I. virginica* .

Fonte: Adaptado de Gardenia (2023).

A grande similaridade entre as características visíveis das flores das espécies *I. setosa*, *I. versicolor* e *I. virginica* representa um grande desafio para sua identificação individual, resultando em um problema de classificação multi classe. Dessa maneira, por se tratar de um problema clássico de *Machine Learning*, a aplicação do conjunto de dados *Iris* de Fisher no contexto de processamento computacional neuromórfico em nanoescala é uma forma efetiva para avaliar a eficiência e a eficácia dos sistemas implementados.

2.4 TECNOLOGIA MOS

O desenvolvimento do transistor MOSFET (*Metal-Oxide-Semiconductor field effect transistor*), ou simplesmente transistor MOS, representa um marco na tecnologia microeletrônica. Os transistores MOS de canal P (PMOS), foram os primeiros dispositivos em tecnologia MOS a serem fabricados em circuitos LSI. Com o avanço da tecnologia MOS, foram desenvolvidos os transistores MOS de canal N (NMOS), os quais devido as suas características de mobilidade e performance rapidamente superaram os modelos PMOS (JAEGER; BLALOCK, 2015).

A combinação de transistores PMOS e NMOS em circuitos integrados deu origem a tecnologia CMOS (*Complementary MOS*). Tal combinação resultou em uma maior performance dos circuitos e na melhora dos processos de manufatura, culminando na atual predominância da tecnologia CMOS na indústria microeletrônica (WAKERLY, 2018).

A tecnologia CMOS é amplamente conhecida por sua capacidade de integração em circuitos eletrônicos e sua eficiência energética. O CMOS permite a integração de um grande número de componentes em um único chip, o que resulta em sistemas complexos e funcionais em um espaço reduzido. A miniaturização dos transistores CMOS e a redução do tamanho dos dispositivos permitem a fabricação de circuitos integrados com alta densidade de componentes (JAEGER; BLALOCK, 2015).

Os transistores CMOS têm uma dissipação de potência muito baixa quando estão em estado de repouso, o que os torna altamente eficientes em termos de consumo de energia, característica fundamental no *design* de NoCs (KUNDU; CHATTOPADHYAY, 2015). Além da capacidade de integração e do baixo consumo de energia, a tecnologia CMOS também oferece outras vantagens, como alta imunidade a ruídos, operação em baixa tensão, robustez e compatibilidade com processos de fabricação em grande escala.

2.4.1 O Transistor MOS

Transistores MOS são estruturalmente construídos utilizando-se elementos semicondutores. O silício é atualmente o elemento semicondutor mais utilizado na concepção de transistores, entretanto elementos como germânio e ligas como arsenieto de gálio, carbeto de silício (SiC) e nitreto de gálio, também são empregadas na manufatura de tais dispositivos.

O silício de alto grau de pureza (hiperpuro) é a designação dada ao silício utilizado na indústria microeletrônica para a manufatura de transistores. O silício hiperpuro ou silício policristalino, é obtido pela decomposição térmica e recristalização do triclorosilano ultrapuro (HSiCl3), o que resulta em semicondutor que apresenta uma fração de impurezas de uma parte por bilhão ou menos.

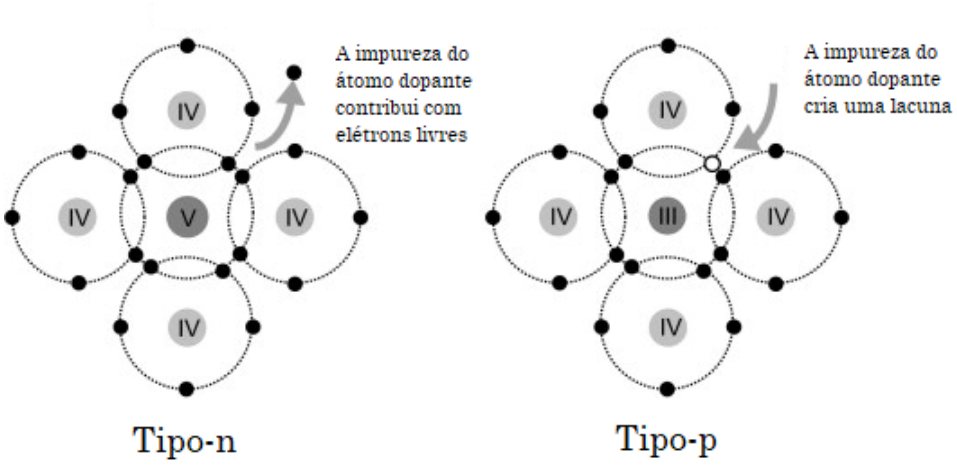
Materiais semicondutores de silício hiperpuro apresentam poucos elétrons livres, culminando em uma baixa condução elétrica. Todavia, visando-se o aumento da condução elétrica, alterações nas propriedades elétricas de semicondutores de silício podem ser obtidas através da adição controlada de impurezas químicas na estrutura cristalina do material, em um processo conhecido por dopagem (DUARTE, 2018).

No processo de dopagem, a adição de impurezas como fósforo e arsênico ao silício ou outro elemento semicondutor, resulta em semicondutores do tipo N. A combinação das impurezas do tipo N ao silício acarreta em ligações covalentes entre os elementos, preenchendo lacunas e aumentando a quantidade de elétrons livres. A designação N (*negative*) do semicondutor se dá em vista da maior negatividade induzida no mesmo, a qual ocorre mediante a presença de uma quantidade maior de elétrons livres circulando através do material.

De maneira semelhante, a submissão do silício à impurezas como gálio e boro, resulta em semicondutores do tipo P. Entretanto, de maneira oposta ao que ocorre em semicondutores do tipo N, a presença de impurezas do tipo P resulta em um aumento na

quantidade de lacunas no elemento semicondutor, aumentando a positividade do mesmo mediante a diminuição de elétrons livres. Portanto, a designação P (*positive*) é resultado de uma quantidade maior de cargas positivas presentes no material (JAEGER; BLALOCK, 2015). A Figura 15 demonstra o efeito da adição de impurezas em semicondutores do tipo N e do tipo P.

Figura 15 – Efeito da dopagem em semicondutores do tipo N e do tipo P.



Fonte: Adaptado de Chen *et al.* (2009).

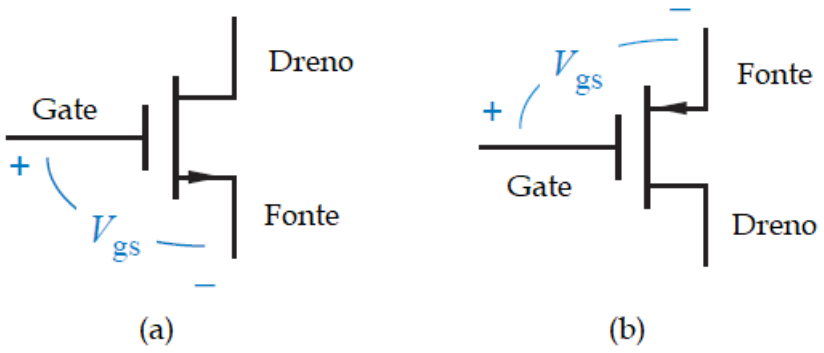
Sendo assim, é através do processo de dopagem que ocorre a diferenciação do material semicondutor que possibilita a concepção de transistores MOS, os quais podem ser classificados como sendo do tipo NMOS ou PMOS. Dessa maneira, os transistores MOS manufaturados com semicondutores do tipo N resultam em transistores NMOS, da mesma forma que transistores MOS manufaturados com semicondutores do tipo P acarretam na concepção de transistores PMOS.

Do ponto de vista físico, os transistores CMOS são compostos por 3 terminais, os quais são designados como *gate* - ***G***, *fonte* (*source* - ***S***) e *dreno* (*drain* - ***D***). A obra de Jaeger e Blalock (2015) (JAEGER; BLALOCK, 2015) destaca a presença de um quarto terminal, o terminal de corpo ou substrato (*body* - ***B***). De maneira genérica, transistores do tipo N são formados por um substrato do tipo P e regiões de fonte e dreno do tipo N.

Por sua vez, transistores do tipo P são complementares aos transistores do tipo N, sendo formados por substrato do tipo N e regiões de fonte e dreno do tipo P. Portanto, nota-se complementariedade entre transistores NMOS e PMOS, os quais, quando implementados em conjunto, definem a tecnologia CMOS. A Figura 16 ilustra a representação de transistores NMOS e PMOS destacando-se seus terminais e a tensão *gate-source* V_{gs}.

Em transistores NMOS, a tensão V_{gs} determina a corrente que passa entre os terminais de dreno e fonte I_{ds}. Nesse sentido, se $V_{gs} = 0$ V, tem-se uma resistência entre dreno e fonte $R_{ds} \approx 6$ MΩ, acarretando em $I_{ds} = 0$ A. Por sua vez, elevando-se o valor da

Figura 16 – Representação de transistores em tecnologia CMOS: (a) transistor MOSFET de canal N (NMOS); (b) transistor MOSFET de canal P (PMOS).



Fonte: Adaptado de Wakerly (2018).

tensão V_{gs} há uma diminuição da região de depleção do transistor NMOS, o que acarreta em uma diminuição de R_{ds} para valores próximos de 0, aumentando-se I_{ds}.

No que se refere ao funcionamento de transistores PMOS, estes apresentam comportamento muito semelhante ao de seus pares NMOS. Todavia, enquanto nos transistores NMOS o terminal de dreno encontra-se normalmente sob uma tensão maior que a tensão de fonte, nos transistores PMOS a tensão da fonte é geralmente maior que a tensão do dreno, acarretando em um fluxo de corrente entre fonte-dreno I_{sd}. De maneira análoga ao caso NMOS, $V_{gs} = 0$ V acarreta em uma resistência entre dreno e fonte R_{ds} de impedância elevada. Entretanto, para este caso, a diminuição da região de depleção do transistor ocorre mediante a diminuição da tensão V_{gs}, resultando em uma diminuição de R_{ds} para valores próximos de 0, aumentando-se I_{sd}.

Conforme pode ser aferido pela simbologia de transistores CMOS, não há conexão elétrica entre os terminais de gate e os demais terminais do transistor. Tal característica é resultante da presença de um material isolante entre o terminal de gate e os terminais de fonte e dreno. A elevada impedância do terminal de gate em transistores CMOS resulta em uma corrente I_g quase nula. A corrente I_g é da ordem de 10^{-6} A (10 μA), sendo comumente tratada como uma corrente de fuga (*leakage current*) (WAKERLY, 2018).

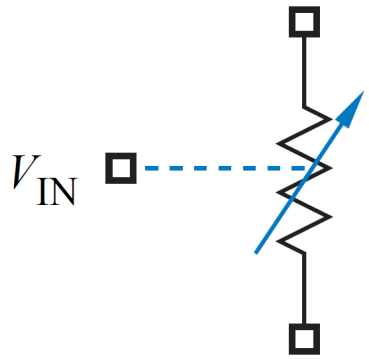
2.4.2 Arquitetura Digital CMOS

Circuitos CMOS são amplamente utilizados para a representação de lógica binária, a qual é fundamental para a implementação de circuitos digitais, desde portas lógicas simples a dispositivos eletrônicos complexos. Apesar da lógica binária apresentar apenas os estados binários 0 ou 1, circuitos CMOS podem ser projetados para exibir saídas lógicas a partir de sinais de tensão e corrente não binários. Nesse sentido, define-se intervalos de tensão ou corrente em que a resposta será interpretada como 0 ou 1.

Conforme destacado por Wakerly (2018), circuitos CMOS geralmente são alimentados a partir de uma fonte de 5 V. Nesse sentido, definindo-se um intervalo de Vcc entre 0 V e 1, 5 V como estado lógico *off* (0) e um intervalo de Vcc entre 3, 5 V e 5 V como estado lógico *off* (1), pode-se aferir que se a tensão de alimentação Vcc determina dois estados do circuito: *on* ou *off* (*high* ou *low*, respectivamente). É factível destacar que, para uma classe de tensão Vcc fora do intervalo definido, tem-se uma indefinição do estado lógico do sistema, o que permite que o mesmo assuma aleatoriamente o valor de 0 ou 1.

Em face da definição lógica acima, bem como considerando-se o funcionamento dos transistores NMOS e CMOS explicitado na seção anterior, é possível aferir que transistores CMOS podem ser modelados como uma resistência variável conectada a 3 terminais. Nesse sentido, conforme pode ser averiguado na Figura 17, a aplicação de uma tensão de entrada sobre um terminal possibilita o controle da resistência presente entre os demais terminais.

Figura 17 – Modelagem de transistores MOS via representação resistiva.



Fonte: Wakerly (2018).

Dessa maneira, considerando-se novamente o modelo NMOS, bem como o modelo de resistência variável, a aplicação de uma tensão $V_{in} = V_{gs}$ suficientemente alta (tipicamente $V_{in} > 0,7$ V, estado lógico $V_{in} = 1$) resulta na formação do canal e diminuição da região de depleção, o que possibilita a passagem de corrente entre dreno-fonte, acarretando em $I_{ds} >> 0$ A, o que permite definir o estado lógico $I_{ds} = 1$. Da mesma forma, ao definir $V_{in} = 0$ V, a tensão V_{in} assume o estado lógico $V_{in} = 0$, o que tem por consequência o bloqueio da corrente, portanto $I_{ds} = 0$ A, o que afere estado lógico $I_{ds} = 0$. Tal analogia pode ser sintetizada na tabela verdade definida abaixo:

Tabela 2 – Comparação entre parâmetros para diferentes topologias de rede.

V_{in}	V_{in} lógico	I_{ds} lógico	Resultado
$V_{in} = 0$ V	0	0	Transistor conduz corrente (fechado)
$V_{in} > 0,7$ V	1	1	Transistor não conduz corrente (aberto)

Fonte: Do autor.

Da analogia acima, observa-se que, de maneira simplificada, um dado transistor CMOS Q pode ser definido por 2 estados: conduz ($Q = 1$) ou não conduz corrente ($Q = 0$). Tais estados são definidos pela tensão V_{in} aplicada no terminal gate do transistor. No caso dos transistores NMOS, considera-se que o transistor esteja sob situações normais de operação quando a tensão de dreno V_d é superior à tensão de fonte V_s, isto é, $V_d > V_s$.

