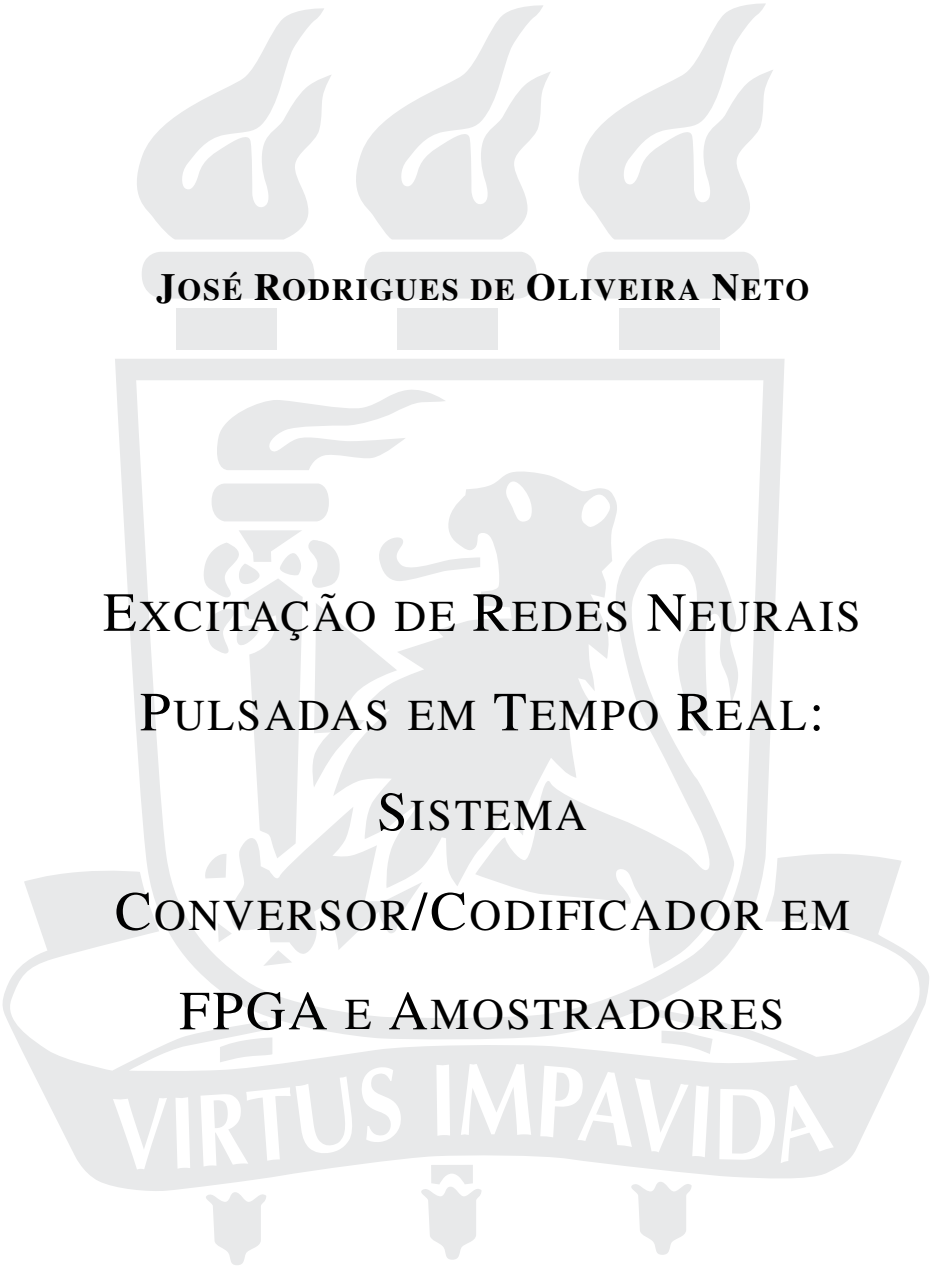
UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE TECNOLOGIA E GEOCIÊNCIAS
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA



JOSÉ RODRIGUES DE OLIVEIRA NETO

EXCITAÇÃO DE REDES NEURAIS PULSADAS EM TEMPO REAL: SISTEMA CONVERSOR/CODIFICADOR EM FPGA E AMOSTRADORES

RECIFE, 2015.

JOSÉ RODRIGUES DE OLIVEIRA NETO

EXCITAÇÃO DE REDES NEURAIS PULSADAS EM TEMPO REAL: SISTEMA CONVERSOR/CODIFICADOR EM FPGA E AMOSTRADORES

Dissertação submetida ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Pernambuco como parte dos requisitos para obtenção do grau de **Mestre em Engenharia Elétrica**

ORIENTADOR: PROF. DR. JOÃO HENRIQUE RANHEL RIBEIRO

COORIENTADOR: PROF. DR. JOÃO PAULO CERQUINHO CAJUEIRO

Recife, 2015.

Catalogação na fonte
Bibliotecária Margareth Malta, CRB-4 / 1198

O48e Oliveira Neto, José Rodrigues de.

Excitação de redes neurais pulsadas em tempo real: sistema conversor/codificador em FPGA e amostradores / José Rodrigues de Oliveira Neto. - Recife: O Autor, 2015.

135 folhas, il., gráfs., tabs.

Orientador: Prof. Dr. João Henrique Ranhel Ribeiro.

Coorientador: Prof. Dr. João Paulo Cerquinho Cajueiro.

Dissertação (Mestrado) – Universidade Federal de Pernambuco. CTG. Programa de Pós-Graduação em Engenharia Elétrica, 2015.

Inclui Referências e Apêndices.

1. Engenharia Elétrica. 2. Redes neurais pulsadas. 3. FPGA. 4. Computação em assembleias neurais. 5. Sistemas embarcados. I. Ribeiro, João Henrique Ranhel. (Orientador). II. Cajueiro, João Paulo Cerquinho. (Coorientador). III. Título.

UFPE

621.3 CDD (22. ed.) BCTG/2015-216

PPGEE
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA
UFPE

Ata de defesa da 264ª Dissertação de Mestrado do Programa de Pós-Graduação em Engenharia Elétrica do Centro de Tecnologia e Geociências da Universidade Federal de Pernambuco, no dia 28 de julho de 2015.

Aos 28 (vinte e oito) dias do mês de julho de dois mil e quinze (2015), às dez horas, no Departamento de Eletrônica e Sistemas do Centro de Tecnologia e Geociências da Universidade Federal de Pernambuco, em sessão pública, teve início a defesa da dissertação intitulada *"Excitação de Redes Neurais Pulsadas em Tempo Real: Sistema Conversor/Codificador em FPGA e Amostradores"*, do aluno **JOSÉ RODRIGUES DE OLIVEIRA NETO**, na área de concentração em Eletrônica, sob a orientação do prof. João Henrique Ranhel Ribeiro. O mestrando cumpriu todos os demais requisitos regimentais para a obtenção do grau de MESTRE em Engenharia Elétrica. A Banca Examinadora foi indicada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica no dia primeiro de junho de dois mil e quinze, na sua 2ª Reunião Ordinária e homologada pela Diretoria de Pós-Graduação através do processo Nº 23076.029102/2015-13 em dez de julho do ano de dois mil e quinze composta pelos professores: Orientador e 1º Examinador Externo: João Henrique Ranhel Ribeiro, DES/UFPE; 2º Examinador Interno: Fernanda Maria Ribeiro de Alencar, DES/UFPE; 3º Examinador Externo: Fernando José Ribeiro Sales, DEB/UFPE; Coorientador e 4º Examinador Externo: João Paulo Cerquinho Cajueiro; Suplente Interno: Marco Aurélio Benedetti Rodrigues e Suplente Externo: Patrícia Silva Lessa, DES/UFPE. Após cumpridas as formalidades, o candidato foi convidado a discorrer sobre o conteúdo da dissertação. Concluída a explanação, o candidato foi arguído pela banca examinadora que, em seguida, reuniu-se para deliberar e conceder ao mesmo a menção de **APROVADO** da referida dissertação. E para constar, eu, Andréa **Tenório** Pinto, secretária do Programa de Pós-Graduação em Engenharia Elétrica, lavrei a presente ata que vai por mim assinada e pelos membros da comissão examinadora. Recife, 28 de julho de 2015.

ANDRÉA TENÓRIO PINTO _____
Secretária do PPGEE

Banca Examinadora:

JOÃO HENRRIQUE RANHEL RIBEIRO _____
Orientador e Membro Titular Interno

FERNANDA MARIA RIBEIRO DE ALENCAR _____
Membro Titular Interno

FERNANDO JOSÉ RIBEIRO SALES _____
Membro Titular Externo

JOÃO PAULO CERQUINHO CAJUEIRO _____
Coorientador e Membro Titular Externo

Aos meus pais.

AGRADECIMENTOS

Aos meus pais, Jó e Valderio, por todo o incentivo, educação, carinho e principalmente amor que sempre demonstraram. Aos meus avós que sempre foram como segundos pais e que gostaria que pudessem estar ainda aqui. Aos meus irmãos Walber e Walderio pelo companheirismo e amizade que nos acompanha desde sempre. À Roberta por me acompanhar em tudo nesses dois últimos anos, fazendo com que as lutas desse trabalho se confundam um pouco com nossa história. A toda a minha família: pais, avós, irmãos, tios, tias, primos, primos-irmãos (Mika e Phillipe), eu os amo muito.
Aos meus amigos distantes, principalmente a Thiego e Rafael, que mesmo distantes fazem questão de permanecerem presentes. Aos amigos da graduação e mestrado que continuaram através dos anos: Ian, Camila, Diego, Vítor, Vanine, Thiago, Kleber, Clarissa, Michelle, Zéca,...
Ao meu orientador Prof. João Ranhel por todos os ensinamentos, paciência, incentivos, cobranças e pela oportunidade de aprender com ele durante esses dois anos. Ao meu coorientador Prof. João Paulo, por acompanhar minha trajetória (me aguentar) desde o primeiro período da graduação, sempre me ensinando, orientando e servindo de inspiração.
Aos professores e colaboradores do BINAC, que influenciaram e ajudaram diretamente tantos aos resultados desse trabalho, quanto a minha formação, em especial a Felipe, Rafael e Clayton, que ajudaram diretamente neste trabalho.
A todos os professores do mestrado, graduação, ensino fundamental e médio, responsáveis por minha formação.
Ao CNPq, pelo suporte financeiro a este trabalho.

JOSÉ RODRIGUES DE OLIVEIRA NETO

Universidade Federal de Pernambuco

Resumo

O presente trabalho descreve a investigação e desenvolvimento de soluções para excitação Redes Neurais Pulsadas de tempo real a partir de grandezas físicas transduzidas e sinais simulados. Para isso foi desenvolvido um hardware dedicado de baixo custo capaz de transformar dados em trens de *spikes*, que são processados por essas redes. O sistema visa converter sinais digitais em *spikes* de neurônios artificiais, que são pulsos de 1 ms de duração. O sistema ainda pode organizar neurônios que disparam conjuntamente, a fim de gerar os três códigos neurais mais importantes descritos na literatura da neurociência: codificação por taxa de disparos, codificação por populações e codificação temporal. São descritas ainda duas topologias de amostradores (*samplers*) que discretizam representações na forma de populações neurais, que devem ser processadas segundo a abordagem Computação por Assembleias Neurais. Uma das topologias recolhe amostras na forma de populações de neurônios ativos durante um período definido (codificação por população), enquanto a outra recolhe amostras baseada na diferença temporal entre *spikes* (codificação temporal). Os sinais resultantes da amostragem podem ser utilizados internamente na rede como representações discretas de informações. Os sinais amostrados podem ainda ser utilizados como entradas de circuitos de tomada de decisão, cuja descrição das características e simulações também é parte deste trabalho.

Palavras-chave: Redes Neurais Pulsadas. FPGA. Computação em Assembleias Neurais. Sistemas Embarcados.

Abstract

This work describes the research and development of solutions for excitement of real-time Spiking Neural Networks from transduced physical quantities and simulated signals. For this it developed a dedicated low cost hardware able to turn data into spike trains, which are processed by these networks. The system aims to convert digital signals into spikes of artificial neurons, which are pulses of 1 ms. The system can even arrange neurons that fire together to generate the three most important neural codes described in the neuroscience literature: rate coding, populations coding and temporal coding. Two topologies of samplers are described; these topologies discretize representations in the form of neural populations that should be processed according to Neural Assembly Computing approach. One of these topologies collects samples as populations of neurons active during a defined period (population coding), while the other topology collects samples based on the time difference between spikes (temporal coding). The signals resulting from the sample can be used internally in the network as discrete representations of information. The sampled signals may also be used as inputs of decision-making circuits, the description of the characteristics and simulation of these circuits is also part of this work.