3 METODOLOGIA

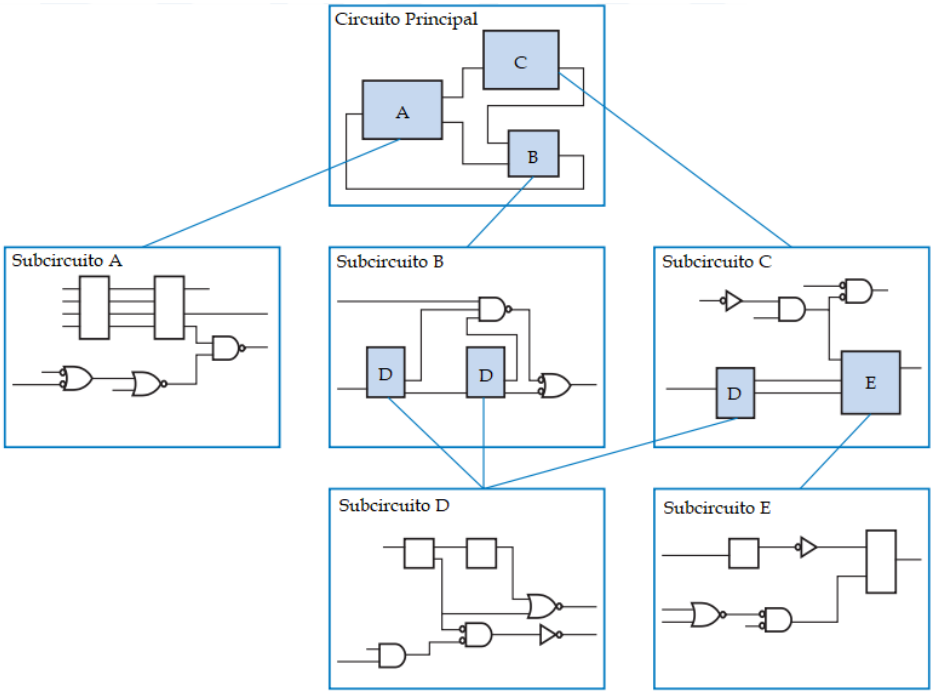
O presente capítulo discorre sobre a metodologia implementada para a idealização do sistema proposto. Nesse sentido, são discutidos os métodos para o desenvolvimento da rede neural proposta e seus subcircuitos, bem como os critérios de avaliação dos circuitos implementados.

3.1 PROCEDIMENTOS METODOLÓGICOS E CRITÉRIOS DE AVALIAÇÃO

A presente investigação foi conduzida partindo-se de ampla pesquisa bibliográfica. Dessa maneira, em vista que a mesma explora tópicos emergentes no campo acadêmico, tais como os estados da arte de NoCs e sistemas nanoeletrônicos bioinspirados, necessitou-se de uma profunda investigação conceitual.

Haja vista a complexidade da rede NoC proposta, o método de projeto utilizada para a concepção dos circuitos digitais consistiu no uso da metodologia hierárquica. Tal metodologia tem como característica a aplicação da técnica "dividir para conquistar" (algoritmo *divide and conquer*), na qual módulos ou circuitos de maior complexidade são divididos em dois ou mais blocos internos. A técnica de divisão dos circuitos em blocos menores pode ser aplicada de maneira recursiva, o que resulta na formação de uma hierarquia de circuitos, conforme pode ser visualizado na Figura 18.

Figura 18 – Aplicação de metodologia hierárquica para simplificação de circuitos.



Fonte: Adaptado de Wakerly (2018).

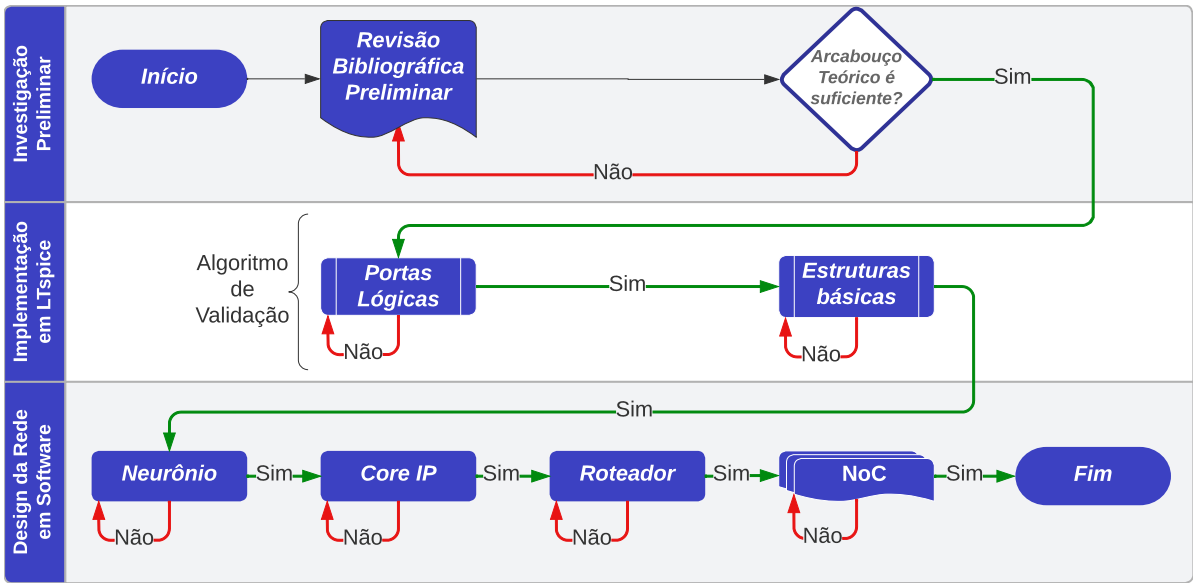
Para o desenvolvimento do sistema de maneira hierárquica, implementou-se a estratégia *bottom-up*. A estratégia *bottom-up* consiste em inicialmente projetar os módulos do nível mais baixo da hierarquia do sistema, os quais servem como base para a implementação das demais etapas de desenvolvimento da rede. Nesse sentido, a integração dos módulos de base possibilita a criação de componentes mais complexos, os quais vão sucessivamente compondo os níveis imediatamente superiores da hierarquia até chegar ao modelo final do sistema.

Para a implementação da rede neural em um modelo de NoC, adotou-se o seguinte passo a passo:

1. **Portas lógicas** são implementadas a partir de transistores CMOS de 16nm;
2. Portas lógicas são agregadas para formar **dispositivos eletrônicos** básicos, como multiplexadores;
3. Dispositivos eletrônicos são combinados para a concepção de um **neurônio**;
4. Neurônios são integrados para formar um ***core* IP**;
5. Um **roteador** é implementado a partir das especificações do *core* IP e;
6. Um **modelo de NoC** em topologia *Torus* é criado a partir da integração de roteadores e *cores* IPs.

Dessa maneira, a fim de exemplificar de maneira intuitiva a sequência lógica (*workflow*) utilizada para a idealização do presente trabalho, desenvolveu-se o fluxograma que pode ser visualizado na Figura 19.

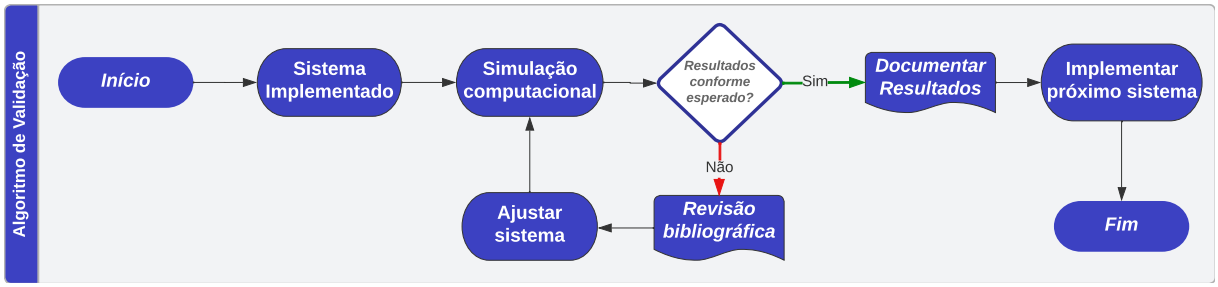
Figura 19 – Representação do *workflow* de desenvolvimento do sistema.



Fonte: Do autor

Fazendo-se alusão ao método de avaliação dos circuitos desenvolvidos, na Figura 19 a passagem de um passo de implementação para outro de complexidade maior se dá apenas mediante a aprovação do funcionamento do subcircuito implementado. Para tal, conforme pode ser visualizado na Figura 20, aplica-se um algoritmo de validação que consiste da averiguação dos resultados obtidos mediante simulação computacional. Nesse sentido, caso os resultados de determinado subcircuito estejam em desacordo com as premissas almejadas, o mesmo deve ser reajustado até a obtenção de resultados satisfatórios. Mediante aprovação no teste de validação, há a abertura de possibilidade para a implementação do circuito imediatamente superior na hierarquia.

Figura 20 – Representação do algoritmo de validação para cada etapa implementada.



Fonte: Do autor

3.2 SIMULAÇÃO COMPUTACIONAL

As simulações computacionais implementadas na presente investigação foram integralmente desenvolvidas através do software LTspice. O LTspice se traduz como uma ferramenta computacional robusta e gratuita, sendo altamente eficiente para a simulação de circuitos em arquitetura CMOS. Apesar de gratuito, o LTspice é definido como um software *freeware*, portanto não *open source*, o que acarreta na impossibilidade de modificação do código fonte.

No que tange aos métodos numéricos de simulação, o LTspice implementa o algoritmo de Newton-Raphson (método das tangentes), o que possibilita a resolução de equações diferenciais através de operações aritméticas de menor complexidade. A aplicação do método de Newthon-Raphson oferece uma aproximação satisfatória quando utilizado na simulação de circuitos lineares, estendendo-se inclusive, a circuitos que detêm de não linearidades monotônicas. Dessa maneira, em vista que os circuitos propostos na presente investigação são de natureza determinística e linear, o software LTspice apresenta-se como um recurso que atende integralmente as premissas de simulação numérica exigidas.

Com relação aos recursos de hardware utilizados no presente estudo, as simulações foram performadas em uma máquina com processador Intel Core i5-11400 de 6 núcleos,

dispondo de frequência base de 2,6 GHz com *turbo boost* de até 4,4 GHz, 16 GB de memória DDR4 (RAM) e 500 GB do tipo SSD.

O modelo em LTSpice dos transistores CMOS de 16nm implementados foi disponibilizado pelo *Nanoscale Integration and Modeling (NIMO) Group* (Grupo de Integração em Nanoescala e Modelagem) da *Arizona State University* (Universidade Estadual do Arizona) (NIMO, 2012). O grupo de pesquisa NIMO tem como premissa avançar no estado da arte em soluções interdisciplinares para o design de sistemas robustos e inteligentes, sendo guiado por avanços em nanoeletrônica e análise de dados. O NIMO disponibiliza diversos modelos de tecnologia preditiva em nanoeletrônica de 7nm a 180nm, dispondo de *layouts* específicos para aplicações que necessitam de alta performance e baixo consumo de potência.

Ao avaliar um sistema, projeto ou desempenho, várias métricas podem ser consideradas para medir diferentes aspectos e indicadores de sucesso. No presente estudo, foram avaliadas métricas relacionadas à performance do sistema proposto, com foco na acurácia obtida por meio da classificação dos dados de entrada pela rede neural. A acurácia foi utilizada como uma métrica fundamental para medir a precisão e o desempenho do sistema. Ela representa a proporção de amostras corretamente classificadas em relação ao total de amostras avaliadas. Ao calcular a acurácia, foi possível avaliar a eficácia do modelo de rede neural em classificar corretamente os dados fornecidos como entrada.

A utilização da acurácia como métrica de avaliação permite uma análise objetiva e quantitativa do desempenho do sistema. Quanto maior a acurácia obtida, maior é a capacidade do modelo neural em realizar classificações precisas e confiáveis.

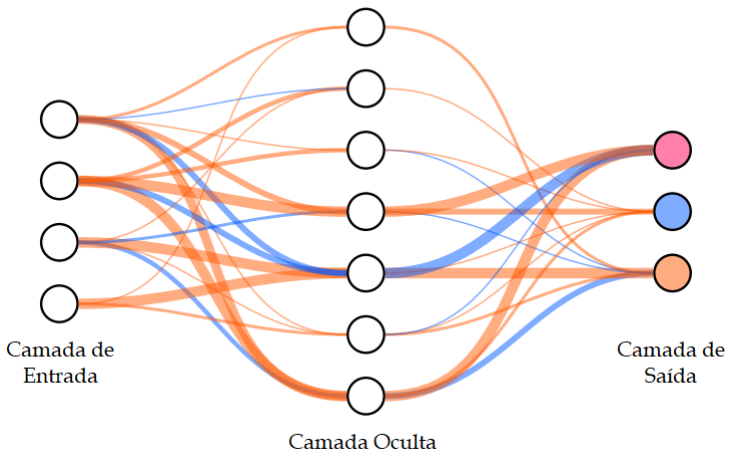
4 IMPLEMENTAÇÃO, ANÁLISE E DISCUSSÃO DOS RESULTADOS

Neste capítulo são apresentados e discutidos os resultados referentes ao desenvolvimento da NoC implementada e dos algoritmos de roteamento desenvolvidos. Para tal, faz-se uma análise a partir da implementação e simulação computacional dos circuitos em metodologia *bottom-up*.

4.1 IMPLEMENTAÇÃO DA NEURAL EM SOFTWARE

A implementação da rede neural para classificação de dados Iris de Fisher demanda a necessidade de modelar e implementar uma arquitetura de rede perceptron multicamada, a qual pode ser estruturada no formato $4 \times 7 \times 3$. Nesse sentido, são dispostos 4 neurônios na camada de entrada, 7 neurônios na camada oculta (processamento) e 3 neurônios na camada de saída. Uma ilustração das conexões entre os neurônios na rede perceptron multicamada para a classificação dos dados Íris de Fisher pode ser visualizada na Figura 21.

Figura 21 – Representação da rede neural $4 \times 7 \times 3$ treinada a partir do conjunto de dados *Iris* de Fisher.



Fonte: Adaptado de Binas *et al.* (2016).

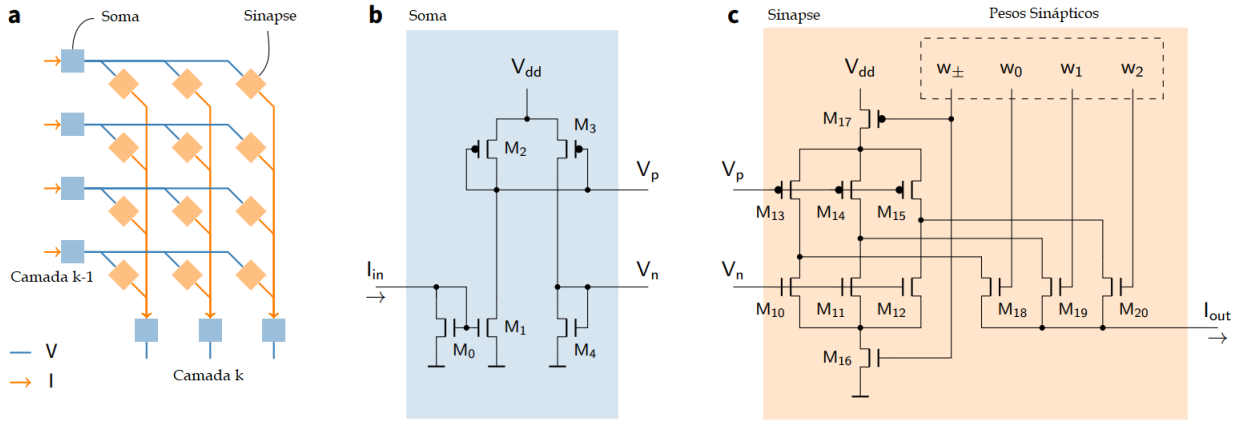
Nesta configuração, a disposição de 4 neurônios de entrada permite o armazenamento e disparo das informações para o início do processamento de classificação da flor de Íris. Dessa maneira, cada neurônio apresenta um dos 4 dados de entrada individualmente: comprimento da sépala, largura da sépala, comprimento da pétala e largura da pétala.

De maneira simplificada, os dados de entrada são processados pelos 7 neurônios da camada oculta, os quais disparam as informações aos neurônios da camada de saída. Baseando-se nos resultados do processamento da camada oculta, os 3 neurônios da camada

de saída permitem a classificação dos dados de entrada entre uma das 3 espécies do gênero *Iris*, sendo elas: *Iris setosa*; *Iris versicolor* e; *Iris virginica*.

A arquitetura da rede perceptron multicamada foi implementada a partir do modelo de neurônio analógico proposto no trabalho de Binas *et al.* (2020) (BINAS *et al.*, 2020). O modelo de Binas *et al.* (2020) permite a obtenção de dois circuitos para a modelagem do neurônio computacional: o circuito de Soma e o circuito de Sinapse. Os modelos conceituais de arquitetura de rede neural, bem como os circuitos de Soma e Sinapse, podem ser visualizados na Figura 22.

Figura 22 – Estrutura de conexão dos circuitos de soma e sinapse dos neurônios nanoeletrônicos: a) Conexão em rede entre Soma-Sinapse através de k camadas neuronais, b) Circuito de Soma de um neurônio computacional e, c) Circuito de Sinapse programável de 3-bit de um neurônio computacional.



Fonte: Adaptado de Binas *et al.* (2020).

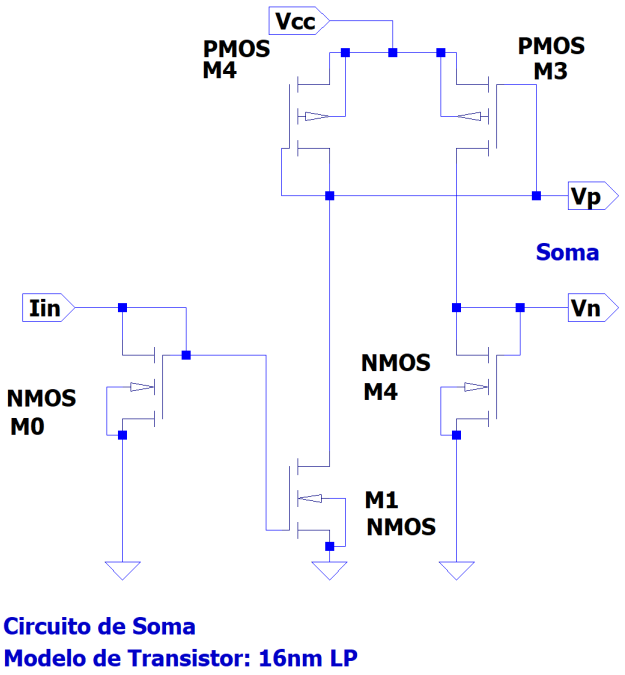
A implementação do modelo de Binas *et al.* (2020) se dá mediante a conexão física de múltiplas camadas de circuitos de Soma através de matrizes de circuitos de Sinapse. Dessa forma, a integração de i circuitos de Soma através de matrizes contendo j circuitos de Sinapses resulta em uma rede perceptron de k camadas, em que necessariamente $i >= 1; j >= 1; k >= 3$.

Tendo em vista que cada circuito de soma m da camada k se conecta a cada circuito de soma n da camada $k+1$ através de S sinapses, a interação entre uma camada e outra se dá através de $S = m \times n$ sinapses. Portanto, para o caso da rede neural para classificação dos dados Íris de Fisher, a presença de 3 camadas com disposição de neurônios do tipo $4 \times 7 \times 3$, apresenta um total de $S = 4 \times 7 + 7 \times 3 = 49$ sinapses.

De maneira geral, a partir da arquitetura de rede da Figura 22.a, a saída dos circuitos de soma é transmitida em forma de tensão a uma série de circuitos de sinapse, os quais implementam multiplicações por escalares baseados nos pesos sinápticos. A conexão das saídas dos circuitos de sinapse da camada $k-1$ permite a soma das correntes

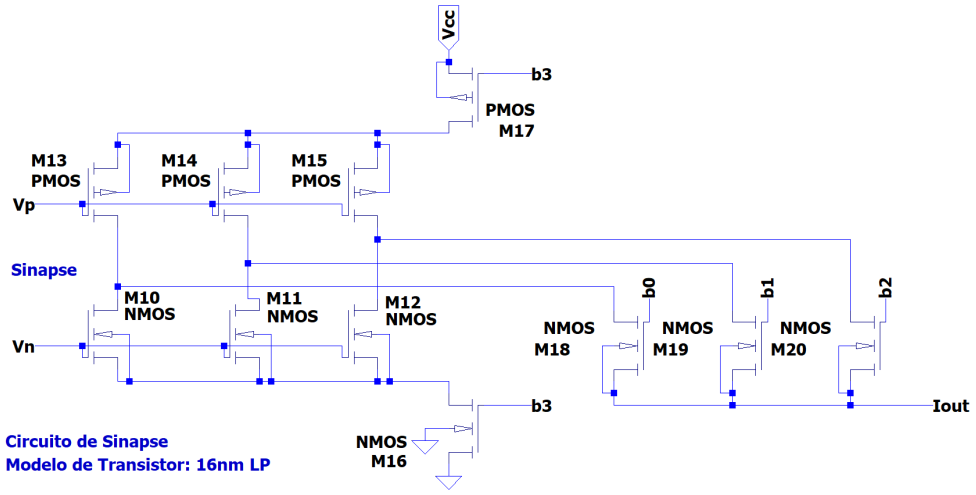
I_{out}, alimentando os circuitos de soma da camada k, os quais aplicam a não-linearidade. Partindo-se do modelo conceitual presente nas Figuras 22.b e 22.c, tem-se os circuitos de de Soma e Sinapse em software LTspice. Portanto, nas Figuras 23 e 24 são apresentados os circuitos de Soma e Sinapse, respectivamente.

Figura 23 – Circuito de Soma em software computacional.



Fonte: Do autor.

Figura 24 – Circuito de Sinapse em software computacional.



Fonte: Do autor.

No circuito de Soma da Figura 22.b, uma corrente I_{in} é retificada por um transistor M_0, o que resulta na obtenção de duas tensões de saída: V_p e V_n. As tensões de saída de um circuito de Soma atuam como as entradas dos circuitos de Sinapse (Figura 22.c) conectados. Os elementos transistores presentes nos circuitos de Soma e Sinapse atuam como amplificadores rudimentares de corrente (espelhos de corrente), resultando em uma saída amplificada I_{out} no circuito de Sinapse. Portanto, a saída I_{out} de uma sinapse é uma versão amplificada da corrente de entrada I_{in} retificada, a qual foi introduzida no circuito de Soma.

O comportamento observado na amplificação da corrente I_{in} através do neurônio nos circuitos de soma e sinapse, indica a aplicação de uma função de ativação ReLU (retificadora) pelo neurônio. Na configuração do circuito da Figura 22.c, a presença dos elementos $w\pm, w_2, w_1, w_0$; controla a aplicação da função de ativação ReLu através da introdução de pesos sinápticos ao sistema. Os elementos $w\pm, w_2, w_1, w_0$ ficam armazenados em um elemento de memória digital intrínseco do neurônio, diminuindo o gargalo entre elementos de processamento e de memória e mitigando um problema típico da arquitetura Von Neumann.

Dessa maneira, a implementação da função de ativação ReLu nos circuitos de Sinapse está configurada a partir de 2×3 amplificadores, os quais podem ser ativados ou desativados a partir de 2^4 possibilidades de pesos sinápticos, determinados através do ajuste de $w\pm, w_2, w_1, w_0$. Logo, os pesos sinápticos operam apenas nos estados binários "ligado" e "desligado" (*on* e *off*, respectivamente), portanto o conjunto de valores para w_2, w_1, w_0 fica restrito aos números binários contidos no intervalo 000 e 111.

O fator de escala, o qual afeta diretamente a função de ativação ReLu para cada um dos amplificadores ou espelhos de corrente, é determinado a partir da contribuição do peso sináptico b juntamente com o fator de largura W do par de transistores que compõe o amplificador. A fim de simplificar os circuitos amplificadores dos neurônios, optou-se pela utilização de valores dimensionais de comprimento (L) e largura (W) constantes para todos os transistores, conforme pode ser visualizado na Tabela 3.

Tabela 3 – Relação de dimensionamento dos transistores de 16 nm implementados.

Transistor	W	L	W/L
$M_0 - M_{20}$	96 nm	16 nm	6 nm

Fonte: Do autor.

Os pesos sinápticos foram determinados a partir dos dados obtidos pela investigação de Binas *et al.* (2016), na qual uma rede neural estruturada no formato $4 \times 7 \times 3$ e implementada em tecnologia CMOS de 180nm foi treinada via uso do algoritmo de *backpropagation* a partir de 80% dos dados provenientes do conjunto de dados Íris de Fisher. Nesse sentido, a Tabela 4 representa os pesos sinápticos para as sinapses entre a camada de entrada e oculta. Da mesma forma, a Tabela 5 apresenta os pesos sinápticos

entre as camadas oculta e de saída.

Tabela 4 – Relação de pesos sinápticos da camada de entrada com relação à camada oculta (notação $w\pm, w_2w_1w_0$) do modelo treinado por Binas *et al.* (2016).

Camada de Entrada	**Camada Oculta**						
	N_1	N_2	N_3	N_4	N_5	N_6	N_7
N_1	-,010	+001	-,001	-,100	+,100	-,001	-,110
N_2	000	-,011	-,011	-,111	+,100	000	-,111
N_3	000	-,001	000	+,010	-,111	-,001	+,011
N_4	-,001	000	000	000	-,111	-,010	000

Fonte: Do autor.

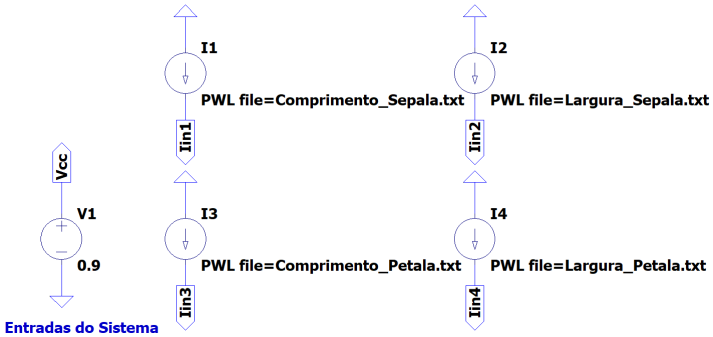
Tabela 5 – Relação de pesos sinápticos da camada oculta com relação à camada de saída (notação $w\pm, w_2w_1w_0$) do modelo treinado por Binas *et al.* (2016).

Camada Oculta	**Camada de Saída**		
	N_1	N_2	N_3
N_1	000	000	-,010
N_2	000	-,001	000
N_3	000	-,001	+,001
N_4	-,111	-,100	+,001
N_5	+,111	-,001	-,111
N_6	+,001	-,001	-,010
N_7	-,111	-,010	+,100

Fonte: Do autor.

Os dados contidos nas Tabelas 4 e 5 foram implementados a partir do uso de 4 fontes de tensão para cada neurônio, as quais representavam os sinais w_0, w_1, w_2 e $w_\pm$. A amplitude dos sinais w_0, w_1, w_2 assume valores 0 ou 1, enquanto o sinal $w_\pm$ assume os valores de -1, 0 ou 1. Para a alimentação dos circuitos de soma e sinapse, fez-se uso de uma fonte de tensão estática de $0,9$V. Sendo assim, a Figura 25 representa as fontes de entrada da rede neural implementada.

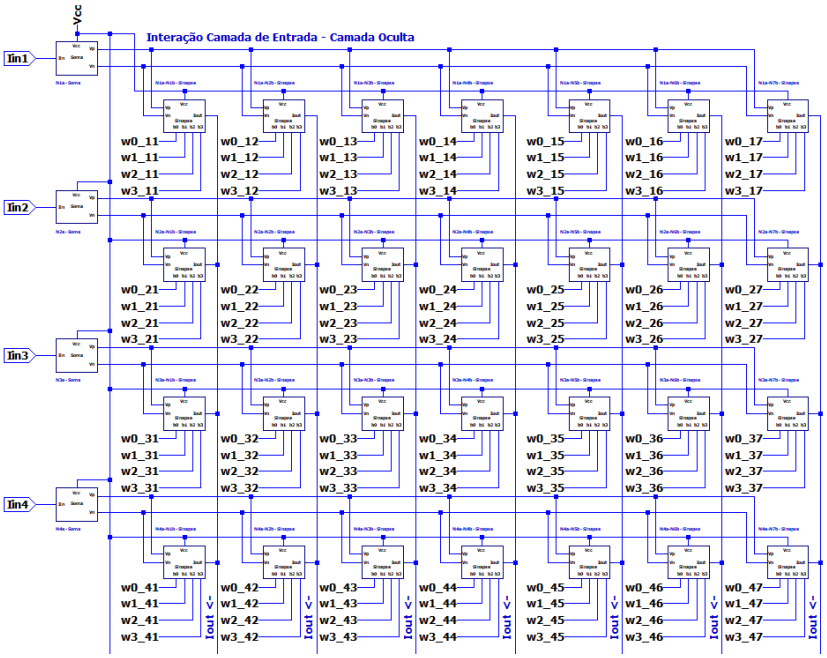
Figura 25 – Fontes de corrente e tensão para a geração dos sinais de entrada do sistema.



Fonte: Do autor.