Keywords: Spiking Neural Networks. FPGA. Neural Assembly Computing. Embedded Systems.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ADALINE *ADAptive LInear NEuron*

ADC *Analog-to-Digital Converter*

CB *Control Block*

CC *Comparator Circuit*

ClkC *Clock Control*

CM *Counter Memory*

DAC *Digital-to-Analog Converter*

EDA *Electronic Design Automation*

EPSP *Excitatory Post-Synaptic Potential*

FLOPS *FLoating-point Operations Per Second*

FPGA *Field Programmable Gate Array*

GCCst Gerador, Conversor e Codificador de Trens de *Spikes*

GMPc Guanosina Monofosfato cíclica

HDL *Hardware Description Language*

IA Inteligencia Artificial

IM *Input Memory*

InC *Input Circuit*

IPSP *Inhibitory Post-Synaptic Potential*

IZH *Izhikevich's Simple Model*

LAB *Logic Array Block*

LE *Logic Elements*

LI&F *Leaky Integrated and Fire*

M&P *MacCulloch* e *Pitts*

MCLK *Main Clock*

NAC *Neural Assembly Computing*

OutC *Output Circuit*

RNA Rede Neural Artificial

SAH *Sample-And-Hold*

SAR *Successive Approximation Register*

SM *Spike Memory*

SNC Sistema Nervoso Central

SNN *Spiking Neural Netwok*

SoC *System on Chip*

TM *Threshold Memory*

UC *Update Circuit*

SUMÁRIO

CAPÍTULO 1

INTRODUÇÃO

1.1 MOTIVAÇÃO E CONTEXTO

REDE Neural Pulsada (SNN - *Spiking Neural Netwok*) [MAASS, 1997] é a mais nova geração de Rede Neural Artificial (RNA) [ROJAS, 1996] dentro do amplo campo da Inteligência Artificial. De acordo com *Maass* [MAASS, 1997], as redes pulsadas formam a terceira geração de RNAs. Nesta classificação, a primeira geração de RNA pode ser considerada a dos neurônios criados por *McCulloch* e *Pitts* em 1943 [MCCULLOCH and PITTS, 1943]. A segunda geração começou em meados dos anos 1980, com as redes recorrentes, os mapas auto-organizados e os algoritmos de retropropagação dos erros [HOPFIELD, 1982], [KOHONEN, 1988], [RUMELHART et al., 1988]. Na primeira geração os neurônios geravam saídas digitais, enquanto na segunda, com o modelo perceptron e saída sigmoidal, os neurônios geravam números reais como saída. Ao contrário do que ocorre nas gerações anteriores de RNAs, nas SNNs os neurônios geram pulsos como saídas. Neurônios que emitem pulsos em vez de valores numéricos de saída imitam melhor o comportamento dos neurônios biológicos [PAUGAM-MOISY and BOHTE, 2010], [GHOSH-DASTIDAR and ADELI, 2009], [MAASS, 1997]. Algumas redes pulsadas levam ainda em conta o atraso na propagação dos sinais pela rede.

Os neurônios biológicos geram potenciais de ação, ou *spikes*. Os potenciais de ação são impulsos elétricos que duram aproximadamente um milissegundo (1 ms) [KANDEL et al., 2000]. O potencial de ação é uma maneira econômica de codificar informações, pois o neurônio apenas gasta energia enquanto está disparando [PAPROCKI et al., 2013]. Além disso, o sincronismo entre *spikes* garante baixo consumo de energia em sistemas neurais pulsados [XING et al., 2012].

A neurociência ainda estuda como o processamento de informações e computação neural ocorrem [BRETTE, 2012]. Por décadas, pensou-se que os neurônios representavam informação e computação por meio da taxa de disparo neural [LEVINE, 2007], [ADRIAN and ZOTTERMAN, 1926]. Mas evidências biológicas empíricas mostraram que o tempo preciso em que ocorre o potencial de ação desempenha um papel importante na representação e computação nos cérebros. Na verdade, muitas respostas comportamentais são concluídas em um tempo que torna inviável calcular a média de disparo do neurônio, como ocorre com circuitos relacionados a visão [VANRULLEN et al., 2005]. Como consequência dessas novas descobertas da neurociência, vários pesquisadores voltaram seus interesses para SNNs, a terceira geração das RNAs [GHOSH-DASTIDAR and ADELI, 2009], [PAUGAM-MOISY and BOHTE, 2010]. São exemplos de abordagens das SNNs: a *Reservoir Computing* [MAASS et al., 2002], [JAEGER, 2001] e a Computação por Assembleias Neurais (NAC - *Neural Assembly Computing*) [RANHEL, 2012c].

SNN usa um formato digital do potencial de ação do neurônio e utiliza a relação temporal entre esses *spikes* para representar informações. Mais que isso, sistemas nervosos pulsados usam precisão temporal e inter-relação entre *spikes* para processar transformações sobre as informações representadas dessa forma. Este ambiente é capaz de lidar com a seleção de entradas, consolidação e combinação de informações aprendidas, informação vinculada a conjuntos de células, entre outros [BUZSÁKI and DRAGUHN, 2004]. Em razão de suas características operacionais, SNNs artificiais requerem *spike trains*, ou trens de *spikes*, como entrada para operar adequadamente [GERSTNER and KISTLER, 2002], [BROWN et al., 2004]. Assim, trens de *spikes* devem ser apresentados à SNN para que a rede os processe.

Como as SNNs são ativadas por *spikes*, interfaces são necessárias a fim de converter os fenômenos físicos para trens de *spikes*, antes de alimentar as SNNs que operam em tempo real. Por outro lado, pode-se também desejar que uma SNN processe dados artificiais, como números gerados em computadores. Ao converter informação em trens de *spikes*, tais interfaces representam as informações convertidas através de alguma codificação. No entanto, a forma como o sistema nervoso codifica informação através de *spikes* ainda é um campo de estudo aberto na neurociência [GERSTNER et al., 1997], [BURACAS and ALBRIGHT, 2014]. Até agora, neurocientistas podem distinguir três classes de codificação neural: *rate coding* ou codificação por taxa de disparo, *temporal coding* ou codificação temporal e *population coding* ou codificação por populações.

Codificação por taxa de disparo, também chamada de codificação em frequência de disparo, considera que a taxa de disparo dos neurônios carrega informações. É a mais tradicional forma

de codificação descrita na literatura [LEVINE, 2007], [ADRIAN and ZOTTERMAN, 1926]. Já a codificação temporal considera o tempo exato do disparo de um neurônio em relação a outros *spikes*, ao estímulo recebido, ou a algum sinal de referência [DAYAN and ABBOTT, 2001]. Codificação temporal inclui: qual neurônio dispara primeiro numa população, o tempo para o primeiro *spike* após um estímulo de referência e a ordem de disparo num conjunto de neurônios em uma janela de tempo [KOSTAL et al., 2007], [DAYAN and ABBOTT, 2001]. Já a codificação por populações considera que a informação é representada pelo número de neurônios ativos em um grupo durante certo período de tempo. Codificação por populações é um dos poucos problemas matematicamente bem formulados na neurociência [WU et al., 2002], [KUMAR et al., 2010].

De um modo geral, SNN requer hardware dedicado para o processamento de informações em tempo real. Isto ocorre porque os modelos de neurônios artificiais necessitam de um tempo de computação proibitivo para trabalhar na arquitetura dos computadores comerciais. Embora um único neurônio artificial não gaste muito tempo de processamento (ver [IZHIKEVICH, 2004]), devido à arquitetura sequencial dos computadores comerciais, torna-se inviável garantir que a rede trabalhe em tempo real quando o número de neurônios da rede cresce para poucas dezenas [IZHIKEVICH, 2004].

Para suplantar essa limitação computacional, o processamento de *spikes* em tempo real pode ser implementado em circuitos neuromórficos [BENJAMIN et al., 2014], [NEFTCI et al., 2013], [POON and ZHOU, 2011], que são circuitos integrados que tentam mimetizar características do sistema nervoso utilizando processamento analógico, por vezes associado a processamento digital. Ou também em sistemas digitais especializados configurados em FPGAs e SoCs [CHEUNG et al., 2012], [WANG et al., 2014], [WANG et al., 2013], [RICE et al., 2009], [CASSIDY et al., 2007], [SHAYANI et al., 2008]. Esses sistemas são capazes de processar grandes quantidades de dados paralelos. Todos estes hardwares especializados podem executar a operação em tempo real sobre trens de *spikes*, mas eles necessitam ser alimentados de forma adequada.

Quando os sinais a serem convertidos em *spikes* já se encontram em alguma codificação digital, um algoritmo de conversão é suficiente para a geração de trens de pulsos para rede. No entanto, quando os sinais vêm de fontes do mundo (sinais luminosos, sons, força mecânica, etc.) ou mesmo são sinais digitais gerados em tempo de execução (sensores remotos em uma rede WiFi, imagens de uma câmera CCD, etc.) um sistema dedicado pode ser necessário para conversão desses sinais em tempo hábil para que a rede funcione em tempo real. Uma solução plausível é o uso de hardware dedicado para conversão desses sinais e geração de trens de *spikes* necessários para alimentação da

SNN.

Outra possibilidade é a criação de circuitos neuromóricos que simulam neurônios sensores [PUMARICA et al., 2007], [PUMARICA and DEL-MORAL-HERNÁNDEZ, 2007], onde entradas de corrente ou tensão geram *spikes* proporcionais a estímulos de entrada. Sensores eletrônicos com saídas compatíveis com esse sistema podem ser ligados diretamente aos neurônios sensores. No capítulo 2 será mostrado que esta solução não é viável em muitos casos. Em resposta a uma série de desvantagens dessa abordagem, será proposto um sistema que executa a tarefa de conversão para *spikes* em hardware digital dedicado.

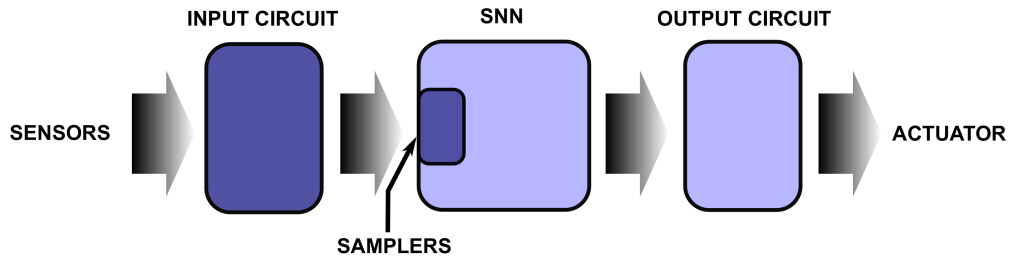
1.2 OBJETIVOS

A busca por alternativas em hardware que funcionem em tempo real para implementações eletrônicas de partes dos sistemas nervoso é um interesse atual [HYNNA and BOAHEN, 2006], [FARQUHAR and HASLER, 2004], [WEN and BOAHEN, 2003], [KAMEDA and YAGI, 2006], [KOICKAL et al., 2006]. A incorporação das novidades encontradas na neurociência aos sistemas digitais pode trazer avanços na construção de sistemas artificiais com um alto poder computacional e a criação de sistemas capazes de interagir com o sistema nervoso humano. Ou ainda, desenvolver modelos que auxiliem o estudo de fenômenos que são difíceis de se estudar em seres vivos, já que o ambiente artificial pode ser mais bem controlado.

O objetivo geral da pesquisa é construir uma SNN que trabalhe em tempo real (com a resolução temporal de sistemas biológicos), pegando dados do mundo, computando e tendo como saída o controle para um agente. No entanto, o desenvolvimento de uma máquina que compute SNN em tempo real implica em uma série de desafios que devem ser transpassados individualmente. De forma geral, um sistema que utiliza SNN de tempo real para controle de um agente possui esquema semelhante ao mostrado na Figura 1.1. Esse esquema é composto de um circuito de entrada (*input circuit*), a SNN e o circuito de saída (*output circuit*). O circuito de entrada transforma, em tempo real, grandezas físicas transduzidas por sensores em trens de pulsos. Esses trens de pulsos são as entradas da SNN. Alimentada por esses *spikes*, a SNN gera saídas que interferem no meio através dos atuadores. No entanto, as saídas da SNN também são trens de pulsos, logo, as saídas da SNN precisam ser convertidas de trens de pulsos para sinais compatíveis com os atuadores. Esse processo é feito pelo circuito de saída.

Devido ao tempo necessário para o estudo de cada um dos pontos da construção de uma SNN de tempo real, este projeto foca em investigar como alimentar a SNN de forma adequada, gerando os

Figura 1.1: *Esquema em blocos de uma SNN de tempo real utilizada para controle de um agente.*



spikes artificiais que alimentarão a SNN que trabalha em tempo real. Essa investigação engloba dois pontos mostrados na Figura 1.1: o *input circuit* e os *samplers*. Esses dois pontos são os objetivos específicos do projeto: como transformar sinais reais e simulados em trens de *spikes* em tempo real (*input circuit*) e como disponibilizar a SNN pode amostrar esses sinais tornando-os úteis a rede (*samplers*).

Primeiramente é mostrado como é possível alimentar SNNs com dados, físicos e/ou simulados, em tempo real através de um hardware dedicado de baixo custo. Esse sistema é capaz de realizar três tarefas independentes e correlatas, operando com 1024 (1k) neurônios por módulo. O sistema é capaz de, para cada um dos neurônios, gerar *spikes* com uma frequência programada pelo usuário. Ele também pode converter dados de entradas digitais em trens de *spikes* em tempo real. Além disso, pode organizar neurônios que disparam conjuntamente, a fim de gerar os três códigos neurais mais importantes atualmente, descritos na literatura da neurociência.

Posteriormente, é mostrado, através de simulação, como trens de pulsos codificados podem ser utilizados dentro da rede. Usando os conceitos de NAC, foram propostas topologias de amostradores de *spikes* tanto para sinais codificados por populações como em codificação temporal. Também são mostrados exemplos de circuitos de tomada de decisão que utilizam sinais amostrados. Esses circuitos são blocos básicos para computações mais complexas dentro da rede, servindo ainda de teste da usabilidade dos amostradores.

1.3 METODOLOGIA

Como descrito na seção anterior, os objetivos específicos desse trabalho de mestrado são como transformar informação do mundo em trens de *spikes* para alimentar SNNs de tempo real e como

amostrar essas informações codificadas em trens de pulsos de forma que a rede pulsada possa trabalhar com eles. Embora as duas partes possam ser desenvolvidas separadamente, como foi o caso, é necessário primeiramente o levantamento de como a informação pode ser transformada em pulsos.

Para isso, primeiramente foi levantado na literatura da neurociência como as informações do ambiente que cerca o ser humano são convertidas em pulsos neurais e como essas informações são codificadas. Como é interesse construir uma SNN de tempo real que trabalhe com a resolução temporal dos sistemas biológicos, é interessante levantar de que forma os sistemas biológicos transduzem a informação do meio em representações internas que serão processadas pela rede. Após esse levantamento, é possível trabalhar de forma separada os dois objetivos do trabalho.

Para o *input circuit* foi necessário fazer o levantamento dos requisitos do sistema, levando em consideração os estudos sobre os sistemas biológicos e confrontar com alternativas artificiais da eletrônica. Após o levantamento do requisitos, possou-se para etapa de implementação do projeto de um circuito em hardware dedicado para realizar a conversão e codificação de dados em trens de *spikes*. O projeto foi implementado por partes e os testes são descritos nesta dissertação. Por último, foi elaborado a documentação e descrição do circuito proposto, tanto para divulgação no meio cientifico, com artigo em congresso [OLIVEIRA NETO et al., 2015], quanto para manutenção do projeto e ampliação.

Já o estudos dos amostradores (*samplers*) utilizaram como metodologia a simulação das redes utilizando o software Matlab [MATHWORKS, 2015]. Para os testes das topologias criadas foram levadas em consideração as codificações neurais levantadas na primeira parte do trabalho. Essas topologias foram montadas obedecendo a abordagem NAC e foi utilizado como código base o *script* Matlab disponível em [RANHEL, 2012b]. Os parâmetros dos testes e os seus resultados, assim como o código resultante, são descritos neste trabalho.

1.4 ESTRUTURA DO DOCUMENTO

O conteúdo desta dissertação está dividido em seis capítulos. O trabalho está dividido da seguinte forma:

No capítulo 2 são introduzidos os conceitos dos transdutores biológicos e suas alternativas digitais. Começando por descrever o neurônio, célula ligada ao processamento do sistema nervoso e ao sistema sensorial. São mostrados em seguida os tipos de transdutores encontrados nos mamíferos e os tipos de codificação neural descritos na neurociência. Após essa pequena introdução aos transdutores biológicos são mostradas alternativas eletrônicas para o neurônio sensor, tanto analógicas como

digitais. Essa investigação continua com a descrição de circuitos que transformam sinais analógicos em sinais digitais na eletrônica.

O capítulo 3 faz uma introdução à área do conhecimento em que este trabalho está inserido, primeiramente explicando a definição de Inteligência Artificial que será usada nesse trabalho e as ramificações dessa área do conhecimento. Uma dessas ramificações são Redes Neurais Artificiais. O estudo sobre essas redes no capítulo é abordado tanto explicando os conceitos básicos como de forma cronológica, mostrando as contribuições dos pesquisadores da área. O capítulo é encerrado explicitando os conceitos básicos da subárea que o trabalho se insere: as Assembleias Neurais Pulsadas, que estão inseridas dentro dos estudos das SNNs.

No capítulo 4 é descrito o projeto do codificador neural e gerador de trem de pulsos para SNNs de tempo real. O capítulo se divide em três graus de abstração decrescentes. Primeiro são mostradas as características do projeto para um usuário que irá utilizar o sistema. No segundo grau de abstração é descrita a arquitetura implementada, dando enfoque maior em como o sistema realiza as operações que são descritas anteriormente. E por último, é mostrado o protótipo funcional implementado e os testes feitos com ele.

Já no capítulo 5, são descritas as topologias de amostradores para trens de pulsos codificados em codificação por populações e temporal. Em cada uma dessas topologias é mostrado um exemplo de circuito tomador de decisões que trabalha com os sinais amostrados.

Finalmente, no capítulo 6 é mostrada a discussão sobre os resultados mostrados nos capítulos 4 e 5, assim como as conclusões do trabalho. Ainda é conteúdo do capítulo a lista de publicações relacionadas ao projeto e indicação de trabalhos futuros.

CAPÍTULO 2

SENSORES BIOLÓGICOS E ELETRÔNICOS

2.1 INTRODUÇÃO

ESTE capítulo traz uma visão geral de como estímulos externos são transduzidos em sinais elétricos que o sistema nervoso consegue computar. Este estudo visa fazer um paralelo entre o sistema sensorial de seres vivos e alternativas eletrônicas. Este paralelo é interessante já que um dos objetivos deste trabalho é alimentar SNNs de tempo real com grandezas físicas do mundo externo a rede. Tais conhecimentos auxiliam a utilização de SNNs de tempo real para controle de um autômato cujos sensores geram respostas compatíveis com a rede pulsada.

Em virtude da ligação do sistema sensorial com o sistema nervoso se dar através do neurônio, serão primeiramente descritos no capítulo o neurônio biológico e suas características. Na segunda seção serão descritos quais são os tipos de sensores encontrados nos seres humanos e suas características físicas e elétricas. Na seção seguinte será mostrado como o sistema nervoso codifica esses sinais nos três grandes grupos de codificação descritos na literatura da neurociência. Na quarta seção serão mostradas alternativas eletrônicas aos sensores encontrados na biologia. Será feito um paralelo de sistemas possíveis que possuem uma resposta semelhante, tanto na eletrônica analógica quanto na eletrônica digital. Seguindo com essa explicação, é mostrado na última seção do capítulo como sinais analógicos são convertidos em sinais digitais na eletrônica.

2.2 O NEURÔNIO

No sistema nervoso humano há cerca de 86 bilhões de neurônios [HERCULANO-HOUZEL, 2009], cada neurônio faz cerca de 10 mil conexões com os neurônios vizinhos e com ele mesmo. Esta extensa e complicada rede de neurônios interconectados é a responsável pelo controle central do organismo, assim como nossa memória, pensamento, tomadas decisões, etc. O neurônio, célula base dessa rede, é composto de **dendritos**, **corpo celular** e **axônio** como mostrado na Figura 2.1 .[1]

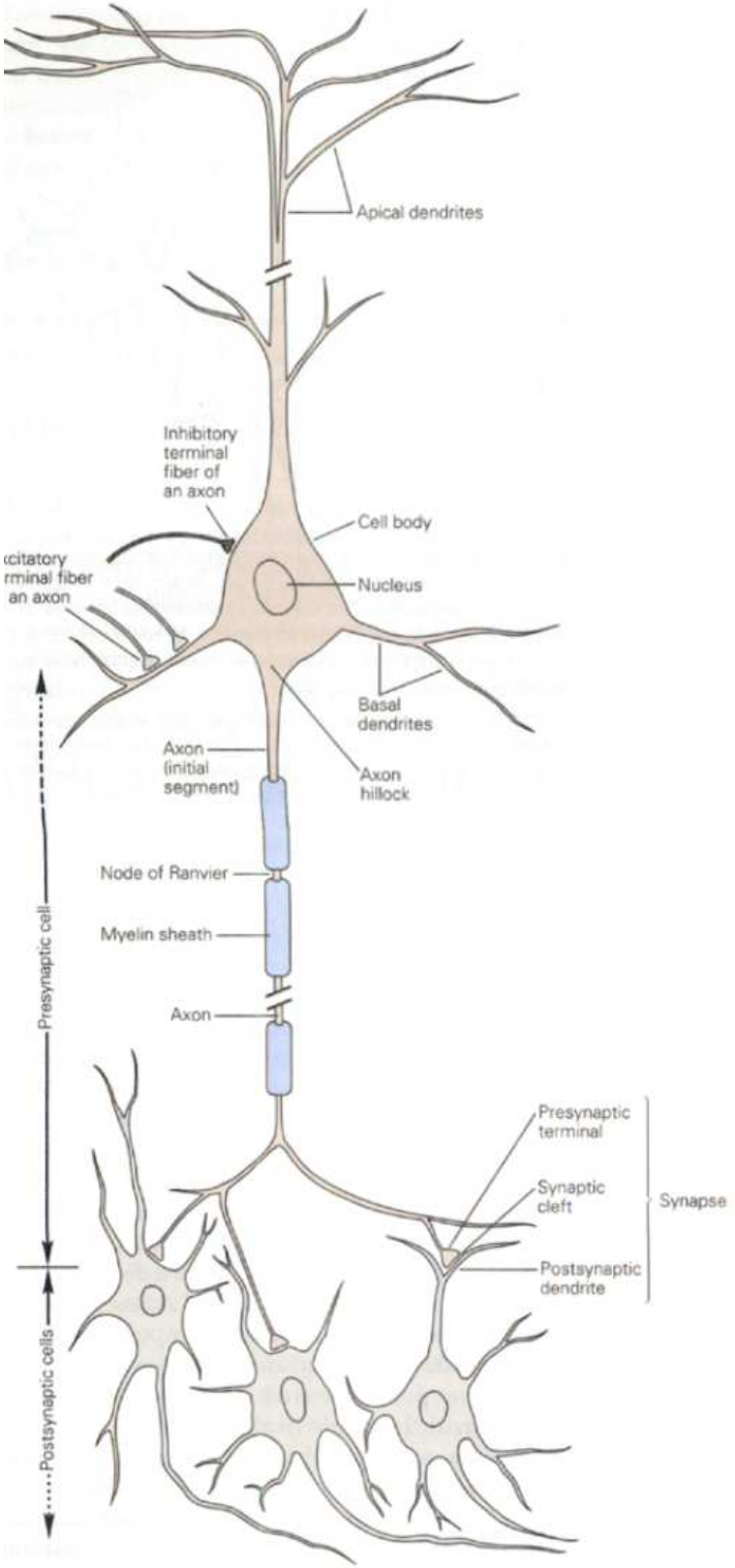
Em seu funcionamento, o neurônio integra os sinais elétricos vindos dos outros neurônios ligados a ele. Dependendo da resultante desses sinais no tempo o neurônio "dispara" ou não. Este "disparo" é chamado **potencial de ação**, também conhecido pelo termo inglês ***spike***. O potencial de ação é um jeito econômico de codificar informação, pois o neurônio fica ativo apenas durante o curto tempo em que ocorre o potencial de ação, logo o custo energético é pequeno [PAPROCKI et al., 2013].

Na maior parte das vezes, os sinais elétricos passam do axônio de um neurônio para o dendrito de outro neurônio ligado a ele, como mostrado na Figura 2.1. A ligação entre os neurônios ganha o nome de **sinapse**, nela ocorre a propagação do sinal elétrico de um neurônio para outro. O neurônio cujo axônio está ligado a uma sinapse é chamado **neurônio pré-sináptico** e o neurônio que receberá o estímulo pelo dendrito participante da sinapse é chamando **neurônio pós-sináptico**. Potenciais de ação nas conexões pré-sinápticas causam um aumento ou diminuição na tensão interna do neurônio pós-sináptico. Quando em repouso, o neurônio possui um potencial negativo em relação o meio extracelular. Quando o potencial de ação é excitatório existe aumento no potencial interno da membrana, este fenômeno é chamado EPSP (*Excitatory Post-Synaptic Potential*). Por outro lado, potenciais de ação inibitórios causam uma diminuição na tensão interna da membrana celular do neurônio pós-sináptico, tornando-a mais negativa, ou hiperpolarizada, chamado IPSP (*Inhibitory Post-Synaptic Potential*) [KANDEL et al., 2000].

Dentro do neurônio, o sinal elétrico é transmitido por uma corrente de íons dos dendritos para os terminais do axônio, mas na sinapse esse estimulo pode ser passado através de **neurotransmissores**, **sinapses químicas**, ou através de íons, quando fluem pelas chamadas **sinapses elétricas**, que podem ser entendidas como conexões diretas entre neurônios. Nas sinapses elétricas a junção entre o axônio do neurônio pré-sináptico e o dendrito do neurônio pós-sináptico possui uma distância de apenas 3 nm aproximadamente, e são formadas por canais, chamados *conexons*, que permitem que íon passem diretamente do citoplasma de uma célula para outra. Essa junção recebe também o nome de junção comunicante [BEAR et al., 2007].

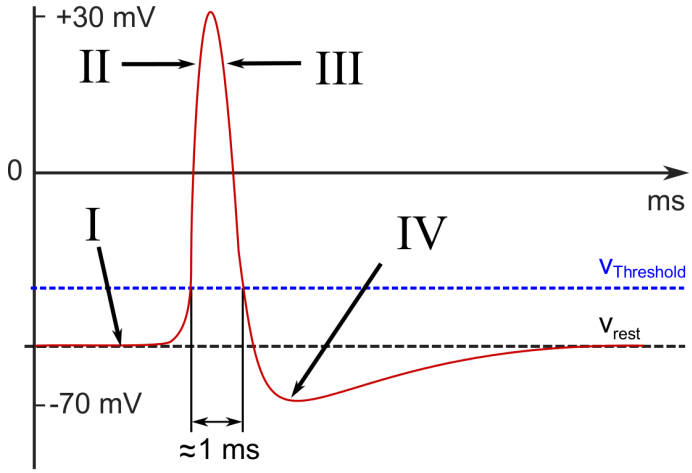
[1] As figuras retiradas de [KANDEL et al., 2000] são reproduzidas com permissão da *McGraw-Hill Education.*

Figura 2.1: *Esquema do neurônio biológico com suas principais partes destacadas e sinapses com neurônios vizinhos (Figura retirada de [KANDEL et al., 2000]).*



No entanto, a maioria das sinapses encontradas nos mamíferos é química, onde o sinal é transmitido através dos neurotransmissores. Os neurotransmissores são substâncias produzidas pelo neurônio e utilizadas como sinalizadores na comunicação celular com outros neurônios, órgãos e músculos. Os neurotransmissores podem ser inibitórios ou excitatórios, gerando os IPSPs e EPSPs, respectivamente. A força do sinal transmitido é uma função dos neurotransmissores e da quantidade de canais receptores presentes na **fenda sináptica** (local físico no qual as conexões acontecem). O conjunto de canais na fenda sináptica associado à quantidade de neurotransmissores emitidos pelo neurônio pré-sináptico determina a eficiência da sinapse, que pode atenuar ou acentuar um sinal entre um neurônio e outro. A esse conjunto é possível atribuir um valor que quantifica a qualidade sináptica, chamado de **peso sináptico**.