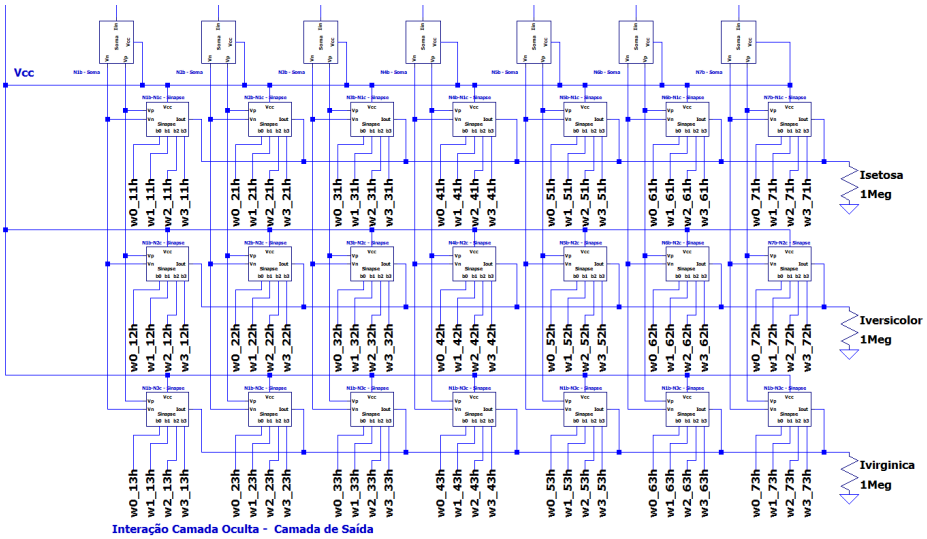
A partir da estrutura de conexão em rede presente na Figura 22.a, construiu-se a rede neural em *software* LTspice. A conexão entre os neurônios da camada de entrada e camada oculta é fornecida na Figura 26. Da mesma forma, a conexão entre os elementos da camada oculta com a camada de saída é dada na Figura 27.

Figura 26 – Estrutura de conexão entre neurônios da camada de entrada e camada oculta.



Fonte: Do autor.

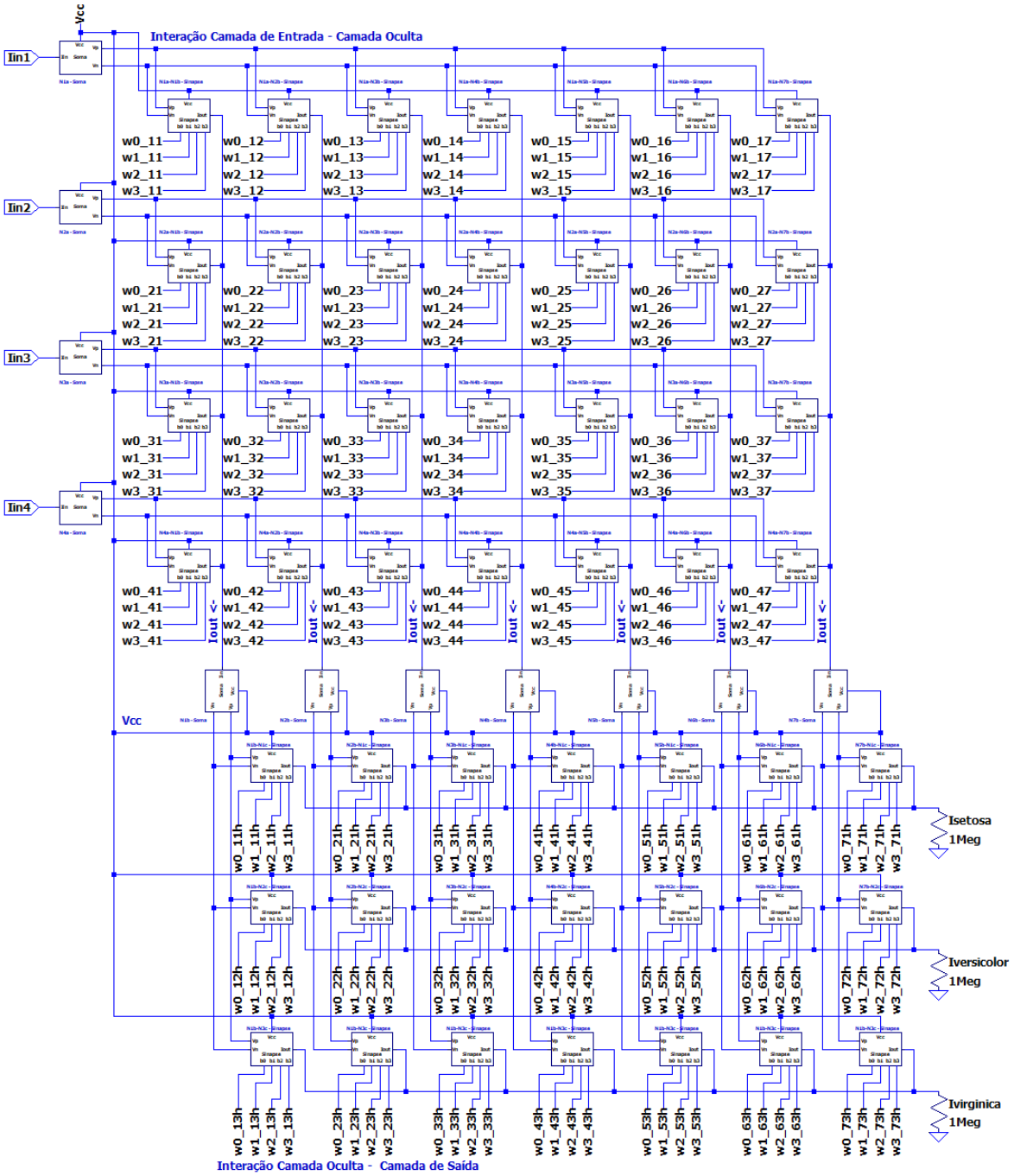
Figura 27 – Estrutura de conexão entre neurônios da camada oculta e camada de saída.



Fonte: Do autor.

A partir da integração entre os neurônios das camadas de entrada, oculta e de saída, obteve-se o modelo completo da rede neural. Devido o design do sistema ter sido implementado utilizando-se de metodologia hierárquica, os circuitos de soma e sinapse (ver Figuras 23 e 24) encontram-se intrínsecos nos blocos de sub circuitos. Portanto, uma visualização integral do sistema interconectado pode ser visualizada na Figura 28.

Figura 28 – Estrutura de conexão em rede entre neurônios da camada oculta e camada de saída.



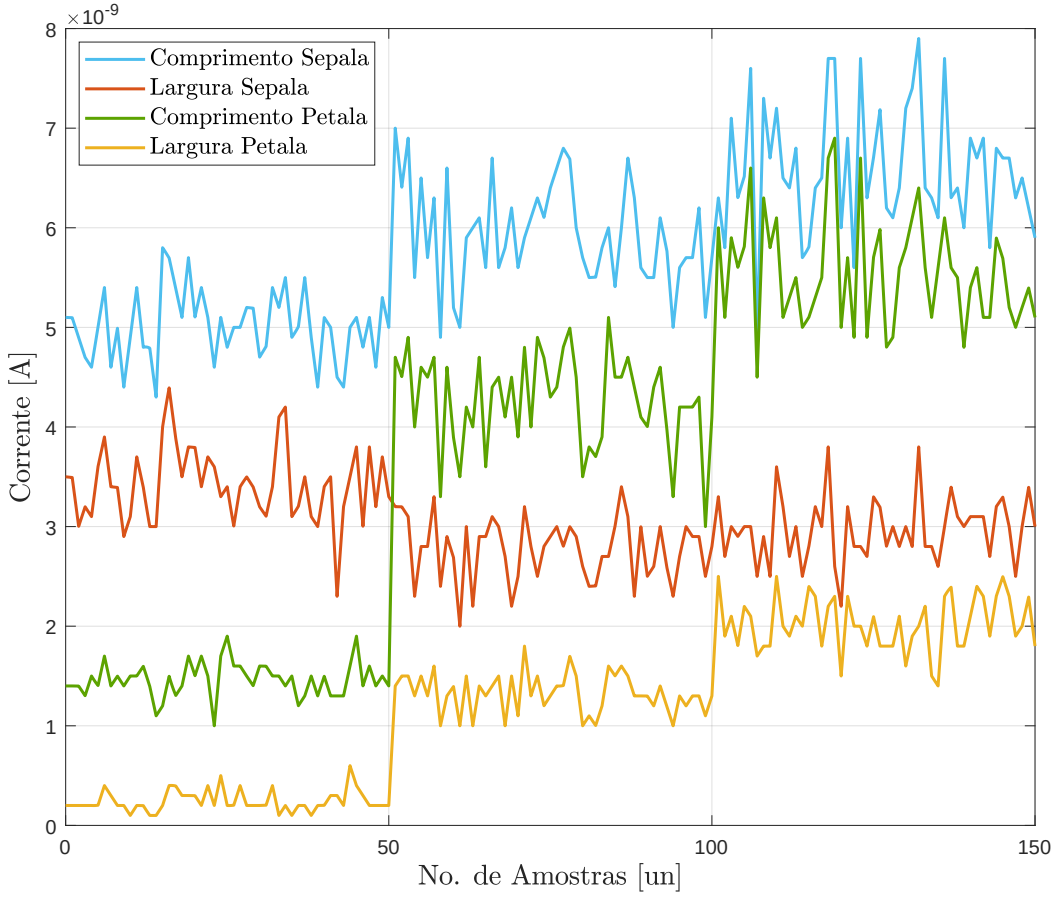
Fonte: Do autor.

4.2 SIMULAÇÃO DO SISTEMA PROPOSTO

A simulação do sistema foi realizada a partir da análise transiente do *software* LTspice. A análise transiente permite a obtenção do comportamento de várias grandezas do circuito dentro de um determinado tempo t. Os dados de entrada do sistema (Comprimento da Sépala, Largura da Sépala, Comprimento da Pétala e Largura da Pétala), os quais encontram-se explícitos no Anexo B, foram inseridos através de fontes de corrente respectivamente nas entradas I_{in1}, I_{in2}, I_{in3} e I_{in4}, assumindo-se ordem de grandeza de 10^{-9}A.

Dessa maneira, o modelo foi simulado considerando cada tempo t como sendo referente a sua unidade amostral respectiva. Portanto, os sinais foram simulados para $t = 150$, considerando-se um passo Δt unitário (amostral). Para tal, o conjunto de dados foi convertido em arquivos .txt para a geração de sinais PWL (*Piecewise linear*) respectivos à cada entrada do sistema. Sendo assim, na Figura 29 são apresentadas as curvas de entrada da rede neural.

Figura 29 – Comparação dos sinais de entrada do sistema referentes às três espécies do gênero *Iris*.

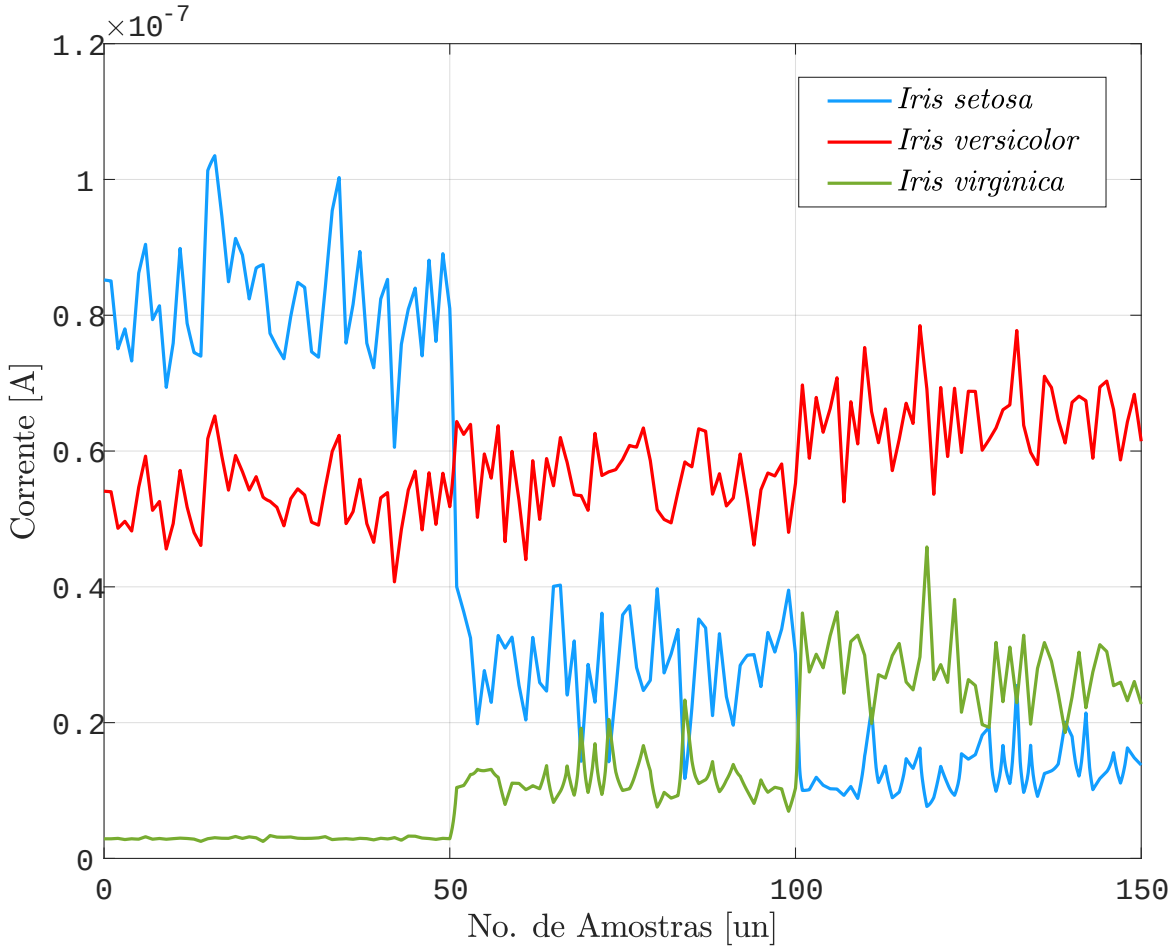


Fonte: Do autor.

Da Figura 29, é possível averiguar diferenças consideráveis na morfologia floral das três espécies do gênero *Iris*, uma vez que há uma clara diferença nos conjuntos amostrais [0,50], [51, 100] e [101,150], os quais referem-se respectivamente às espécies *Iris setosa*, *Iris versicolor* e *Iris virginica*. Todavia, é importante se atentar ao fato de que variáveis como o comprimento da sépala e largura da sépala, apresentam grande similaridade para as espécies *Iris versicolor* e *Iris virginica*.

O comportamento de classificação da rede foi obtido a partir das correntes de saída I_{setosa}, $I_{versicolor}$ e $I_{virginica}$, as quais foram mensuradas através cargas resistivas de $1 \cdot 10^6$ Ω. A mensuração através de cargas resistivas em tal grandeza busca recriar a presença de elementos digitais conectados às saídas do sistema. Nesse sentido, a presença de conversores analógico digitais possibilitaria a integração da rede neural em NoCs, os quais conectariam os núcleos de processamento IP, definidos pelos neurônios, aos elementos roteadores. Nesse sentido, a Figura 30 apresenta as correntes na saída do sistema.

Figura 30 – Correntes de saída para o modelo treinado por Binas *et al.* (2016).



Fonte: Do autor.

Para a análise dos resultados presentes na Figura 30, convém-se observar a taxa de precisão na classificação das 3 espécies pela rede neural implementada. Portanto, mediante

a estruturação dos dados em software de análise numérica, os quais foram exportados das curvas obtidas em LTspice, foi possível aferir a eficácia da rede neural utilizando-se os dados de treinamento da investigação de Binas *et al.* (2020). Nesse sentido, dentro de cada conjunto amostral [1,50], [51,100] e [101,150], o sinal com maior amplitude representa a espécie classificada pela rede neural. Sendo assim, na Tabela 6 é apresentada a análise numérica dos dados obtidos pelas curvas da Figura 30.

Tabela 6 – Resultados numéricos de performance da rede neural treinada pelo modelo de Binas *et al.* (2016)

Espécies	No. de Amostras	No. Acertos	No. Falhas	Taxa de Acurácia
Iris setosa	50	50	0	100,00%
Iris versicolor	50	50	0	100,00%
Iris virginica	50	0	50	0%
Total da Rede	150	100	50	**66,66%**

A partir da análise dos dados obtidos, bem como cruzando-se com os resultados referência do conjunto de dados Íris de Fisher (ver Anexo B), observa-se que o modelo fornecido pela investigação de Binas *et al.* (2016) apresenta resultados satisfatórios na classificação das espécies *Iris setosa* e *Iris versicolor*, pois apresentou o maior sinal de saída correto para os conjuntos [1,50] e [51,100]. Entretanto, o modelo apresenta eficácia insuficiente na classificação da espécie *Iris virginica*, visto que não obteve nenhum acerto dentro do conjunto [101,150], classificando erroneamente os dados de entrada como sendo pertencentes à espécie *Iris versicolor*, com larga diferença com relação à espécie correta.

Sendo assim, apesar de apresentar eficácia total de 100% na classificação do conjunto de dados Íris de Fisher utilizando-se tecnologia CMOS de 180nm, o modelo proposto pela investigação de Binas *et al.* (2016) apresentou apenas $66,66\%$ de precisão quando implementado via uso de transistores CMOS de 16nm. Entretanto, apesar do resultado insatisfatório com relação à classificação da espécie *Iris virginica*, a correta classificação das demais espécies não descarta a validade do modelo.

Tabela 7 – Pesos sinápticos da rede neural retreinada.

Camada Oculta	**Camada de Saída**		
	N_1	N_2	N_3
N_1	000	-,111	-,111
N_2	000	-,111	000
N_3	000	-,001	+,001
N_4	-,111	000	+,001
N_5	+,111	000	-,111
N_6	+,001	-,001	-,010
N_7	-,111	-,010	+,100

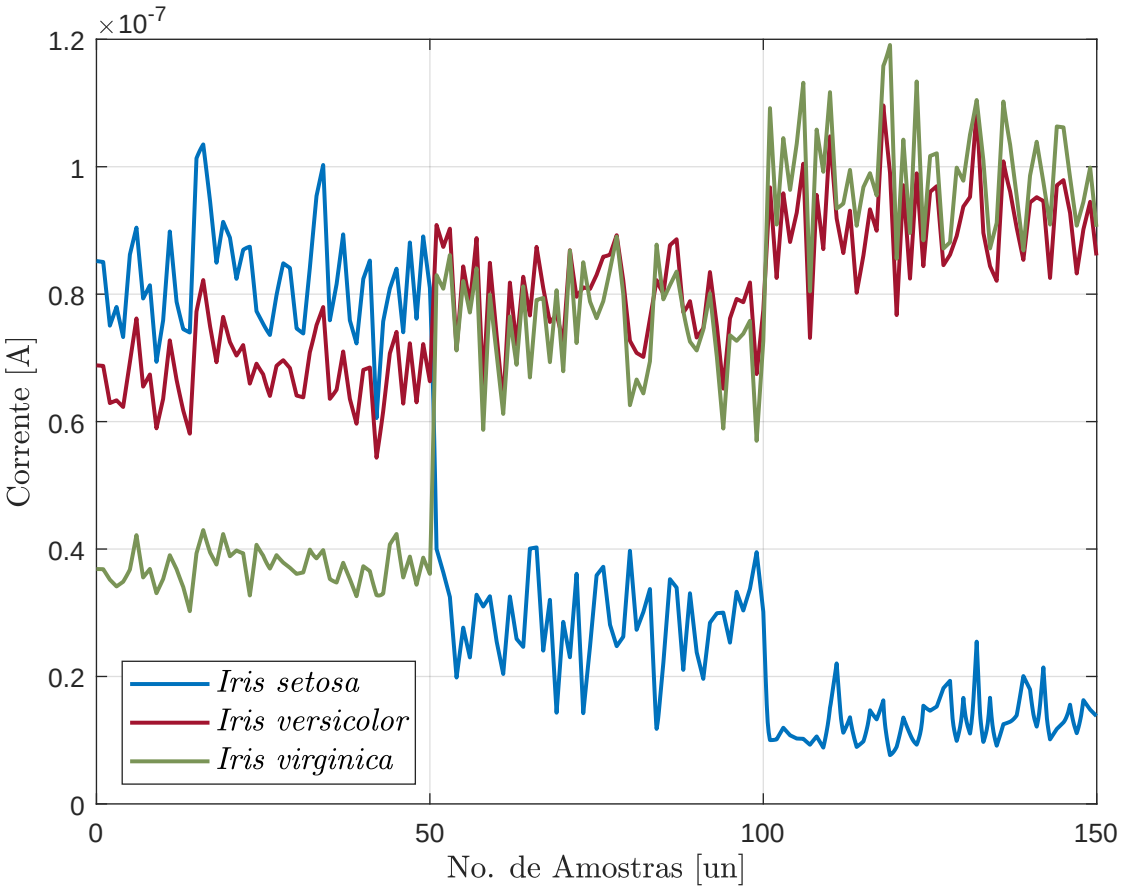
Fonte: Do autor.

Dessa maneira, com o intuito de obter uma rede neural nanoeletrônica capaz de realizar a classificação satisfatória de todo o conjunto de dados Íris de Fisher, realizou-

se o retreinamento da rede a partir da alteração manual via tentativa e erro dos pesos sinápticos dos neurônios entre a camada oculta e a camada de saída. Uma vez que os dados de entrada referentes à espécie *Iris virginica* foram classificados como sendo pertencentes à espécie *Iris versicolor*, os pesos sinápticos dos neurônios N_2 e N_3 da camada de saída, os quais detém de maior influência na classificação de ambas as espécies, foram ajustados. Portanto, na Tabela 7 são apresentados os pesos sinápticos da rede retreinada.

Alterando-se os pesos sinápticos no modelo em software computacional e realizando-se novamente uma simulação de natureza transiente, obteve-se a dinâmica de classificação da rede retreinada. Deste modo, a Figura 31 demonstra as correntes de saída do sistema.

Figura 31 – Correntes de saída para o modelo retreinado.



Fonte: Do autor.

Dos resultados obtidos a partir da simulação do modelo retreinado, faz-se necessária a realização da análise numérica para a distinção das curvas, principalmente no que tange às espécies *I. versicolor* e *I. virginica*. À vista disso, realizou-se procedimento análogo ao empregado na verificação dos dados expostos na Tabela 6. Portanto, no que segue tem-se os resultados numéricos de eficácia para a rede neural ajustada.

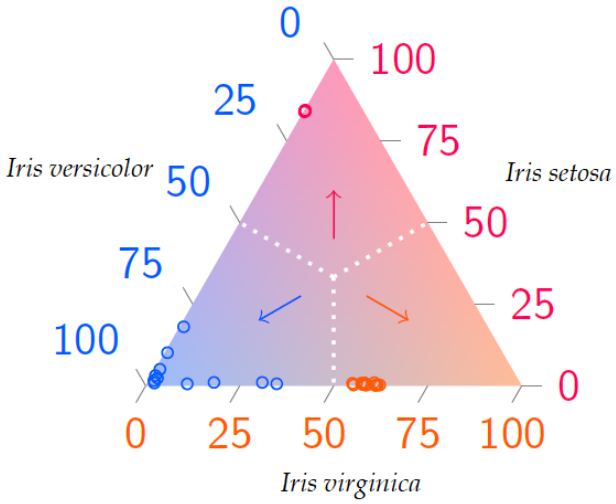
Tabela 8 – Resultados numéricos de performance da rede neural retreinada.

Espécies	No. de Amostras	No. Acertos	No. Falhas	Taxa de Acurácia
Iris setosa	50	50	0	100,00%
Iris versicolor	50	46	4	92,00%
Iris virginica	50	50	0	100,00%
Total da Rede	150	146	4	**97,33%**

Ao analisar a performance obtida da rede retreinada, observa-se um aumento considerável de acurácia na classificação do *data set*. Nesse sentido, o aumento de 66,66% para 97,33% representa um avanço de 46,09% na taxa de acurácia da rede, o que demonstra uma alta eficácia do algoritmo de pesos sinápticos implementado. Dessa maneira, em apenas 4 casos (amostras de nº 69, 73, 84 e 88) houve a falha do sistema em determinar a correta classificação das espécies, nais quais a rede neural indicou a espécie *Iris virginica* ao invés da espécie *Iris versicolor*.

A validade dos resultados obtidos é reforçada pela comparação com diversos estudos recém publicados, os quais encontram-se disponíveis na literatura. Nesse sentido, na investigação de Srinivasarao (2022) são aplicadas diversas técnicas computacionais para a classificação do *dataset*, tais como o classificador k-nn (*k-nearest neighbors*) para $k = 5$ e a regressão logística, obtendo-se eficácia de $96,66\%$ e 95,00%, respectivamente (SRINIVASARAO, 2022). Por sua vez, o trabalho de Mohsin Abdulazeez *et al.* (2021) evidencia altas taxas de acurácia utilizando-se dos algoritmos de árvores de decisões (98%), florestas de decisão aleatória ($99,33\%$) e o classificador k-nn (*k-nearest neighbors*) para $k = 10$ (MOHSIN ABDULAZEEZ *et al.*, 2021).

Figura 32 – Classificação do *dataset* pelo estudo de Binas *et al.* (2016) exibido em coordenadas baricêntricas.



Fonte: Adaptado de Binas *et al.* (2016).

O estudo de Pinto, Kelur e Shetty (2018), ao implementar as técnicas de *machine learning* SVM (*Support Vector Machine*), regressão logística e o classificador k-nn (*k-nearest neighbors*) para $k = 3$, obteve como resultados 96%, 91% e 93%, respectivamente, na classificação do conjunto de dados Iris de Fisher. Além do mais, tal investigação reforça a grande similaridade dos parâmetros do *dataset* para as espécies *I. versicolor* e *I. virginica*, corroborando em justificar a dificuldade das redes neurais na classificação 100% precisa dos dados provindos do *dataset* (PINTO; KELUR; SHETTY, 2018).

A semelhança entre os dados das espécies *I. versicolor* e *I. virginica* é reforçada pela análise empregada na investigação de Binas *et al.* (2016). Tal como pode ser visualizado na Figura 32, o resultado da classificação realizada pelo sistema para a espécie *Iris virginica* é próximo ao limiar que separa a espécie do seu par *I. versicolor*, reforçando a similaridade entre as espécies.

4.3 ALGORITMOS DE ROTEAMENTO

A integração dos elementos de processamento (cores IP) em redes-em-chip se dá através do uso de elementos roteadores (FÉ, 2017). Os algoritmos de roteamento são usados em redes de computadores para determinar a melhor maneira de encaminhar dados de um ponto a outro. Eles ajudam a otimizar a eficiência da rede, minimizar a latência e maximizar a taxa de transferência. Existem vários algoritmos de roteamento, cada um com suas próprias características e métodos de tomada de decisão. Cada um deles tem suas vantagens e desvantagens e é mais adequado para diferentes tipos de redes e requisitos específicos (KUROSE; ROSS, 2014).

Em algoritmos de roteamento estático, como os algoritmos XY, as rotas são configuradas manualmente pelos administradores de rede. Elas não mudam com frequência, a menos que haja uma reconfiguração manual. O roteamento estático é adequado para redes pequenas e estáveis, onde as alterações na topologia da rede são raras. Em vista que o presente modelo apresenta uma topologia de rede estável do tipo *Torus*, é possível propor algoritmos de rede estáticos que simulem a integração da rede neural implementada utilizando de NoCs.

O algoritmo de roteamento XY é uma estratégia comumente utilizada em NoCs. Nesse algoritmo, os pacotes de dados são roteados em duas etapas, uma para a direção X e outra para a direção Y. Em cada etapa, o pacote é movido para a próxima posição na rede, de acordo com as coordenadas X e Y do destino final. O roteamento ocorre em malhas bidimensionais, onde as conexões entre os elementos da rede são organizadas em uma matriz (MAHENDRA; GAIKWAD; PATRIKAR, 2016).

Com base nos conceitos apresentados anteriormente, na presente investigação foram desenvolvidos modelos conceituais de algoritmos de roteamento. Os modelos conceituais de algoritmos de roteamento permitem uma representação abstrata dos diferentes métodos e estratégias utilizados para determinar as rotas mais eficientes em uma rede de comuni-

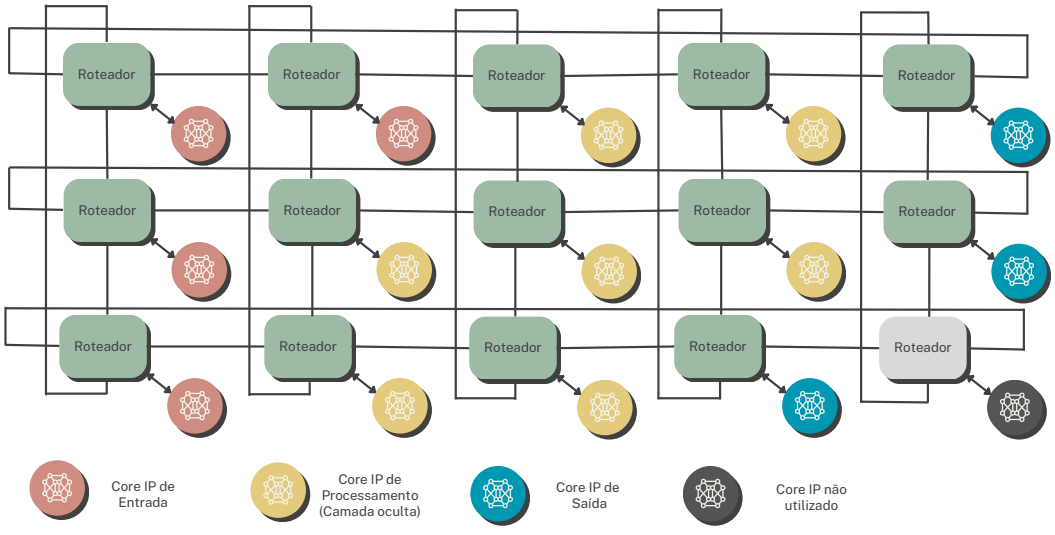
cação. Eles descrevem os principais componentes e as interações envolvidas no processo de roteamento.