Figura 2.2: *Potencial de ação do neurônio biológico. No gráfico é mostrado o tempo (ms) no eixo das abcissas e a tensão interna da membrana do neurônio (mV) no eixo das ordenadas.*



A Figura 2.2 mostra o potencial de ação típico de um neurônio biológico, ou seja, o disparo do neurônio. A tensão interna na membrana quando o neurônio está em repouso é aproximadamente de -65 mV, indicado po V_{rest} na Figura 2.2. No entanto, quando os EPSPs fazem o potencial da membrana passar de certo limiar, uma sucessão de eventos acontecem na célula: proteínas, que transpassam a membrana do neurônio, se torcem causando a abertura de canais na membrana. Então, cargas positivas entram na célula, tornando a tensão interna cada vez maior. Quando a tensão da membrana chega a $\sim +30$ mV (Figura 2.2 - [II]), é dito que o neurônio **disparou**. O disparo acarreta um processo de recuperação: íons positivos escapam da célula, fazendo com que a tensão na membrana volte para valores negativos (Figura 2.2 - [III]). O processo de repolarização deixa a membrana com uma tensão menor que -65 mV, e a volta à tensão de repouso demora alguns milissegundos (Figura 2.2 - [IV]). O período indicado em (Figura 2.2 - [III e IV]) é chamado refratário,

em que o neurônio se recupera do disparo. A duração e amplitude do potencial de ação (Figura 2.2 - [II e III]) são mais ou menos fixos em 1 ms e $+30$ mV. Na Figura 2.3 [2] são mostrados os tipos de resposta encontrada em neurônios biológicos para diferentes estímulos de entrada e, embora tenham comportamentos diferentes, o potencial de ação em si não difere muito em amplitude ou duração. Então, assume-se que o potencial de ação carrega apenas a informação do disparo, que pode ser representado por um bit no tempo: se o neurônio disparou (1) ou não (0).

2.3 TRANSDUTORES SENSORIAIS BIOLÓGICOS

O Sistema Sensorial dos seres vivos é responsável por reconhecer estímulos internos e do ambiente. Engloba vários tipos de **receptores sensoriais** responsáveis por fazer a transdução de informações de natureza **química**, **óptica**, **mecânica** e **térmica** para sinais de ativação compatíveis com o sistema sensorial do agente. Como neste trabalho serão comumente tratadas as características dos seres vivos (agentes biológicos) e dos sistemas eletrônicos bioinspirados (agentes artificiais), quando pertinente, será utilizado o termo agente para ambos.

Embora o Sistema Sensorial trate tanto dos estímulos internos ao agente como externos, será dado enfoque aos receptores sensoriais que transduzem as informações vindas do ambiente que o cerca. Essa escolha deve-se ao fato deste trabalho tratar desses tipos de receptores específicos e suas versões digitais. Nas subseções seguintes serão explicitados esses receptores sensoriais, dividindo-os a partir da natureza do sinal que os excita.

2.3.1 QUIMIORRECEPTORES

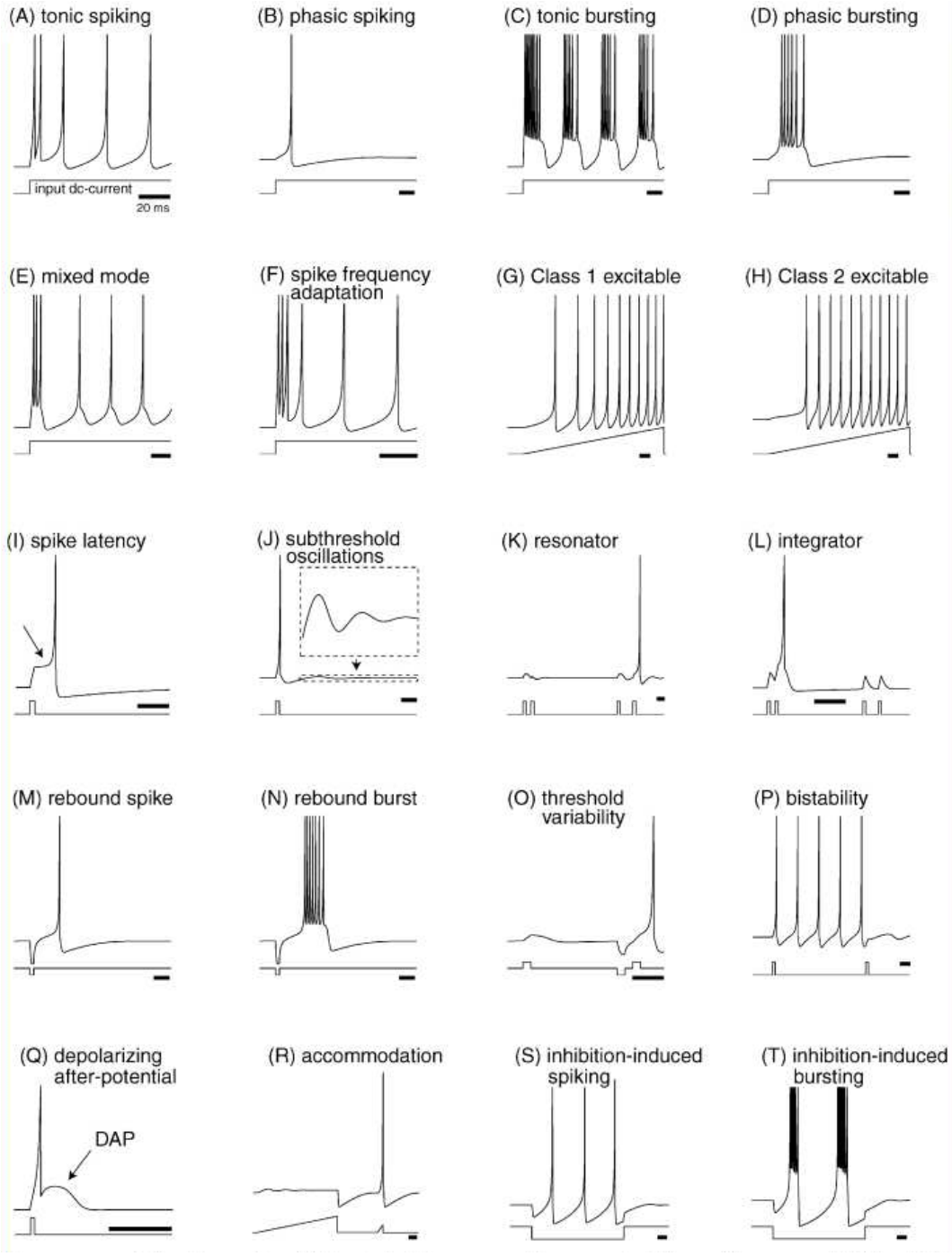
Olhando a escala evolutiva, desde as bactérias primitivas são encontrados sensores que verificam, por exemplo, o PH do meio, a presença de substancias nocivas, gradientes que indiquem alimento, etc. [BEAR et al., 2007]. Nos seres humanos, os receptores químicos responsáveis por transduzir informações vindas do meio externo estão relacionados a sentir "gostos", o paladar, e a sentir "cheiros", o olfato.

Receptores Gustativos

O ser humano é capaz de agrupar as substancias químicas em uma pequena classe de sabores. Embora ainda não exista um consenso no meio acadêmico, acredita-se que distinguamos não mais

[2] Versão eletrônica das figuras de [IZHIKEVICH, 2004] e permissões de reprodução estão disponíveis gratuitamente em www.izhikevich.com.

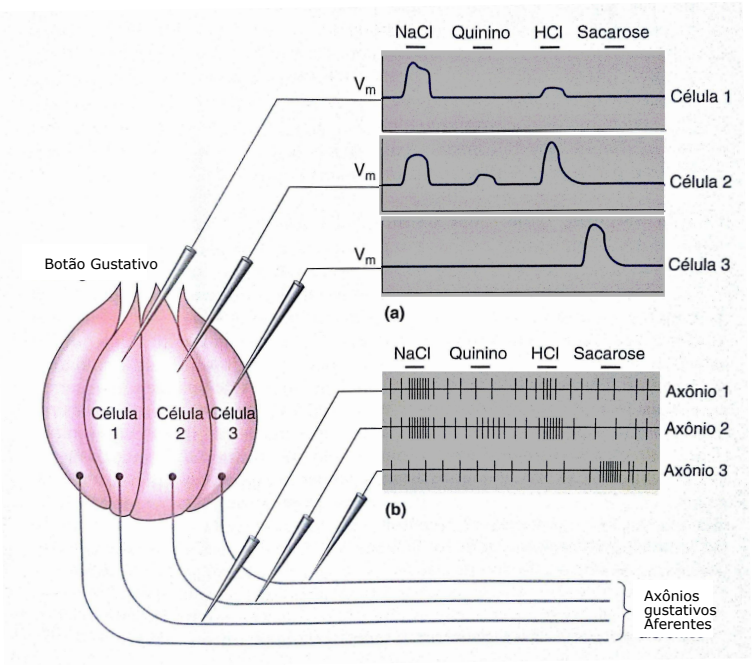
Figura 2.3: *Tipos de padrão de resposta para diferentes tipos de neurônio (Figura retirada de [IZHIKEVICH, 2004]).*



de 4 ou 5 sabores. Esses sabores seriam os quatro mais conhecidos – doce, azedo, amargo e salgado – e um quinto sabor, relacionado à percepção do glutamato. No entanto, os padrões organolépticos "fundamentais" para o paladar, que contornam qualquer ambiguidade no significado da sensação são, respectivamente, o ácido cítrico (azedo/ácido), o quinino (amargo) a glicose (doce) e o $NaCl$ (salgado) [BEAR et al., 2007]. A infinidade de sabores, gostos, que o ser humano consegue distinguir é devido a esses sabores combinados com os cheiros e textura da substância.

Quando os receptores gustativos são ativados por uma substância química apropriada, seu potencial de membrana muda, tornando mais negativa, o que é chamado de **potencial receptor**. Se o potencial receptor é despolarizante o suficiente, os receptores gustativos podem disparar potenciais de ação. A transmissão sináptica básica que ocorre entre os receptores gustativos e os neurônios aferentes é através da abertura de canais de íons de cálcio (Ca^{2+}) na membrana dos receptores gustativos. A entrada desses íons no citoplasma libera neurotransmissores nas ligações com os neurônios aferentes. A quantidade de canais que são abertos é uma função da despolarização da membrana, assim como a quantidade de neurotransmissores liberados é proporcional a quantidade de Ca^{2+} no citoplasma.

Figura 2.4: *Responsividade das células gustativas e axônios gustativos. (a) Três células diferentes foram expostas aos estímulos diferentes com $NaCl$, quinino, HCl e sacarose e seus potenciais de membrana foram registrados com eletrodos. (b) Foi registrada a descarga do potencial de ação nos axônios aferentes. Cada deflexão vertical no registro é um spike (Figura adaptada de [BEAR et al., 2007]).*



Em sua maioria, os receptores gustativos respondem a dois ou mais dos sabores básicos. Essa seletividade não é muito grande, fazendo com que gradientes grandes de qualquer uma das substâncias gerem um potencial receptor. No entanto, as células gustativas e seus axônios correspondentes são bem específicas em seu padrão de resposta, cada um dos quatro tipos de axônio gustativo possui um padrão de *spikes* característico para cada um dos sabores básicos, como pode ser visto na Figura 2.4. Mesmo com essa especificidade, os neurônios "disparam" pelo menos para dois tipos de sabores. A distinção entre sabores só é feita no Sistema Nervoso Central (SNC), por uma avaliação dos estímulos que chegam de todos os receptores gustativos.

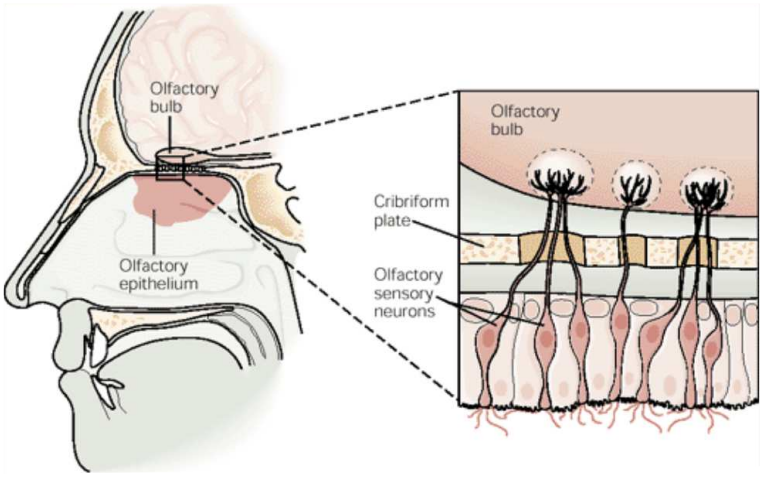
Receptores Olfativos

Como dito na seção anterior, o olfato auxilia a gustação na percepção dos sabores. Mas, além disso, ele é responsável por detectar no meio gradientes de substâncias nocivas. Devido a isso, a quantidade de odores desagradáveis que o olfato humano consegue distinguir é maior que a quantidade de cheiros agradáveis, que se restringem a cerca de 20% apenas [BEAR et al., 2007].