Sendo assim, assumindo as variáveis x e y como sendo respectivas à posição do roteador na matriz (x, y) dentro de uma NoC em arquitetura *Torus* genérica $m \times m$ e tendo como base o algoritmo XY proposto pela investigação de Yiping e Takahiro (2008) (*XY routing algorithm*), propõe-se o seguinte algoritmo de roteamento estático para uma arquitetura *Torus* genérica:

- Caso $0 < x_2 - x_1 \leq \frac{m}{2}$ ou $-\frac{m}{2} \leq x_2 - x_1$, a transmissão de dados do roteador se dará no sentindo *leste*;
- Caso $\frac{m}{2} < x_2 - x_1$ ou $x_2 - x_1 < -\frac{m}{2}$, a transmissão de dados do roteador se dará no sentindo *oeste*;
- Se $x_2 - x_1 = 0$, cai-se em 2 situações, as quais:
    1. Caso $0 < y_2 - y_1 \leq \frac{m}{2}$ ou $-\frac{m}{2} \leq y_2 - y_1$, a transmissão se dará no sentido *sul*;
    2. Caso $\frac{m}{2} < y_2 - y_1$ ou $y_2 - y_1 < -\frac{m}{2}$, a transmissão se dará no sentido *norte*;
- Se nenhuma das condições forem atendidas, a transmissão será no sentido *roteador-core IP*.

Considerando-se cada neurônio da Figura 28 como sendo um Core IP, a rede neural pode ser implementada em uma NoC com arquitetura *Torus* do tipo 3×5 $(n \times m)$, conforme pode ser visualizado na Figura 33.

Figura 33 – Visualização da NoC implementada em arquitetura *Torus* 3×5.



Fonte: Do autor.

A partir do modelo da Figura 28, tem-se o seguinte algoritmo de roteamento XY para uma rede $n \times m$:

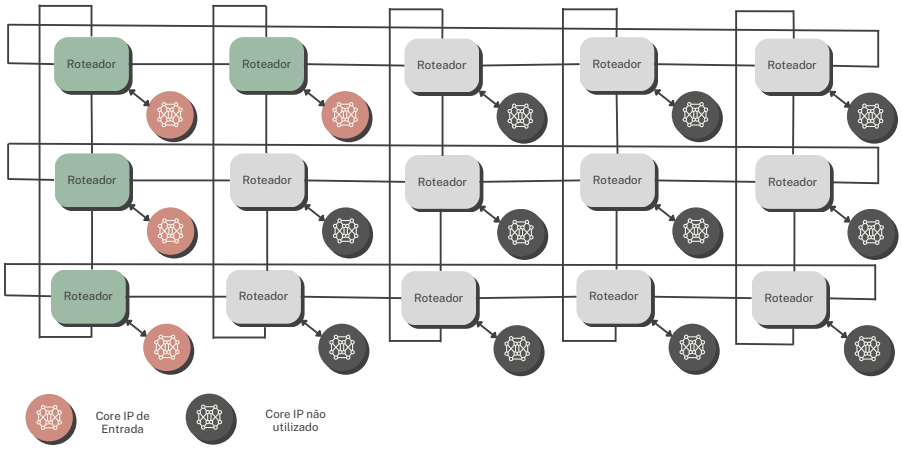
- Caso $0 < x_2 - x_1 \leq \frac{m}{2}$ ou $-\frac{m}{2} \leq x_2 - x_1$, a transmissão de dados do roteador se dará no sentindo *leste*;
- Caso $\frac{m}{2} < x_2 - x_1$ ou $x_2 - x_1 < -\frac{m}{2}$, a transmissão de dados do roteador se dará no sentindo *oeste*;
- Se $x_2 - x_1 = 0$, cai-se em 2 situações, as quais:
    1. Caso $0 < y_2 - y_1 \leq \frac{n}{2}$ ou $-\frac{n}{2} \leq y_2 - y_1$, a transmissão se dará no sentido *sul*;
    2. Caso $\frac{n}{2} < y_2 - y_1$ ou $y_2 - y_1 < -\frac{n}{2}$, a transmissão se dará no sentido *norte*;
- Se nenhuma das condições forem atendidas, a transmissão será no sentido *roteador-core IP*.

Conforme demonstrado pelo estudo de Mahendra, Gaikwad e Patrikar (2016) algoritmos de roteamento XY apresentam grande vantagem em redes NoC de topologia *Torus* quando comparados em aplicações de redes NoC em arquitetura *Mesh* (YI-PING; TAKAHIRO, 2008; MAHENDRA; GAIKWAD; PATRIKAR, 2016). Nesse sentido, considerando-se a necessidade de transmissão de informações do roteador na posição $(0, 0)$ para o roteador na posição $(2, 4)$, o caminho ótimo de transmissão apresenta apenas um roteador na rota (roteador da posição $(0, 4)$ ou o roteador da posição $(2, 0)$). Por sua vez, realizando-se o mesmo procedimento em uma arquitetura *Mesh* os pacotes passariam por ao menos 5 roteadores até o destino final.

O trabalho de Dong *et al.* (2010) propõe um algoritmo de roteamento estático em que cada camada (entrada, oculta e saída) é implementada em um intervalo de tempo diferente utilizando-se a mesma estrutura de hardware (DONG *et al.*, 2010). Dessa maneira, tomando-se como relevância a NoC em topologia *Torus* 3×5 implementada, o algoritmo funcionaria da seguinte forma:

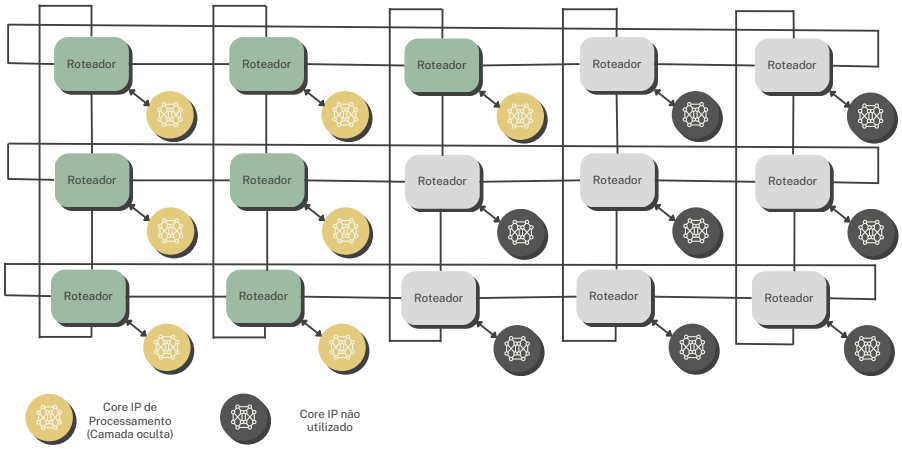
1. A camada de entrada recebe os pesos sinápticos e os valores de entrada, os quais são armazenados em um elemento de memória, conforme pode ser visualizado na Figura 34.
2. A camada oculta realiza a leitura dos dados armazenados pela camada de entrada, realizando o processamento das informações e armazenando o resultado em um elemento de memória. Este processo por ser conferido conforme ilustrado na Figura 35.
3. A camada de saída procede com a leitura dos dados armazenados no elemento de memória utilizados pela camada oculta. Após a leitura, os dados são computados resultando nas saídas do sistema. A Figura 36 demonstra a implementação da camada de saída.

Figura 34 – Implementação da camada de entrada na NoC.



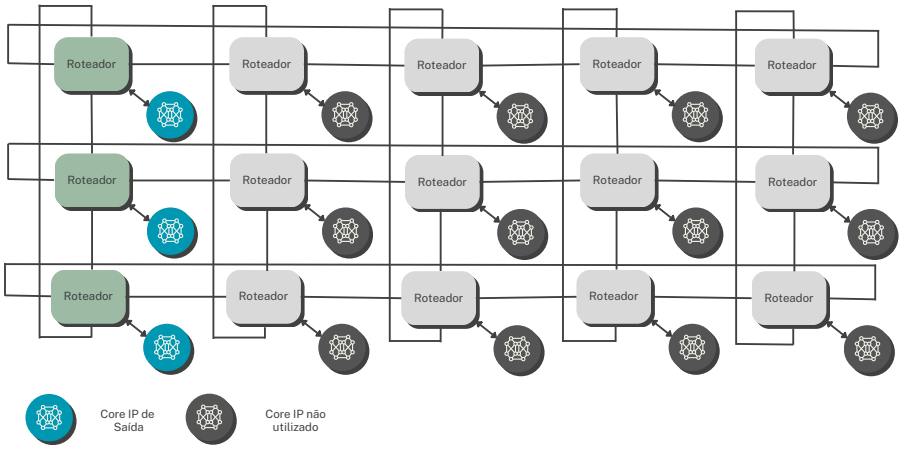
Fonte: Do autor.

Figura 35 – Implementação da camada oculta na NoC.



Fonte: Do autor.

Figura 36 – Implementação da camada de saída na NoC.



Fonte: Do autor.

A implementação da estratégia de roteamento e configuração da rede vista nas Figuras 34, 35 e 36, permite a obtenção de uma rede-em-chip altamente modular e escalável. A obtenção de uma rede reconfigurável é de alto interesse no contexto dos sistemas neuromórficos, visto que tal modelo permite o uso de uma mesma malha de roteadores e cores IP para diferentes problemas de *machine learning*. Por exemplo, para uma rede perceptron de $n = 3$ camadas implementada em uma NoC com arquitetura *Torus* 5×5 a limitação do número de cores IP para processamento passa de 25 para $3 \times 5 \times 5 = 75$ cores IP, o que representa um aumento de 200% no potencial de processamento do sistema.

5 CONSIDERAÇÕES FINAIS

A presente investigação teve como foco o desenvolvimento de um sistema neuromórfico em nanoescala utilizando-se de tecnologia CMOS de 16nm. Para tal, foi implementado o modelo proposto por Binas *et al.* (2016) na classificação do conjunto de dados Iris de Fisher (FISHER, 1936). A partir dos dados obtidos da simulação do modelo de Binas *et al.* (2016), houve o arcabouço para propor melhorias no sistema visando a obtenção de uma melhor performance da rede neural. A alta taxa de acertos obtida na classificação do conjunto de dados, bem como a comparação dos resultados com investigações em perspectivas semelhantes, afere a validade do uso de transistores em nanoescala em aplicações utilizando-se de inteligência artificial.

A classificação do conjunto de dados Iris de Fisher nunca foi implementada em escalas tão pequenas. A aplicação de transistores CMOS de 16nm representou uma diminuição aproximada de 91.11% no tamanho dos transistores quando comparada com a implementação da rede neural via uso de transistores CMOS de 180nm proposta por Binas *et al.* (2016). Tal característica demonstra a iniciativa inovadora do presente trabalho, além de reforçar a viabilidade da implementação de tecnologia nanoeletrônica para o desenvolvimento de sistemas neuromórficos.

Em perspectivas semelhantes a abordada na presente investigação, observa-se que existem várias iniciativas de pesquisa e projetos em andamento que buscam explorar a engenharia neuromórfica e a nanoeletrônica. Apesar dessas abordagens ainda estarem em pleno desenvolvimento, tais iniciativas buscam em conjunto o aprimoramento dos conceitos, o desenvolvimento de novas tecnologias e a exploração de aplicações reais utilizando sistemas neuromórficos em nanoescala.

Os desafios impostos no decorrer do desenvolvimento do presente estudo, bem como as possibilidades inimagináveis da inteligência artificial em nanoescala implementada em NoCs, promoveram *insights* úteis para o desenvolvimento de investigações futuras. Nesse sentido, como propostas de trabalho futuro destacam-se:

- Exploração de outros problemas reais de classificação via uso de *datasets* clássicos de *machine learning*, como os conjuntos de dados MNIST (letras e números), Linnerud (medições fisiológicas de atletas), Breast Cancer Wisconsin (câncer de mama), Olivetti Faces (rostos humanos), dentre outros;
- Implementação da rede-em-chip nanoeletrônica completa via desenvolvimento de roteadores, elementos de memória e demais blocos digitais;
- Aplicação de demais transistores nanoeletrônicos disponibilizado pelo *Nanoscale Integration and Modeling (NIMO) Group*, como os modelos preditivos CMOS de 7nm, 10nm e 14nm, além dos transistores MOS para aplicações de alta performance (NIMO, 2012).

REFERÊNCIAS

AGARWAL, Ankur; SHANKAR, Ravi. Survey of Network on Chip (NoC) Architectures and Contributions. **Journal of Engineering, Computing and Architecture**, v. 3, jan. 2009.

AHMAD, Khurshid; SETHI, Muhammad. Review of Network on Chip Routing Algorithms. **EAI Endorsed Transactions on Context-aware Systems and Applications**, v. 7, p. 12, dez. 2020.

BINAS, Jonathan; NEIL, Dan; INDIVERI, Giacomo; LIU, Shih-Chii; PFEIFFER, Michael. Precise neural network computation with imprecise analog devices, p. 22, fev. 2020.

BINAS, Jonathan; NEIL, Daniel; INDIVERI, G.; LIU, Shih-Chii; PFEIFFER, Michael. Precise deep neural network computation on imprecise low-power analog hardware. **ArXiv**, abs/1606.07786, 2016.

BOND, Eric. Essentials of a Theory for How Brain Structure Contributes to the Substance of Consciousness. **World Journal of Neuroscience**, v. 12, p. 8–21, jan. 2022.

CHEN, Wei; QI, Dongchen; GAO, Xingyu; WEE, Andrew Thye Shen. Surface transfer doping of semiconductors. **Progress in Surface Science**, v. 84, n. 9, p. 279–321, 2009. ISSN 0079-6816.

DALLY, Willian James; TOWLES, Brian. **Principles and Practices of Interconnection**. 1. ed. San Francisco, CA: Morgan Kaufmann Publishers, 2004.

DONG, Yiping; LIN, Zhen; LI, Yan; WATANABE, Takahiro. High performance implementation of Neural Networks by networks on chip with 5-port 2-virtual channels. *In*: PROCEEDINGS of 2010 IEEE International Symposium on Circuits and Systems. [*S.l.*: *s.n.*], 2010. P. 381–384.

DUARTE, Renan Rodrigo. **Estudo comparativo entre semicondutores de silício e nitreto de gálio em circuitos de acionamento de leds**. Mar. 2018. Universidade Federal de Santa Maria, Santa Maria.

FÉ, Beatriz Oliveira Câmara da. **Roteador nanoeletrônico para redes-em-chip baseado em transistores monoelétron**. Mar. 2017. Universidade Nacional de Brasília, Brasília.

FISHER, R. A. The use of multiple measurements in taxonomic problems. **Annals of Eugenics**, v. 7, n. 2, p. 179–188, 1936. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1469-1809.1936.tb02137.x.

FISHER, R. A. **Iris**. [*S.l.*: *s.n.*], 1988. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C56C76.

GARDENIA. **Iris plants**. 2023. Disponível em: https://www.gardenia.net/. Acesso em: 5 jun. 2023.

HANSON, G. W. **Fundamentals of nanoelectronics**. 14. ed. Upper Saddle River, NJ: Pearson/Prentice Hall, 2008.

HAYKIN, S.S. **Neural Networks and Learning Machines**. [*S.l.*]: Pearson, 2009. (Pearson International Edition). ISBN 9780131293762.

JAEGER, Richard C.; BLALOCK, Travis N. **Microelectronic circuit design**. 5. ed. New York: McGraw-Hill Education, 2015.

KUNDU, Santanu; CHATTOPADHYAY, Santanu. **Network-on-Chip: the next generation of System-on-Chip integration**. 1. ed. Boca Raton: CRC Press, 2015.

KUROSE, Jim F.; ROSS, Keith W. **Redes de computadores e a Internet: uma abordagem top-down**. 6. ed. São Paulo: Pearson Education do Brasil, 2014.

MAHENDRA, Chawade; GAIKWAD, Mahendra; PATRIKAR, Rajendra. Review of XY Routing Algorithm for Network-on-Chip Architecture. **International Journal of Computer and Communication Technology**, out. 2016.

MEAD, C. Neuromorphic electronic systems. **Proceedings of the IEEE**, v. 78, n. 10, p. 1629–1636, 1990.

MOHSIN ABDULAZEEZ, Adnan; ZEEBAREE, Diyar; ZEBARI, Dilovan; JIJO, Bahzad. Machine Learning Classifiers Based Classification For IRIS Recognition. **Qubahan Academic Journal**, v. 1, mai. 2021.

MOORE, Gordon E. Cramming more components onto integrated circuits, Reprinted from Electronics, volume 38, number 8, April 19, 1965, pp.114 ff. **IEEE Solid-State Circuits Society Newsletter**, v. 11, n. 3, p. 33–35, 2006.

NELSON, Max; SHIPBAUGH, Calvin. **The potential of nanotechnology for molecular manufacturing**. 1. ed. [*S.l.*]: Pearson, 1995.

NIMO. **Predictive Technology Model (PTM)**. [*S.l.*: *s.n.*], 2012. http://ptm.asu.edu/. Acesso em 14 de Setembro de 2022.

OLIVEIRA FLORENTINO, Helenice de; FÁTIMA VILELA BISCARO, Adriana de; SOUZA PASSOS, José Raimundo de. Funções sigmoidais aplicadas na determinação da atividade metanogênica específica - AME. *In*: REV. Bras. Biom. São Paulo: [*s.n.*], 2010. P. 141–150.

PACHECO, César Augusto Rodrigues; PEREIRA, Natasha Sophie. Deep Learning conceitos e utilização nas diversas áreas do conhecimento. **Revista Ada Lovelace**, Oxford University Press, v. 2, n. 1, p. 34–49, 2018.

PINA, Kleber Vieira; PINTO, Luciano Rodrigues; MORATORI, Raquel Barbosa; SOUZA, Cristina Gomes de; BARBASTEFANO, Rafael Garcia. Nanotecnologia e nanobiotecnologia: estado da arte, perspectivas de inovação e investimentos. *In*: REVISTA Gestão Industrial. Ponta Grossa: [*s.n.*], 2006. P. 115–125.

PINTO, Joylin Priya; KELUR, Soumya; SHETTY, Jyothi. Iris Flower Species Identification Using Machine Learning Approach. *In*: 2018 4th International Conference for Convergence in Technology (I2CT). [*S.l.*: *s.n.*], 2018. P. 1–4.

RAJENDRAN, Bipin; SEBASTIAN, Abu; SCHMUKER, Michael; SRINIVASA, Narayan; ELEFTHERIOU, Evangelos. Low-Power Neuromorphic Hardware for Signal Processing Applications, jan. 2019.

RAMOS, JEAN. Redes neurais artificiais na classificação de frutos: cenário bidimensional. **Ciência e Agrotecnologia**, v. 27, abr. 2003.

RUMELHART; E., David; MCCLELLAND, James; L., James. **Parallel distributed processing: explorations in the microstructure of cognition. Volume 1. Foundations**. [*S.l.*: *s.n.*], jan. 1986.

RUSSELL, Stuart; NORVIG, Peter. **Artificial intelligence**. 3. ed. Rio de Janeiro: Elsevier, 2013.

SALMINEN, E; KULMALA, A; HÄMÄLÄINEN, Timo. Survey of networks-on-chip proposals. **OCP International Partnership**, 13 p, jan. 2009.

SANTOS PÊS, Beatriz dos. **Estudo sobre o desempenho de redes neurais nanoeletrônicas**. 2012. F. 90. Monografia (Monografia) – Universidade Nacional de Brasília, Brasília.

SCHUMAN, Catherine D.; POTOK, Thomas E.; PATTON, Robert M.; BIRDWELL, J. Douglas; DEAN, Mark E.; ROSE, Garrett S.; PLANK, James S. A Survey of Neuromorphic Computing and Neural Networks in Hardware. **CoRR**, abs/1705.06963, 2017. arXiv: 1705.06963.

SEGUNDO, Fábio Rafael. **Roteamento em redes tolerantes a atrasos e interrupções: uma abordagem baseada em redes neurais**. 2015. Tese (Doutorado) – Universidade Federal de Santa Catarina.

SRINIVASARAO, Tumma. Iris Flower Classification Using Machine Learning. v. 9, p. 2455–6211, mar. 2022.

STODOLNA, A. S.; ROUZÉE, A.; LÉPINE, F.; COHEN, S.; ROBICHEAUX, F.; GIJSBERTSEN, A.; JUNGMANN, J. H.; BORDAS, C.; VRAKKING, M. J. J. Hydrogen Atoms under Magnification: Direct Observation of the Nodal Structure of Stark States. **Phys. Rev. Lett.**, American Physical Society, v. 110, p. 213001, 21 mai. 2013.

SUSSILLO, David; ABBOTT, L. F. **Random Walk Initialization for Training Very Deep Feedforward Networks**. [*S.l.*: *s.n.*], 2015. arXiv: 1412.6558 [cs.NE].

TANIGUCHI, N. On the basic concept of nanotechnology. *In*: PROCEEDINGS of the International Conference on Production Engineering. Tokyo: [*s.n.*], 1974. P. 18–23.

WAKERLY, John F. **Digital Design: Principles and Practice**. 5. ed. [*S.l.*]: Pearson, 2018.

WANG, Ting; SU, Zhiyang; XIA, Yu; QIN, Bo; HAMDI, Mounir. NovaCube: A low latency Torus-based network architecture for data centers. **2014 IEEE Global Communications Conference, GLOBECOM 2014**, p. 2252–2257, fev. 2015.

WUERGES, Artur Filipe Ewald; BORBA, José Alonso. Redes Neurais, lógica nebulosa e algoritmos genéticos: aplicações e possibilidades em finanças e contabilidade. **Journal of Information Systems and Technology Management**, 2010.

YANG, Jia-Qin; WANG, Ruopeng; REN, Yi; MAO, Jingyu; WANG, Zhanpeng; ZHOU, Ye; HAN, Su-Ting. Neuromorphic Engineering: From Biological to Spike-Based Hardware Nervous Systems. **Advanced Materials**, v. 32, p. 2003610, dez. 2020.

YIPING, Dong; TAKAHIRO, Watanabe. High performance NoC architecture for two hidden layers BP Neural Network. *In*: 2008 International SoC Design Conference. [*S.l.*: *s.n.*], 2008. v. 01, p. i-269-i–272.

YOUNG, Aaron R.; DEAN, Mark E.; PLANK, James S.; S. ROSE, Garrett. A Review of Spiking Neuromorphic Hardware Communication Systems. **IEEE Access**, v. 7, p. 135606–135620, 2019.

ANEXO A – Código SPICE do transistor CMOS 16nm LP

Este anexo apresenta o código SPICE do transistor de 16nm de baixa potência (16nm LP) disponibilizado pelo *Nanoscale Integration and Modeling (NIMO) Group* (Grupo de Integração em Nanoescala e Modelagem) da *Arizona State University* (Universidade Estadual do Arizona) (NIMO, 2012).

```
* PTM Low Power 16nm Metal Gate / High-K / Strained-Si
* nominal Vdd = 0.9V

.model   nmos   nmos   level = 54

+version = 4.0       binunit = 1         paramchk= 1         mobmod  = 0
+capmod  = 2         igcmod  = 1         igbmod  = 1         geomod  = 1
+diomod  = 1         rdsmod  = 0         rbodymod= 1         rgatemod= 1
+permod  = 1         acnqsmod= 0         trnqsmod= 0

+tnom    = 27        toxe = 1.2e-009     toxp = 9e-010       toxm    = 1.2e-009
+dtox    = 3e-010    epsrox  = 3.9       wint    = 5e-009    lint    = 0
+ll      = 0         wl      = 0         lln     = 1         wln     = 1
+lw      = 0         ww      = 0         lwn     = 1         wwn     = 1
+lwl     = 0         wwl     = 0         xpart   = 0         toxref  = 1.2e-009

+vth0    = 0.68191   k1      = 0.4           k2      = 0         k3      = 0
+k3b     = 0         w0      = 2.5e-006      dvt0    = 1         dvt1    = 2
+dvt2    = 0         dvt0w   = 0         dvt1w   = 0         dvt2w   = 0
+dsub    = 0.1       minv    = 0.05      voffl   = 0         dvtp0   = 1e-011
+dvtp1   = 0.1       lpe0    = 0         lpeb    = 0         xj      = 5e-009
+ngate   = 1e+023    ndep    = 7e+018    nsd     = 2e+020    phin    = 0
+cdsc    = 0         cdscb   = 0         cdscd   = 0         cit     = 0
+voff    = -0.1014   nfactor = 1.6       eta0    = 0.0095    etab    = 0
+vfb     = -0.55     u0      = 0.028     ua      = 6e-010    ub      = 1.2e-018
+uc      = 0         vsat    = 200000    a0      = 1         ags     = 0
+a1      = 0         a2      = 1         b0      = 0         b1      = 0
+keta    = 0.04      dwg     = 0         dwb     = 0         pclm    = 0.02
+pdiblc1 = 0.001     pdiblc2 = 0.001     pdiblcb = -0.005    drout   = 0.5
+pvag    = 1e-020    delta   = 0.01      pscbe1  = 8.14e+008 pscbe2  = 1e-007
+fprout  = 0.2       pdits   = 0.01      pditsd  = 0.23      pditsl  = 2300000
+rsh     = 5         rdsw    = 170       rsw     = 75        rdw     = 75
+rdswmin = 0         rdwmin  = 0         rswmin  = 0         prwg    = 0
+prwb    = 0         wr      = 1         alpha0  = 0.074     alpha1  = 0.005
+beta0   = 30        agidl   = 0.0002    bgidl   = 2.1e+009  cgidl   = 0.0002
+egidl   = 0.8       aigbacc = 0.012     bigbacc = 0.0028    cigbacc = 0.002
+nigbacc = 1         aigbinv = 0.014     bigbinv = 0.004     cigbinv = 0.004
+eigbinv = 1.1       nigbinv = 3         aigc    = 0.015211  bigc    = 0.0027432
+cigc    = 0.002     aigsd   = 0.015211  bigsd   = 0.0027432 cigsd   = 0.002
```

```
+nigc     = 1          poxedge = 1        pigcd    = 1          ntox     = 1
+xrcrg1   = 12         xrcrg2  = 5

+cgso     = 5e-011     cgdo    = 5e-011   cgbo     = 2.56e-011  cgdl = 2.653e-010

+cgsl     = 2.653e-010 ckappas = 0.03     ckappad = 0.03       acde     = 1
+moin     = 15         noff    = 0.9      voffcv  = 0.02

+kt1      = -0.11      kt1l = 0           kt2      = 0.022      ute = -1.5
+ua1      = 4.31e-009  ub1  = 7.61e-018   uc1      = -5.6e-011  prt = 0
+at       = 33000

+fnoimod = 1                tnoimod = 0
+jss      = 0.0001     jsws    = 1e-011   jswgs    = 1e-010     njs      = 1
+ijthsfwd= 0.01       ijthsrev= 0.001    bvs      = 10         xjbvs    = 1
+jsd      = 0.0001     jswd    = 1e-011   jswgd    = 1e-010     njd      = 1
+ijthdfwd= 0.01       ijthdrev= 0.001    bvd      = 10         xjbvd    = 1
+pbs      = 1          cjs     = 0.0005   mjs      = 0.5        pbsws    = 1
+cjsws    = 5e-010     mjsws   = 0.33     pbswgs   = 1          cjswgs   = 3e-010
+mjswgs   = 0.33       pbd     = 1        cjd      = 0.0005     mjd      = 0.5
+pbswd    = 1          cjswd   = 5e-010   mjswd    = 0.33       pbswgd   = 1
+cjswgd   = 5e-010     mjswgd  = 0.33     tpb      = 0.005      tcj      = 0.001
+tpbsw    = 0.005      tcjsw   = 0.001    tpbswg   = 0.005      tcjswg   = 0.001
+xtis     = 3          xtid    = 3
+dmcg     = 0          dmci    = 0        dmdg     = 0          dmcgt    = 0
+dwj      = 0          xgw     = 0        xgl      = 0

+rshg     = 0.4        gbmin   = 1e-010   rbpb     = 5          rbpd     = 15
+rbps     = 15         rbdb    = 15       rbsb     = 15         ngcon    = 1

.model   pmos   pmos   level = 54

+version = 4.0        binunit = 1        paramchk= 1           mobmod   = 0
+capmod  = 2          igcmod  = 1        igbmod   = 1          geomod   = 1
+diomod  = 1          rdsmod  = 0        rbodymod= 1           rgatemod= 1
+permod  = 1          acnqsmod= 0        trnqsmod= 0

+tnom     = 27    toxe     = 1.22e-009   toxp     = 9e-010     toxm     = 1.22e-009
+dtox     = 3.2e-010 epsrox  = 3.9       wint     = 5e-009     lint     = 8e-010
+ll       = 0     wl       = 0           lln      = 1          wln      = 1
+lw       = 0     ww       = 0           lwn      = 1          wwn      = 1
+lwl      = 0     wwl      = 0           xpart    = 0          toxref   = 1.22e-009

+vth0     = -0.6862   k1   = 0.4         k2       = -0.01      k3       = 0
+k3b      = 0         w0   = 2.5e-006    dvt0     = 1          dvt1     = 2
+dvt2     = -0.032    dvt0w  = 0         dvt1w    = 0          dvt2w    = 0
+dsub     = 0.1       minv   = 0.05      voffl    = 0          dvtp0    = 1e-011
```

```
+dvtp1  = 0.05     lpe0   = 0        lpeb   = 0        xj     = 7.2e-009
+ngate  = 1e+023   ndep   = 4.4e+018 nsd    = 2e+020   phin   = 0
+cdsc   = 0        cdscb  = 0        cdscd  = 0        cit    = 0
+voff   = -0.08    nfactor = 1.8     eta0   = 0.0095   etab   = 0
+vfb    = 0.55     u0     = 0.0075   ua     = 2e-009   ub     = 5e-019
+uc     = 0        vsat   = 195000   a0     = 1        ags    = 1e-020
+a1     = 0        a2     = 1        b0     = 0        b1     = 0
+keta   = -0.047   dwg    = 0        dwb    = 0        pclm   = 0.12
+pdiblc1 = 0.001   pdiblc2 = 0.001   pdiblcb = 3.4e-008 drout  = 0.56
+pvag   = 1e-020   delta  = 0.01     pscbe1 = 8.14e+008 pscbe2 = 9.58e-007
+fprout = 0.2      pdits  = 0.08     pditsd = 0.23      pditsl = 2300000
+rsh    = 5        rdsw   = 220      rsw    = 72.5      rdw    = 72.5
+rdswmin = 0       rdwmin = 0        rswmin = 0         prwg   = 0
+prwb   = 0        wr     = 1        alpha0 = 0.074     alpha1 = 0.005
+beta0  = 30       agidl  = 0.0002   bgidl  = 2.1e+009  cgidl  = 0.0002
+egidl  = 0.8      aigbacc = 0.012   bigbacc = 0.0028   cigbacc = 0.002
+nigbacc = 1       aigbinv = 0.014   bigbinv = 0.004    cigbinv = 0.004
+eigbinv = 1.1     nigbinv = 3       aigc   = 0.0097    bigc   = 0.00125
+cigc   = 0.0008   aigsd  = 0.0115   bigsd  = 0.00125   cigsd  = 0.0008
+nigc   = 1        poxedge = 1       pigcd  = 1         ntox   = 1
+xrcrg1 = 12       xrcrg2 = 5

+cgso   = 5e-011   cgdo   = 5e-011 cgbo   = 2.56e-011 cgdl  = 2.653e-010
+cgsl   = 2.653e-010 ckappas = 0.03 ckappad = 0.03     acde  = 1
+moin   = 15       noff   = 0.9    voffcv = 0.02

+kt1    = -0.11     kt1l   = 0      kt2    = 0.022     ute    = -1.5
+ua1    = 4.31e-009 ub1    = 7.61e-018 uc1 = -5.6e-011 prt    = 0
+at     = 33000

+fnoimod = 1              tnoimod = 0
+jss    = 0.0001   jsws   = 1e-011  jswgs  = 1e-010   njs    = 1
+ijthsfwd= 0.01    ijthsrev= 0.001  bvs    = 10       xjbvs  = 1
+jsd    = 0.0001   jswd   = 1e-011  jswgd  = 1e-010   njd    = 1
+ijthdfwd= 0.01    ijthdrev= 0.001  bvd    = 10       xjbvd  = 1
+pbs    = 1        cjs    = 0.0005  mjs    = 0.5      pbsws  = 1
+cjsws  = 5e-010   mjsws  = 0.33    pbswgs = 1        cjswgs = 3e-010
+mjswgs = 0.33     pbd    = 1       cjd    = 0.0005   mjd    = 0.5
+pbswd  = 1        cjswd  = 5e-010  mjswd  = 0.33     pbswgd = 1
+cjswgd = 5e-010   mjswgd = 0.33    tpb    = 0.005    tcj    = 0.001
+tpbsw  = 0.005    tcjsw  = 0.001   tpbswg = 0.005    tcjswg = 0.001
+xtis   = 3        xtid   = 3

+dmcg   = 0        dmci   = 0       dmdg   = 0        dmcgt  = 0
+dwj    = 0        xgw    = 0       xgl    = 0
+rshg   = 0.4      gbmin  = 1e-010  rbpb   = 5        rbpd   = 15
+rbps   = 15       rbdb   = 15      rbsb   = 15       ngcon  = 1
```