Outra diferença encontrada é que, ao contrário dos receptores gustativos, os **receptores olfativos** são neurônios, com axônios que levam a informação em direção ao SNC. Esses sensores são encontrados em uma fina camada no alto da cavidade nasal, chamado de **epitélio olfativo**. No ser humano essa área tem cerca de 10 cm^2 e consiste de uma camada de **células olfativas** (os sensores), **células de suporte** (auxiliam na produção do muco) e **células basais** (fonte de nervos receptores), como mostrado na Figura 2.5. Substâncias odoríferas dissolvem-se na camada de muco entram em contato com os cílios das células olfativas. Os axônios dessas células olfativas penetram na cavidade óssea cribiforme em direção ao SNC.

Existem no ser humano certa de 1000 diferentes tipos de receptores olfativos [KANDEL et al., 2000], que diferem pelo tipo de substância química a que são sensíveis. No entanto, todos os receptores olfativos seguem o mesmo processo de transdução: quando existe a ligação do receptor com uma dada substância química há estimulação da proteína G_{olf}, proteína G é uma classe de proteínas envolvida na transdução de sinais celulares. A presença dessa proteína no citoplasma desencadeia a formação de outras substâncias, enzimas, que modificam a concentração de guanosina monofosfato cíclica (GMPc), o aumento da concentração de GMPc no meio intracelular ativa os canais de íons que acarretam na mudança do potencial da membrana do neurônio. Quando o estímulo é grande o suficiente ocorre o potencial de ação.

Figura 2.5: *Localização dos receptores olfativos na cavidade nasal e a estrutura do epitélio olfativo. O epitélio olfativo (olfactory epithelium) consiste de três tipos de células: células olfativas (olfactory sesory neurons), células de suporte (entre as células olfativas) e basais (nervos receptores no olfactory bulb) (Figura retirada de [KANDEL et al., 2000]).*



2.3.2 FOTORRECEPTORES

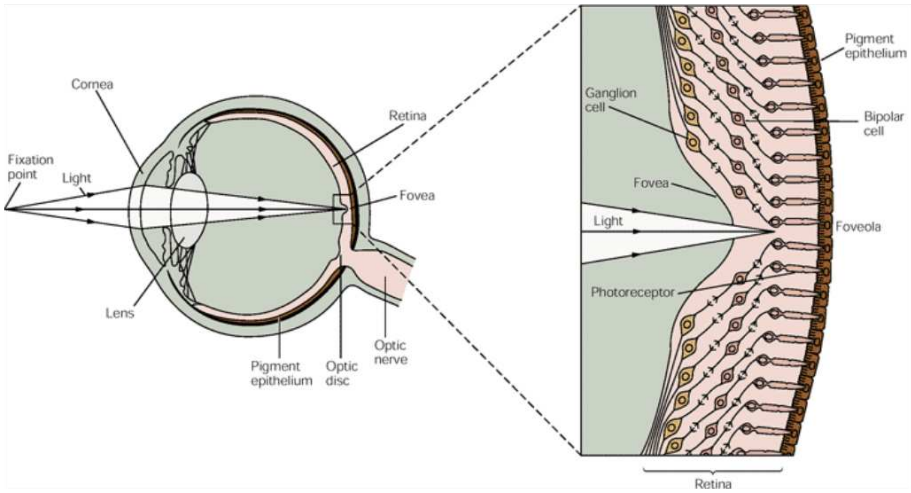
A visão humana é um sentido bem complexo, sendo composto desde a parte óptica do olho onde temos um conjunto de lentes, passando por um sistema de transdução da luz em informação, até a representação da imagem no cérebro. Cerca de metade do córtex cerebral está envolvido com a análise do mundo visual [BEAR et al., 2007]. Devido ao foco deste trabalho, será exposto em maiores detalhes apenas a parte do sistema visual responsável pela transdução do sinal luminoso em *spikes*, caracterizando o fotosensor e seu funcionamento.

Na retina se encontram as células responsáveis pela transdução da informação luminosa em sinais elétricos, sendo considerada parte do encéfalo. Ela está disposta em camadas de células responsáveis tanto pela transdução do sinal luminoso em atividade neural como um pré-processamento da imagem. As únicas células fotossensíveis na retina se encontram na camada mais interna, a luz transpassa as outras camadas e é absorvida pelos **fotorreceptores**. O ser humano possui dois tipos de fotorreceptores, os **cones** e os **bastonetes**, localizados na parte anterior da retina humana. Existindo cerca de 125 milhões de fotorreceptores no olho [BEAR et al., 2007].

Fototransdução

Nos fotorreceptores são encontrados fotopigmentos sensíveis a luz. Nos bastonetes, o fotopigmento que é estimulado por fótons é a proteína **rodopsina**. A rodopsina, assim como as demais proteínas sensíveis a fótons, é uma **opsina**. Assim como nos receptores olfativos, o pigmento fotor-

Figura 2.6: *Estrutura óptica do olho. Raios de luz incidem no olho e são refratados pela córnea (cornea) e pelo cristalino (lens). A luz refratada incide na parte traseira do olho chamada de retina (Figura retirada de [KANDEL et al., 2000]).*



receptor está ligado a uma proteína G. Quando fótons incidem na rodopsina existe uma mudança de conformação da proteína. Essa mudança de conformação ativa a proteína G, que ativa uma enzima que muda a concentração de um segundo mensageiro no meio intracelular, a GMPc, assim como ocorre no receptor olfativo. No entanto, enquanto nas células olfativas existe o aumento da concentração de GMPc, que abre mais canais iônicos, nos bastonetes temos a hidrólise da GMPc. A diminuição de sua concentração no citoplasma fecha os canais iônicos que ocasionam uma hiperpolarização da membrana celular, podendo levar ao *spike*.

Como mostrado na seção 2.2, o potencial na membrana de um neurônio quando em repouso é aproximadamente -65 mV. No entanto, nos bastonetes, quando não existe estímulo luminoso, os canais iônicos encontram-se abertos, gerando um potencial na membrana de aproximadamente -30 mV, chamada de **corrente de escuro**. Isso ocorre porque nos fotorreceptores a proteína GMPc é continuamente produzida pela célula e, como mostrado, ela regula a abertura dos canais iônicos na célula.

Nos cones o processo de fototransdução é análogo ao encontrado nos bastonetes. No entanto temos duas diferenças: uma se dá pela quantidade de fotopigmento, que é bem menor nos cones, e a outra é que o fotopigmento muda nos três tipos de cones. A opsina encontrada em um cone pode ser mais sensível a comprimentos de onda ao redor de 430 nm, chamados cones "para o azul", ao redor de 530 nm, chamados cones "para o verde", ou ao redor de 560 nm, os cones "para o vermelho". Os cones são, então, responsáveis pela percepção de cor, o que explica vermos cinza em ambientes de baixa lumiosidade.

2.3.3 MECANORRECEPTORES

Mecanorreceptores são transdutores de deformações mecânicas em sinais elétricos. O ser humano possui mecanorreceptores ligados aos sentidos do tato, audição e o sistema vestibular, responsável pelo equilíbrio. O sistema auditivo e o sistema vestibular estão ligados anatomicamente, embora tragam informações não correlacionadas para o agente. Nesta seção serão abordados os **receptores auditivos** e os **receptores táteis**.

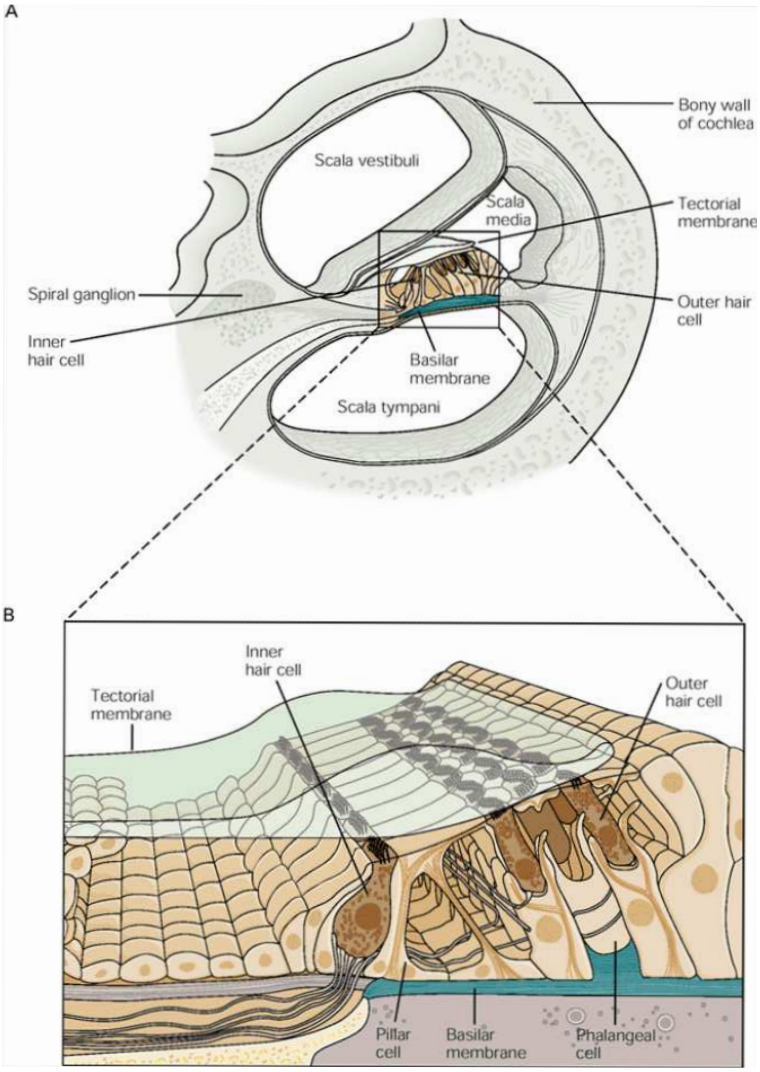
Receptores Auditivos

Os receptores auditivos são as células ciliadas, elas ficam localizadas no ouvido interno, mais especificamente na **cóclea** e estão dispostas em quatro fileiras acomodadas entre a **membrana basal** e a **membrana tectorial**, como mostrado na Figura 2.7. As células ciliadas recebem esse nome por possuírem cerca de uma centena de cílios cada. Esses cílios estão agrupados em três fileiras com tamanhos diferentes, formando uma escada. Os cílios de um degrau possuem ligações laterais entre si e possuem uma ligação do topo de um cílio com topo do cílio adjacente da camada vizinha. Como os cílios estão ligados entre si, quando uma onda sonora se propaga na cóclea os cílios se movem em conjunto. A percepção do ouvido humano pode pegar uma amplitude de movimento nos cílios de apenas $0,3$ nm, o diâmetro de um átomo grande, e tem sua resposta saturada para um movimento de 20 nm.

Essa percepção de movimento é feita através da abertura de canais de potássio no topo dos cílios. A ligação no topo de um cílio com a camada adjacente controla mecanicamente a abertura dos canais de potássio em ambos os cílios através de um filamento elástico. Quando em repouso, a tensão no filamento é suficiente para deixar o canal de potássio parcialmente aberto, fazendo íons de potássio do meio extracelular entrar na célula ciliada. Quando ocorre o movimento em uma direção a tensão no filamento aumenta e o canal de potássio abre mais, passando mais íons de potássio para o meio intracelular. Quando o movimento é no sentido contrário a tensão no filamento diminui fechando o canal de potássio.

A entrada de potássio na célula despolariza-a e faz com que canais de cálcio dependentes da tensão sejam abertos. O cálcio no meio intracelular dispara a liberação de neurotransmissores, gerando um potencial de ação que é propagado para os neurônios espinhais que propagam o sinal para o nervo auditivo.

Figura 2.7: *Visão transversal da cóclea mostrando as membranas que separam a regiam interna em três cavidades: escala vestibular (scala vestibuli), escala média (scala media) e escala timpânica (scala tympani). A área destacada na parte de baixo da figura mostra a disposição dos receptores auditivos - células ciliadas internas (inner hair cell) e células ciliadas externas (outer hair cell) (Figura retirada de [KANDEL et al., 2000]).*

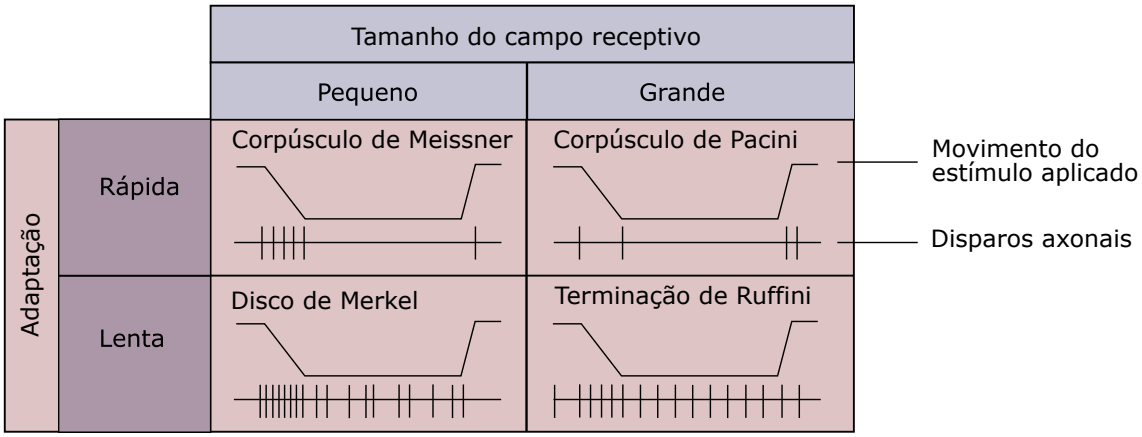


Receptores Táteis

A pele humana tem como principais funções a proteção mecânica para os órgãos internos e evitar que seja perdido água em excesso para o ambiente seco em que o ser humano habita. No entanto ela é também o maior órgão sensorial do corpo humano, nela estão distribuídos mecanorreceptores responsáveis pelo sentido de tato.