ANEXO B – Conjunto de dados *Iris* de Fisher (flor *Iris*)

O presente anexo exibe o conjunto de dados *Iris* de Fisher (*Iris flower data set*), o qual encontra-se disponível no repositório de aprendizado de máquina do *Center for Machine Learning and Intelligent Systems* (Centro de Aprendizado de Máquina e Sistemas Inteligentes) da UCI - University of California (Universidade da Califórnia) (FISHER, 1988).

Item	Comp. Sépala	Larg. Sépala	Comp. Pétala	Larg. Pétala	Espécie
1	5.1	3.5	1.4	0.2	*I. setosa*
2	4.9	3.0	1.4	0.2	*I. setosa*
3	4.7	3.2	1.3	0.2	*I. setosa*
4	4.6	3.1	1.5	0.2	*I. setosa*
5	5.0	3.6	1.4	0.3	*I. setosa*
6	5.4	3.9	1.7	0.4	*I. setosa*
7	4.6	3.4	1.4	0.3	*I. setosa*
8	5.0	3.4	1.5	0.2	*I. setosa*
9	4.4	2.9	1.4	0.2	*I. setosa*
10	4.9	3.1	1.5	0.1	*I. setosa*
11	5.4	3.7	1.5	0.2	*I. setosa*
12	4.8	3.4	1.6	0.2	*I. setosa*
13	4.8	3.0	1.4	0.1	*I. setosa*
14	4.3	3.0	1.1	0.1	*I. setosa*
15	5.8	4.0	1.2	0.2	*I. setosa*
16	5.7	4.4	1.5	0.4	*I. setosa*
17	5.4	3.9	1.3	0.4	*I. setosa*
18	5.1	3.5	1.4	0.3	*I. setosa*
19	5.7	3.8	1.7	0.3	*I. setosa*
20	5.1	3.8	1.5	0.3	*I. setosa*
21	5.4	3.4	1.7	0.2	*I. setosa*
22	5.1	3.7	1.5	0.4	*I. setosa*
23	4.6	3.6	1.0	0.2	*I. setosa*
24	5.1	3.3	1.7	0.5	*I. setosa*
25	4.8	3.4	1.9	0.2	*I. setosa*
26	5.0	3.0	1.6	0.2	*I. setosa*
27	5.0	3.4	1.6	0.4	*I. setosa*
28	5.2	3.5	1.5	0.2	*I. setosa*
29	5.2	3.4	1.4	0.2	*I. setosa*

30	4.7	3.2	1.6	0.2	*I. setosa*
31	4.8	3.1	1.6	0.2	*I. setosa*
32	5.4	3.4	1.5	0.4	*I. setosa*
33	5.2	4.1	1.5	0.1	*I. setosa*
34	5.5	4.2	1.4	0.2	*I. setosa*
35	4.9	3.1	1.5	0.2	*I. setosa*
36	5.0	3.2	1.2	0.2	*I. setosa*
37	5.5	3.5	1.3	0.2	*I. setosa*
38	4.9	3.6	1.4	0.1	*I. setosa*
39	4.4	3.0	1.3	0.2	*I. setosa*
40	5.1	3.4	1.5	0.2	*I. setosa*
41	5.0	3.5	1.3	0.3	*I. setosa*
42	4.5	2.3	1.3	0.3	*I. setosa*
43	4.4	3.2	1.3	0.2	*I. setosa*
44	5.0	3.5	1.6	0.6	*I. setosa*
45	5.1	3.8	1.9	0.4	*I. setosa*
46	4.8	3.0	1.4	0.3	*I. setosa*
47	5.1	3.8	1.6	0.2	*I. setosa*
48	4.6	3.2	1.4	0.2	*I. setosa*
49	5.3	3.7	1.5	0.2	*I. setosa*
50	5.0	3.3	1.4	0.2	*I. setosa*
51	7.0	3.2	4.7	1.4	*I. versicolor*
52	6.4	3.2	4.5	1.5	*I. versicolor*
53	6.9	3.1	4.9	1.5	*I. versicolor*
54	5.5	2.3	4.0	1.3	*I. versicolor*
55	6.5	2.8	4.6	1.5	*I. versicolor*
56	5.7	2.8	4.5	1.3	*I. versicolor*
57	6.3	3.3	4.7	1.6	*I. versicolor*
58	4.9	2.4	3.3	1.0	*I. versicolor*
59	6.6	2.9	4.6	1.3	*I. versicolor*
60	5.2	2.7	3.9	1.4	*I. versicolor*
61	5.0	2.0	3.5	1.0	*I. versicolor*
62	5.9	3.0	4.2	1.5	*I. versicolor*
63	6.0	2.2	4.0	1.0	*I. versicolor*
64	6.1	2.9	4.7	1.4	*I. versicolor*
65	5.6	2.9	3.6	1.3	*I. versicolor*
66	6.7	3.1	4.4	1.4	*I. versicolor*
67	5.6	3.0	4.5	1.5	*I. versicolor*
68	5.8	2.7	4.1	1.0	*I. versicolor*

69	6.2	2.2	4.5	1.5	*I. versicolor*
70	5.6	2.5	3.9	1.1	*I. versicolor*
71	5.9	3.2	4.8	1.8	*I. versicolor*
72	6.1	2.8	4.0	1.3	*I. versicolor*
73	6.3	2.5	4.9	1.5	*I. versicolor*
74	6.1	2.8	4.7	1.2	*I. versicolor*
75	6.4	2.9	4.3	1.3	*I. versicolor*
76	6.6	3.0	4.4	1.4	*I. versicolor*
77	6.8	2.8	4.8	1.4	*I. versicolor*
78	6.7	3.0	5.0	1.7	*I. versicolor*
79	6.0	2.9	4.5	1.5	*I. versicolor*
80	5.7	2.6	3.5	1.0	*I. versicolor*
81	5.5	2.4	3.8	1.1	*I. versicolor*
82	5.5	2.4	3.7	1.0	*I. versicolor*
83	5.8	2.7	3.9	1.2	*I. versicolor*
84	6.0	2.7	5.1	1.6	*I. versicolor*
85	5.4	3.0	4.5	1.5	*I. versicolor*
86	6.0	3.4	4.5	1.6	*I. versicolor*
87	6.7	3.1	4.7	1.5	*I. versicolor*
88	6.3	2.3	4.4	1.3	*I. versicolor*
89	5.6	3.0	4.1	1.3	*I. versicolor*
90	5.5	2.5	4.0	1.3	*I. versicolor*
91	5.5	2.6	4.4	1.2	*I. versicolor*
92	6.1	3.0	4.6	1.4	*I. versicolor*
93	5.8	2.6	4.0	1.2	*I. versicolor*
94	5.0	2.3	3.3	1.0	*I. versicolor*
95	5.6	2.7	4.2	1.3	*I. versicolor*
96	5.7	3.0	4.2	1.2	*I. versicolor*
97	5.7	2.9	4.2	1.3	*I. versicolor*
98	6.2	2.9	4.3	1.3	*I. versicolor*
99	5.1	2.5	3.0	1.1	*I. versicolor*
100	5.7	2.8	4.1	1.3	*I. versicolor*
101	6.3	3.3	6.0	2.5	*I. virginica*
102	5.8	2.7	5.1	1.9	*I. virginica*
103	7.1	3.0	5.9	2.1	*I. virginica*
104	6.3	2.9	5.6	1.8	*I. virginica*
105	6.5	3.0	5.8	2.2	*I. virginica*
106	7.6	3.0	6.6	2.1	*I. virginica*
107	4.9	2.5	4.5	1.7	*I. virginica*

108	7.3	2.9	6.3	1.8	*I. virginica*
109	6.7	2.5	5.8	1.8	*I. virginica*
110	7.2	3.6	6.1	2.5	*I. virginica*
111	6.5	3.2	5.1	2.0	*I. virginica*
112	6.4	2.7	5.3	1.9	*I. virginica*
113	6.8	3.0	5.5	2.1	*I. virginica*
114	5.7	2.5	5.0	2.0	*I. virginica*
115	5.8	2.8	5.1	2.4	*I. virginica*
116	6.4	3.2	5.3	2.3	*I. virginica*
117	6.5	3.0	5.5	1.8	*I. virginica*
118	7.7	3.8	6.7	2.2	*I. virginica*
119	7.7	2.6	6.9	2.3	*I. virginica*
120	6.0	2.2	5.0	1.5	*I. virginica*
121	6.9	3.2	5.7	2.3	*I. virginica*
122	5.6	2.8	4.9	2.0	*I. virginica*
123	7.7	2.8	6.7	2.0	*I. virginica*
124	6.3	2.7	4.9	1.8	*I. virginica*
125	6.7	3.3	5.7	2.1	*I. virginica*
126	7.2	3.2	6.0	1.8	*I. virginica*
127	6.2	2.8	4.8	1.8	*I. virginica*
128	6.1	3.0	4.9	1.8	*I. virginica*
129	6.4	2.8	5.6	2.1	*I. virginica*
130	7.2	3.0	5.8	1.6	*I. virginica*
131	7.4	2.8	6.1	1.9	*I. virginica*
132	7.9	3.8	6.4	2.0	*I. virginica*
133	6.4	2.8	5.6	2.2	*I. virginica*
134	6.3	2.8	5.1	1.5	*I. virginica*
135	6.1	2.6	5.6	1.4	*I. virginica*
136	7.7	3.0	6.1	2.3	*I. virginica*
137	6.3	3.4	5.6	2.4	*I. virginica*
138	6.4	3.1	5.5	1.8	*I. virginica*
139	6.0	3.0	4.8	1.8	*I. virginica*
140	6.9	3.1	5.4	2.1	*I. virginica*
141	6.7	3.1	5.6	2.4	*I. virginica*
142	6.9	3.1	5.1	2.3	*I. virginica*
143	5.8	2.7	5.1	1.9	*I. virginica*
144	6.8	3.2	5.9	2.3	*I. virginica*
145	6.7	3.3	5.7	2.5	*I. virginica*
146	6.7	3.0	5.2	2.3	*I. virginica*

147	6.3	2.5	5.0	1.9	*I. virginica*
148	6.5	3.0	5.2	2.0	*I. virginica*
149	6.2	3.4	5.4	2.3	*I. virginica*
150	5.9	3.0	5.1	1.8	*I. virginica*