Existem diversos tipos de receptores encontrados na pele. Enquanto alguns desses receptores são terminações nervosas livres, que reagem a estímulos mecânicos, químicos e térmicos, outros são organizados em corpúsculos mecanorreceptores. Os mecanorreceptores são os **Corpúsculos de Meissner**, que percebem pressões de frequências diferentes, os **Discos de Merkel**, que percebem movimentações e pressões leves, os **Corpúsculos de Vater-Pacini**, que percebem pressões e são encontrados em grande número nas pontas dos dedo e os **Corpúsculos de Ruffini**, que percebem distensões na pele e calor.

Figura 2.8: *Quadro mostrando as respostas dos receptores táteis em relação ao campo de atuação e a intensidade do estímulo (Figura adaptada de [BEAR et al., 2007]).*



Como ocorre nos receptores auditivos, os potenciais de ação nos receptores táteis são gerados a partir da abertura de canais iônicos devido à deformação no receptor. A sensibilidade entre os diferentes corpúsculos é relacionada, então, pela diferença física de seus terminais sensíveis. Por exemplo, o Corpúsculo de Pacini possui uma cápsula sensitiva que se assemelha a uma cebola alongada, possuindo entre 20 a 70 camadas concêntricas de tecido conectivo com o terminal nervoso no centro. Quando a cápsula é comprimida, a energia mecânica é transferida ao terminal nervoso, sua membrana é deformada e canais mecanossensíveis se abrem. Dependendo da intensidade do estímulo é gerado um potencial de ação. As diferentes respostas encontradas nos diferentes receptores podem ser vistas na Figura 2.8.

As camadas da cápsula do Corpúsculo de Pacini são lisas e possuem um fluido viscoso entre elas. Se a pressão de estímulo for mantida, as camadas deslizam umas sobre as outras e transferem a energia do estímulo entre as camadas, fazendo que o terminal nervoso no centro da cápsula não sofra tração mecânica. Como não ocorre a abertura de canais iônicos, não é gerado um potencial de ação. Quando o a pressão é liberada, a cápsula volta ao normal tornando-se sensível a uma nova pressão.

2.3.4 TERMORRECEPTORES

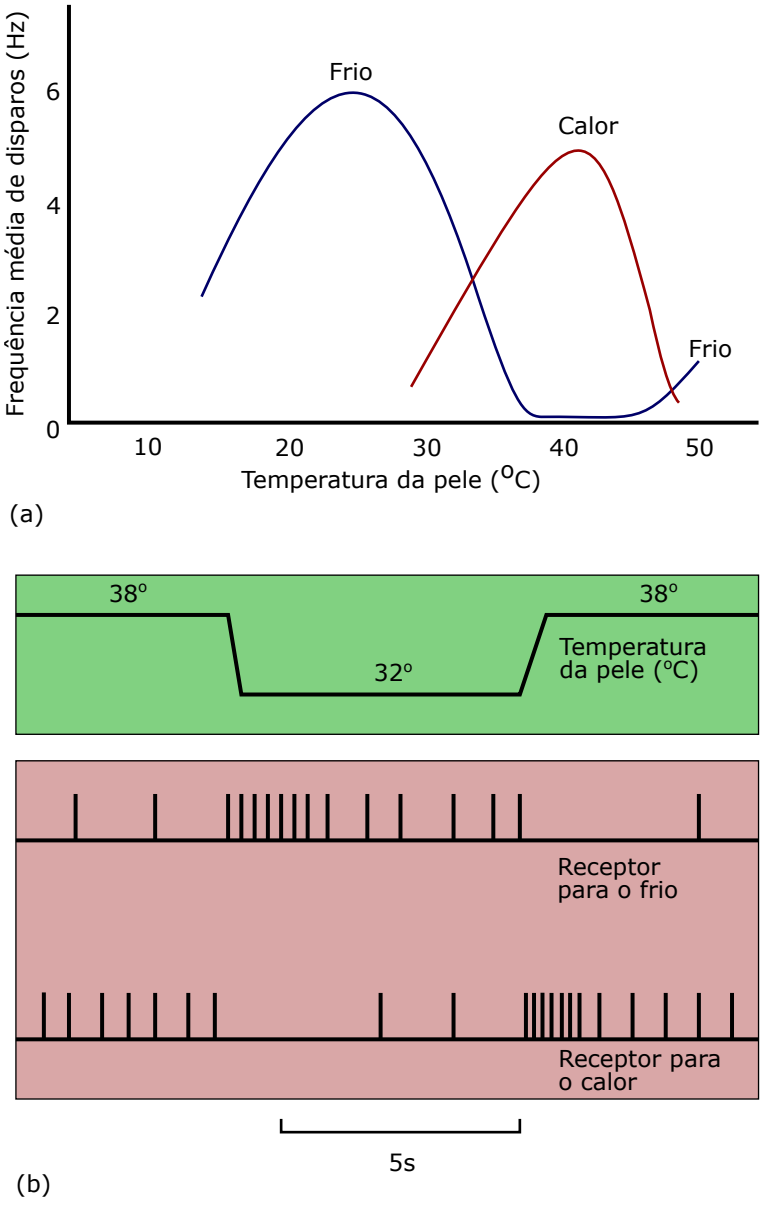
Existem no corpo humano ao menos dois tipos de receptores de temperatura, que, assim como os receptores táteis, se localizam sob a pele. Os dois tipos de receptores de temperatura se diferenciam pela faixa de temperatura a que respondem: se são sensíveis ao frio ou ao calor, como mostrado na Figura 2.9. Enquanto os receptores de calor respondem melhor na faixa de 30 a 45 graus Celsius, os receptores para o frio trabalham na faixa ente 10 e 35 graus Celsius.

Os receptores de temperatura são terminações livres de neurônios distribuídos entre a pele. Nesses terminais livres existem canais iônicos sensíveis à temperatura, proteínas incrustadas na membrana plasmática que mudam sua configuração espacial quando sob efeito da temperatura, "abrindo" um canal por onde íons podem entrar na célula. Dependendo da temperatura haverá mais ou menos canais iônicos abertos, ocorrendo uma despolarização da membrana maior ou menor. Essa despolarização pode gerar um *spike*.

2.4 TIPOS DE CODIFICAÇÃO NEURAL

Nas seções anteriores foi discutido como neurônios realizam a transdução de diferentes formas de energia presentes no meio em sinais neurais, os *spikes*. No entanto os *spikes* apenas carregam um bit de informação: se o neurônio disparou ou não. O *spike* é um pulso que acontece por aproximadamente 1 ms e que tem aproximadamente a mesmo comportamento em diferentes tipos de neurônio. Devido a isso, informações transduzidas para *spikes* devem estar representadas em algum tipo de padrão de disparo que representa algum tipo de 'código' para o sistema nervoso. A detecção e correta operação sobre tais padrões de disparo resulta no processamento da informação, ou na computação que os sistemas nervosos executam sobre as percepções externas. Até agora, neurocientistas classificam três tipos de codificação neural: a **codificação por taxa de disparos**, a **codificação temporal** e a **codificação por populações**.

Figura 2.9: *(a) Taxa de disparos dos potenciais de ação em função da temperatura da pele para os receptores para frio e calor. (b) Respostas dos receptores para frio e calor a uma rápida redução da temperatura da pele. (Figura adaptada de [BEAR et al., 2007]).*

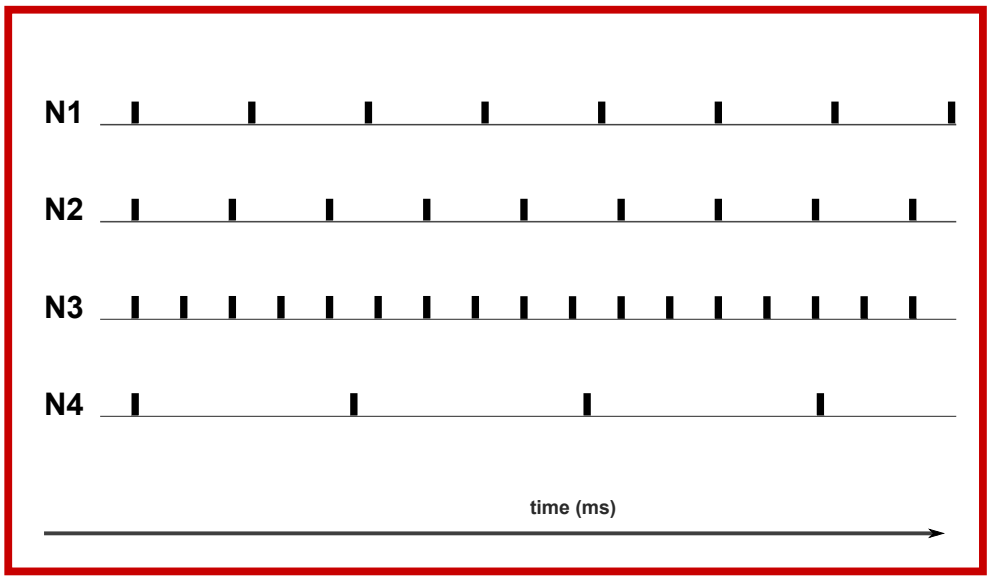


2.4.1 Codificação por Taxa de Disparos

Codificação por taxa de disparos é também conhecida como codificação em frequência de disparos. Nesse tipo de codificação é considerado que a taxa de disparo do neurônio por intervalo de tempo (geralmente um segundo), é que carrega informação. Esta é a forma de codificação mais tradicional descrita na literatura da neurociência [LEVINE, 2007], [ADRIAN and ZOTTERMAN, 1926].

Pode ser visto na Figura 2.10 quatro neurônios (**N1** a **N4**) disparando no tempo com taxas de disparos diferentes entre si, onde os disparos são indicados pelos pulsos verticais. Levando em consideração a codificação por taxa de disparos, **N3** representaria a maior grandeza entre os quatro neurônios, por outro lado **N4** seria a menor magnitude.

Figura 2.10: *Exemplo de codificação por taxa de disparos*



Embora a codificação por taxa de disparos seja um conceito bastante intuitivo, e que existam estudos dede a primeira década do século XX [LEVINE, 2007] comprovando que neurônios disparam com maior frequência para sinais mais intensos, em operação normal, neurônios disparam poucas vezes por segundo [KANDEL et al., 2000]. Para alguns estímulos, como a visão humana [THORPE et al., 1996], a resposta dada pelo SNC é mais rápida que o necessário para que seja calculada uma taxa de disparos dos neurônios envolvidos. Para certos tipos de estímulos são encontradas formas diferentes de codificação neural que envolvem a associação entre neurônios. Na literatura da neurociência são descritos dois tipos de codificação com associação de neurônios, a codificação temporal [DAYAN and ABBOTT, 2001], [KOSTAL et al., 2007] e a codificação por populações [WU et al., 2002], [KUMAR et al., 2010].

2.4.2 CODIFICAÇÃO TEMPORAL

Codificação temporal considera o instante específico em que um *spike* acontece [DAYAN and ABBOTT, 2001]. A codificação temporal pode associar a ordem de chegada de *spikes* de um grupo de neurônios, flutuações na frequência de disparo em relação a um sinal de sincronismo interno ou mesmo qual foi o primeiro neurônio a disparar em relação a um sinal de sincronismo [KOSTAL et al., 2007].

A Figura 2.11 ilustra alguns tipos de codificações temporais com sincronismo. Algumas possibilidades interessantes de codificação temporal, descritas a seguir, partindo da classificação de *Gerstner* e *Kistler* [GERSTNER and KISTLER, 2002].

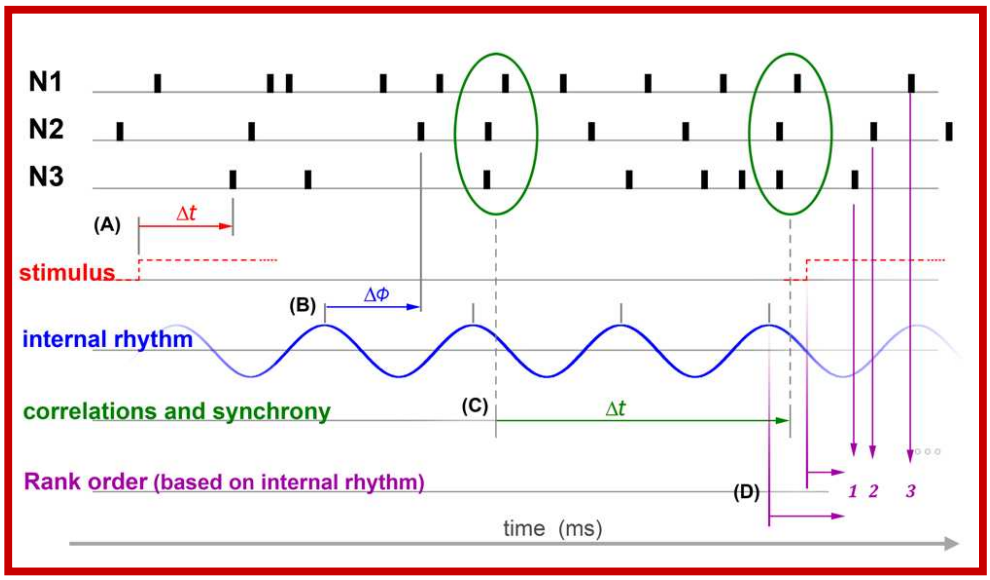
(A) Pela latência ou tempo de atraso de um pulso (*Time-to-First-Spike*). Este tipo de codificação se baseia no tempo de atraso, ou latência, dos pulsos em relação a um sinal de referência.

(B) Pela fase do pulso em relação a oscilações internas (*Phase coding*), que pode ser entendido como uma variação do primeiro.

(C) Pela correlação e sincronismo (*Correlations and Synchrony*) entre neurônios ou assembleias, sem uma referência fixa.

(D) Ordem de chegada (*Rank order*), em que a ordem de chegada dos pulsos representa um sistema de codificação relevante. Thorpe e colaboradores demonstraram que uma particular ordem de chegada dos pulsos é apenas uma das $N!$ ordens possíveis, N representando o número de neurônios envolvidos na codificação [THORPE et al., 2001].

Figura 2.11: *Tipos de Codificações Temporais (Figura retirada de [RANHEL, 2012b])*

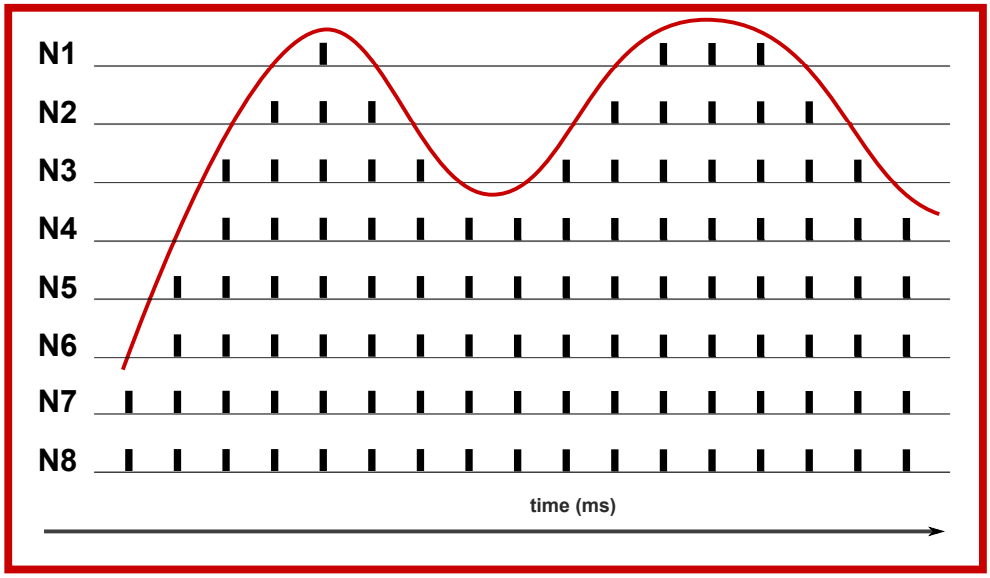


2.4.3 Codificação por Populações

Na codificação por populações é considerado que a informação é representada pelo número de neurônios disparando em um grupo durante certo período de tempo. Codificação por populações é um dos problemas matematicamente bem formulados dentro da neurociência [WU et al., 2002], [KUMAR et al., 2010].

Na Figura 2.12 é mostrada uma população de oito neurônios modulando uma grandeza ao longo do tempo. Ou seja, a intensidade do sinal representado é diretamente proporcional ao número de neurônios da população que estão disparando em um determinado momento.

Figura 2.12: *Exemplo de neurônios disparando em conjunto formando uma população.*

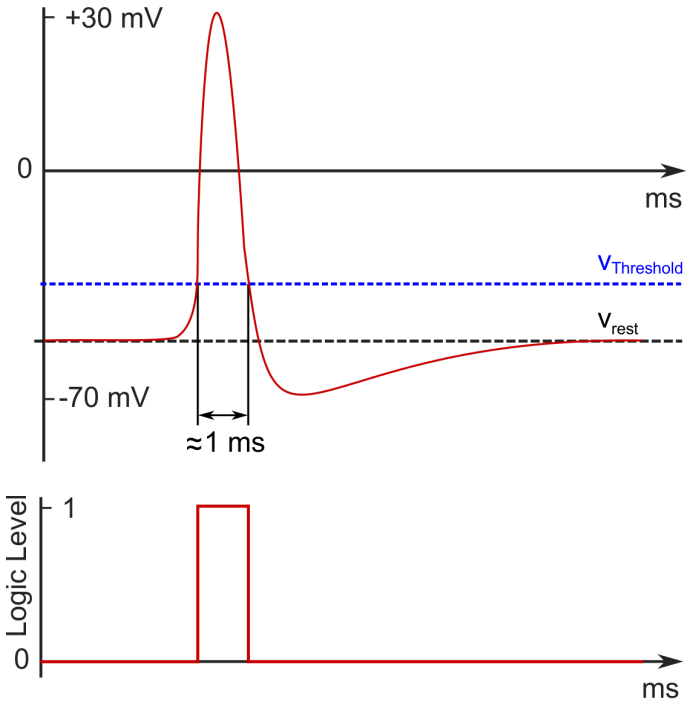


2.5 Engenharia de Sistemas Sensoriais

Como dito na seção 2.2 do presente capítulo, os potenciais de ação são impulsos elétricos gerados pelos neurônios biológicos que duram, em média, um milissegundo (1ms) [KANDEL et al., 2000]. A informação que um potencial de ação ocorreu pode ser transmitido como um único bit (disparou = '1', não disparou = '0') , como mostrado na figura 2.13. Isto ocorre porque a amplitude e duração do potencial de ação não são função do estímulo que o gerou [KANDEL et al., 2000], [BEAR et al., 2007].

Sistemas artificiais que tem como modelo comportamentos observados no sistema nervoso podem utilizar pulsos de 1 ms para transmissão de informação, utilizando os tipos de codificação neural descritos. Um dos modelos de sistemas artificiais que têm como inspiração o sistema nervoso são as

Figura 2.13: *Potencial de Ação Biológico (acima) e seu equivalente digital, o Potencial de Ação Artificial (abaixo).*



Redes Neurais Artificiais (RNAs), que serão mais bem explicadas no capítulo 3. As RNAs que utilizam a temporização entre os *spikes* são as Redes Neurais Pulsadas (SNNs - *Spiking Neural Networks*), que também serão descritas no capítulo 3.

SNNs aproveitam o *spike-timing* individual, ou tempo de disparo individual, para codificação e processamento de informações. A associação de *spike-timing* e sincronismo nas SNNs levam a um ambiente computacional rico, capaz de lidar com a seleção de entradas, consolidação e combinação de informações aprendidas, entre outros [BUZSÁKI and DRAGUHN, 2004]. Devido à sua sua forma de operação, SNNs de tempo real requerem *spike trains*, ou trens de *spikes*, para operar adequadamente [GERSTNER and KISTLER, 2002], [BROWN et al., 2004]. Assim, a informação apresentada à rede deve ser transformada em trens de *spikes* antes de serem processadas pela rede.

Neste sentido, interfaces são necessárias a fim de converter os fenômenos físicos, ou dados simulados, para trens de *spikes* antes de alimentar as SNNs. Além disso, além de converter informação em trens de *spikes*, também é desejável que tais interfaces possam restringir as informações convertidas dentro de algum esquema de 'código'. Quando essas informações alimentarem sistemas que trabalham em tempo real, essa conversão e codificação necessitam ser feita também em tempo real.

Na Engenharia Eletrônica são utilizados diversos sensores que fazem papel similar aos sensores

encontrados no corpo humano. Os sensores eletrônicos podem transduzir sinais do mundo (luz, som, temperatura, etc.) em tensão ou corrente proporcional, mudanças na resistência elétrica de um material, etc. Por outro lado, os sensores biológicos transduzem esses sinais externos em *spikes*. Sistemas artificiais que tentam modelar o sistema nervoso de forma mais plausível biologicamente necessitam que a informações sejam convertidas em *spikes* para serem processadas.

Na Tabela 2.1 são mostrados exemplos de sensores comerciais encontrados na eletrônica que transduzem grandezas físicas, separados pela classificação utilizada na seção 2.3. Também é possível ver na tabela o tipo de resposta mais encontrada para cada tipo de sensor.

Tabela 2.1: *Exemplos de Equivalentes Eletrônicos para os Sensores Biológicos.*

Sensor Biológico	Equivalente Eletrônico	Exemplos e Referências	Tipo de Resposta
QUIMIORRECEPTORES			
Gustativo	Eletrodo de Íon Específico	[FERNANDES et al., 2001], [ELIT, 2015]	tensão, corrente
Olfativo	Sensor de Gás	[HANWEI, 2015], [HONEYWELL, 2011]	tensão, resistência
FOTORRECEPTORES			
	Fotodiodo	[HONEYWELL, 2009]	corrente
	Fototransistor	[EVERLIGHT, 2012], [HONEYWELL, 2012]	corrente
	CMOS sensor	[SEMICONDUCTOR, 2013]	carga em capacitores
	CCD - *charge-coupled device*	[PANASONIC, 2005]	carga em capacitores
MECANORRECEPTORES			
Auditivo	microfone dinâmico de bobina móvel	[KNOWLES, 2005a]	corrente
	microfone piezoelétrico	[KNOWLES, 2005b]	corrente
	microfone de eletreto	[CUI, 2013]	corrente
Tátil	*strain gauge*	[VISHAY, 2010]	resistência
	Sensor Capacitivo	[ATMEL, 2013], [ON, 2013]	carga em capacitores
TERMORRECEPTORES			
	Termistor (NTC e PTC)	[MURATA, 2014], [VISHAY, 2008], [EPCOS, 2014]	resistência
	Par Diferencial	[TEXAS, 2015]	tensão
	Termopar	[SPECTRUM, 2009], [OMRON, 2011]	tensão

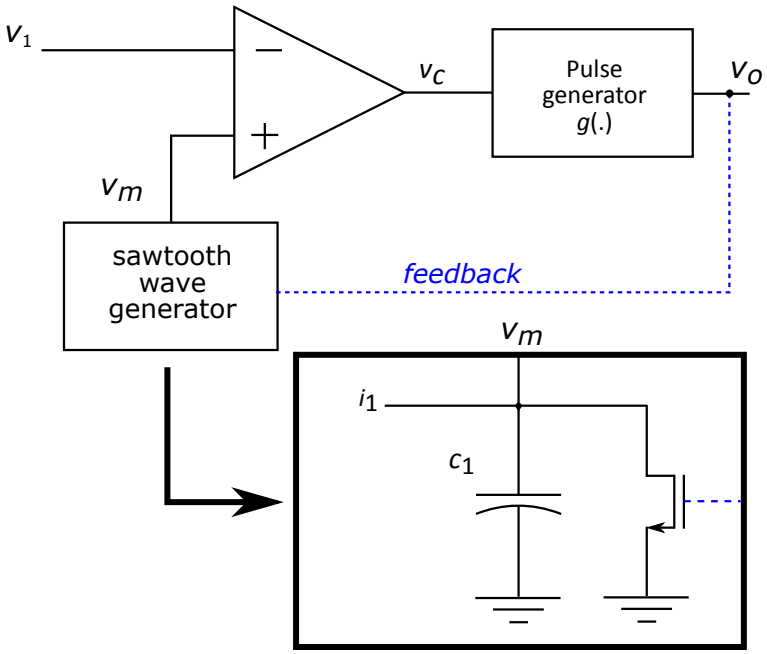
2.5.1 UM EQUIVALENTE ARTIFICIAL DO NEURÔNIO SENSOR

Na Figura 2.14 é mostrado o esquema elétrico de um circuito analógico que funciona como transdutor de sinais de corrente ou tensão em *spikes*. Este circuito proposto por *Saldaña-Pumarica et al* [PUMARICA et al., 2007], [PUMARICA and DEL-MORAL-HERNÁNDEZ, 2007] é composto de um gerador de sinal dente de serra (v_m) formado pela corrente i_1 e o capacitor c_1, além de um circuito comparador. O circuito comparador compara a onda dente serra com uma tensão de referência v_1 gerando uma tensão v_c. Essa tensão v_c aciona o bloco gerador de pulsos $g(.)$. O sinal v_o funciona

como saída do circuito e como sinal o *feedback* que descarga o capacitor c_1.

O circuito pode funcionar sensível a corrente i_1, onde v_1 é usado como tensão de referência, ou sensível a tensão v_1, neste caso, i_1 é usada como corrente de referência. No funcionamento sensível a tensão, i_1 e c_1 geram uma onda dente de serra de inclinação fixa. Quando a tensão v_m se torna maior que v_1, a tensão v_c fica positiva e ativa $g(.)$. A taxa de *spikes* é controlada pelo sinal de *feedback*, v_o, pois, quanto maior o tensão v_1, maior tempo Δt para que a v_m fique maior que v_1. Com isso, a taxa de disparos é inversamente proporcional a tensão de entrada.

Figura 2.14: *Esquema do circuito do Neurônio Sensor Analógico (Figura adaptada de [PUMARICA et al., 2007]).*



Quando o circuito está sendo utilizado com entrada em corrente, a tensão v_1 é fixa. Neste caso a inclinação da rampa da função dente de serra é variável e controlada pela corrente i_1. Quanto maior a corrente i_1, mais rápido a tensão v_m será maior que v_1, ou seja, um menor Δt. Como ocorre no caso anterior, v_c ativa a função $g(.)$ que gera o pulso. O sinal de *feedback*, v_o, descarrega c_1. Logo, a taxa de pulsos é diretamente proporcional a corrente de entrada i_1.

2.5.2 Uma Alternativa na Eletrônica Digital

Uma das vantagens do neurônio sensor artificial mostrado na seção anterior é o circuito gerar uma taxa de *spike* proporcional ao sinal de entrada (corrente ou tensão). Pelo motivo de, como é possível ver na Tabela 2.1, grande parte dos sensores comerciais terem como resposta sinais de corrente ou

tensão elétrica. No entanto, para cada neurônio sensor é necessário um fio de entrada e um de saída, sendo isto um limitante do número de neurônios sensores possível por encapsulamento. Para um número elevado de sensores, o circuito torna-se inviável fisicamente de ser implementado em uma única pastilha de silício, dado o grande número de pinos de entrada e saída que o circuito integrado precisaria ter.

Uma alternativa encontrada é a Eletrônica Digital, domínio em que grande número de sinais paralelos pode ser trocado pela comunicação serial desses dados, ao menos para a saída dos circuitos, uma vez que cada sensor de entrada ainda requer uma conexão com o sensor. Circuitos digitais podem ser utilizados para agrupar sinais vindos de vários sensores, e criar pacotes de dados e os disponibilizar serialmente na rede, diminuindo o número de fios utilizados. A conversão de diferentes grandezas do mundo físico para uma mesma linguagem de sinais digitais possibilita a utilização de diferentes sensores comerciais, diminuindo o custo do sistema e tornando o sistema mais flexível. Também existe a possibilidade de embutir codificação nos dados, agrupando neurônios para que transmitam conjuntamente uma informação (codificação temporal e por populações).

No entanto, para ser possível passar o problema do domínio analógico para o digital é necessária a conversão dos sinais analógicos para sinais digitais, algo que será abordado nas seções seguintes.

2.6 DOS FENÔMENOS AOS NÚMEROS

Na eletrônica, fenômenos no mundo físico geralmente são convertidos em sinais elétricos contínuos no tempo e amplitude. Esses sinais são denominados sinais analógicos. Assim, é necessário circuitos que funcionem como conversores dessas grandezas físicas em sinais digitais para que sejam reconhecíveis pelos sistemas digitais e possam ser computados. O circuito que faz essa conversão é conhecido como Conversor Analógico-Digital, Conversor AD ou simplesmente ADC do inglês *Analog-to-Digital Converter*. Nos dias atuais, os circuitos ADC são encontrados como periféricos de praticamente todo microcontrolador (uC), como nos exemplos mostrados na Tabela 2.2. Além de serem encontrados como dispositivos discretos [ANALOG, 2014], [LINEAR, 2005], [TEXAS, 2011].

Existem vários tipos diferentes de circuitos ADC, dos quais serão mostrados nas seções seguintes alguns tipos mais comuns encontrados comercialmente. Mas antes, será descrito o circuito como uma caixa preta, onde será descrito apenas os sinais de entrada e saída desejados, sem descrever a arquitetura utilizada internamente no conversor. Existe uma extensa literatura sobre a conversão de sinais analógicos para digitais, como [ZUMBAHLEN, 2008], [PLASSCHE, 2003], etc., e várias

Tabela 2.2: *Exemplos de Famílias de Microcontroladores Comercias com ADCs.*

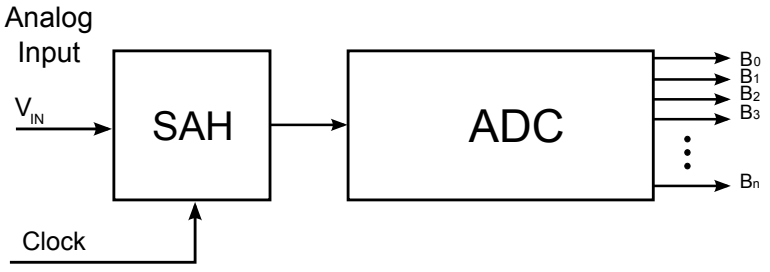
uC Family	Referência	circuito ADC	Freq. Máx.(MHz)	Periféricos de Comunicação
TEXAS INSTRUMENTS				
MSP430	[TEXAS, 2013b]	12 bits, 200ksps	16	UART, I2C, SPI, USB
ARM Tiva C	[TEXAS, 2013c]	12 bits, 1 Msps	80	UART, I2C, SPI, USB
C2000	[TEXAS, 2013a]	12bits, 25 Msps	300	UART, I2C, SPI, USB, ETHERNET
FREESCALE				
S12	[FREESCALE, 2008]	16 bits	50	UART, I2C, SPI, USB
Kinetis L ARM	[FREESCALE, 2012]	16 bits	48	UART, I2C, SPI, USB
Kinetis K ARM	[FREESCALE, 2012]	16 bits	150	UART, I2C, SPI, USB, ETHERNET
NXP				
Cortex M0	[NXP, 2011a]	10 bits, 400 ksps	50	UART, I2C, SPI, USB
Cortex M4	[NXP, 2011b]	10 bits, 400 ksps	180	UART, I2C, SPI, USB, ETHERNET
MICROCHIP				
PIC 16 bits	[MICROCHIP, 2013]	12 bits, 500 ksps	48	UART, I2C, SPI, USB
PIC 32 bits	[MICROCHIP, 2012]	10 bits	80	UART, I2C, SPI, USB, ETHERNET

arquiteturas de circuitos ADC, que, por não serem o foco deste trabalho, não serão descritos

2.6.1 O Conversor como uma Caixa Preta

O ADC básico é mostrado na Figura 2.15. O circuito conversor AD discretiza um sinal tanto no tempo como na amplitude, gerando o que é chamado de sinal digital. A maior parte dos circuitos ADC também inclui alguma circuitaria de suporte, como um oscilador gerador de *clock*, um sinal de referência (R_{ref}), uma função de *Sample-And-Hold* (SAH) e registradores internos e de saída.

Figura 2.15: *Esquema em blocos do ADC básico, visto como uma caixa preta*



O SAH é encontrado nos circuitos ADC síncronos, ele serve para estabilizar o sinal de entrada durante o tempo em que o conversor realiza a conversão. Ele funciona tomando uma amostra do sinal de entrada (*sample*) e retendo/memorizando este valor analogicamente (*hold*). Um circuito

SAH simples pode ser feito com uma chave analógica, quando fechada, o sinal de entrada carrega um capacitor. Quando aberta a chave, o capacitor mantém a tensão de entrada. Esse capacitor é ligado a um *buffer* de onde a tensão é lida.

Gerando o Valor Digital

Para uma tensão de entrada V_{IN}, o valor digitalizado dessa tensão, D_{out}, é dado pela equação 2.1, onde q_e é o erro de quantização, gerado pela diferença entre os "degraus" do valor digital. Esse "degrau" é dado por R_{ref}, o valor de referência do circuito. Para geração do esse valor de referência no circuito prático, pode ser utilizada a tensão de alimentação, uma fonte de corrente, carga de um circuito capacitivo, etc.

$$D_{out} + q_e = \frac{V_{IN}}{R_{ref}} \tag{2.1}$$

Como D_{out} é um valor digital de n bits, D_{out} pode ser escrito em função dos bits de saída B_m, segundo a equação 2.2.

$$D_{out} = \sum_{m=0}^{n-1} B_m 2^m \tag{2.2}$$

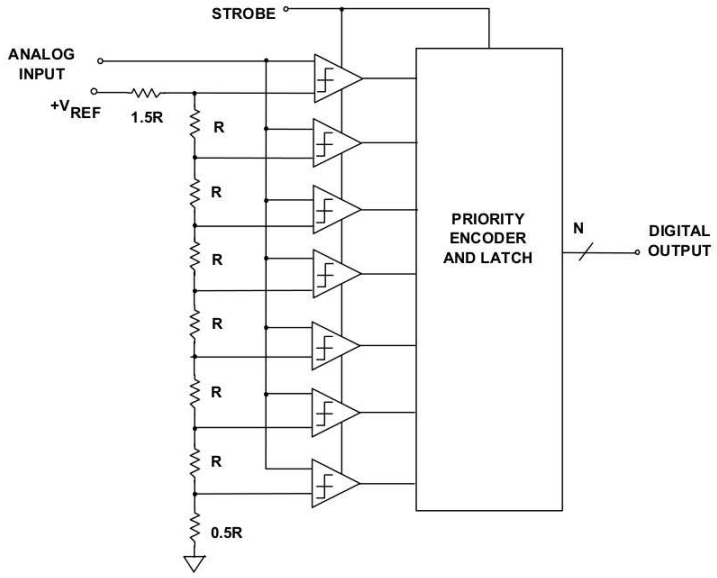
Além do erro de quantização, que é tão menor quando o valor de R_{ref}, deve ser levado em consideração a frequência do sinal que está sendo amostrado. A maior frequência do sinal amostrado tem que ser no mínimo duas vezes menor que a taxa de amostragem para não ocorrer *aliasing*, obedecendo a frequência de Nyquist (para maiores detalhes ver [OPPENHEIM et al., 1999]). Então, o uso adequado de um dado ADC deve levar em consideração que o sinal amostrado deve ser banda limitada (a energia do sinal esta contida em um faixa de frequência finita) e que a taxa de amostragem mínima para o sinal é obedecida. Em muitos casos é aconselhável o sinal passar por um filtro passa-baixas antes de ser convertido, garantindo que o sinal não sofrerá distorções devido à discretização.

2.6.2 CONVERSOR *Flash*

Conversor *Flash*, ou conversor paralelo, funciona graças a uma malha de comparadores ligadas a uma tensão de referência. A tensão de referência é dividia através de resistores, como mostrado na Figura 2.16. Um circuito codificador de prioridades analisa as saídas dos comparadores e gera uma saída digital com 'n' bits, para um número de comparadores igual a $2^n - 1$.

Como ele funciona paralelamente, a resposta é bastante rápida, por isso o nome conversor *flash*. Isso ocorre porque o tempo de resposta do circuito é devido apenas à demora de propagação do sinal

Figura 2.16: *Esquema do circuito ADC flash (Figura adaptada de [ZUMBAHLEN, 2008])*



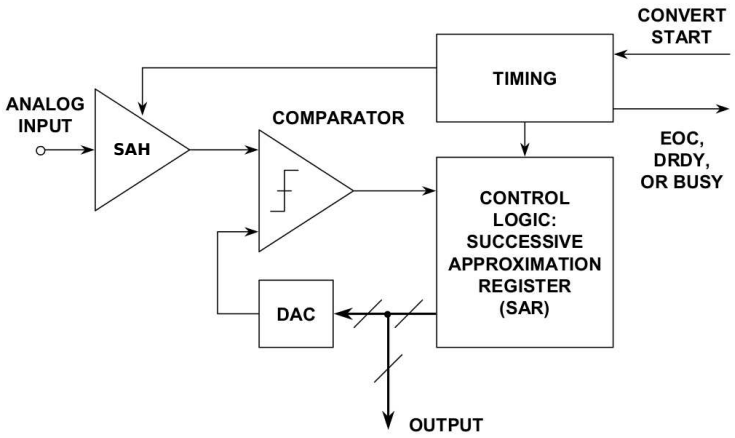
através dos transistores do circuito. No entanto, existe o problema do tamanho do circuito que cresce exponencialmente com o número de bits da saída. Por exemplo, para um número de bits $n = 8$, são necessários 255 comparadores; porém, quando o valor de 'n' é aumentado para 16 o número de comparadores sobe para 65.535.

2.6.3 CONVERSOR DE APROXIMAÇÕES SUCESSIVAS

O circuito ADC de aproximações sucessivas tem sido o principal circuito ADC utilizado comercialmente por muitos anos. Implementações mais recentes permitem circuitos que trabalham na faixa de Mega-Hertz [ANALOG, 2014]. Ele utiliza um circuito DAC (do inglês *Digital-to-Analog Converter*) para aproximar o valor analógico lido de um valor digital. O digrama de blocos do circuito pode ser visto na Figura 2.17. Ao contrário do circuito ADC *flash*, o circuito ADC de aproximações sucessivas é um circuito síncrono, necessitando de um sinal *clock* para seu funcionamento, e inicia a conversão através de um sinal de início, `CONVERT START`.

Ao ser iniciada a conversão, o sinal de entrada (`ANALOG INPUT`) passa pelo circuito SAH. Com o valor amostrado, o bloco de controle começará a aproximar um valor digital guardado no *Successive Approximation Register* (SAR). Começando com bit mais significativo do SAR em nível lógico '1', o conversor utiliza o valor do SAR para gerar um sinal analógico através do DAC, e o compara com o valor guardado no SAH. A resposta do comparador (`COMPARATOR`) irá indicar se aquele bit permanecerá em '1' ou não. No próximo pulso de *clock* o processo é repetido para o

Figura 2.17: *Esquema do circuito ADC Aproximações Sucessivas (Figura adaptada de [ZUMBAHLEN, 2008])*



próximo bit do SAR, caminhando do bit mais significativo para o menos significativo. O processo é repetido até que o valor analógico gerado pelo DAC seja 'igual' (levando em consideração o erro de quantização) ao valor guardado no SAH. Ou seja, a conversão demorará no máximo 'n' pulsos de *clock*, onde 'n' é número de bits do conversor, tornando o circuito suficientemente rápido.

2.6.4 CONVERSOR DE RAMPA

Também conhecido como conversor tipo realimentação, o circuito do conversor de rampa tem seu digrama de blocos mostrado na Figura 2.18. Ao iniciar a conversão, o gerador de rampa é inicializado junto a um contador. Quando o valor da rampa chega ao valor de entrada o contador é parado e o valor do contador será o valor digital do sinal de entrada.

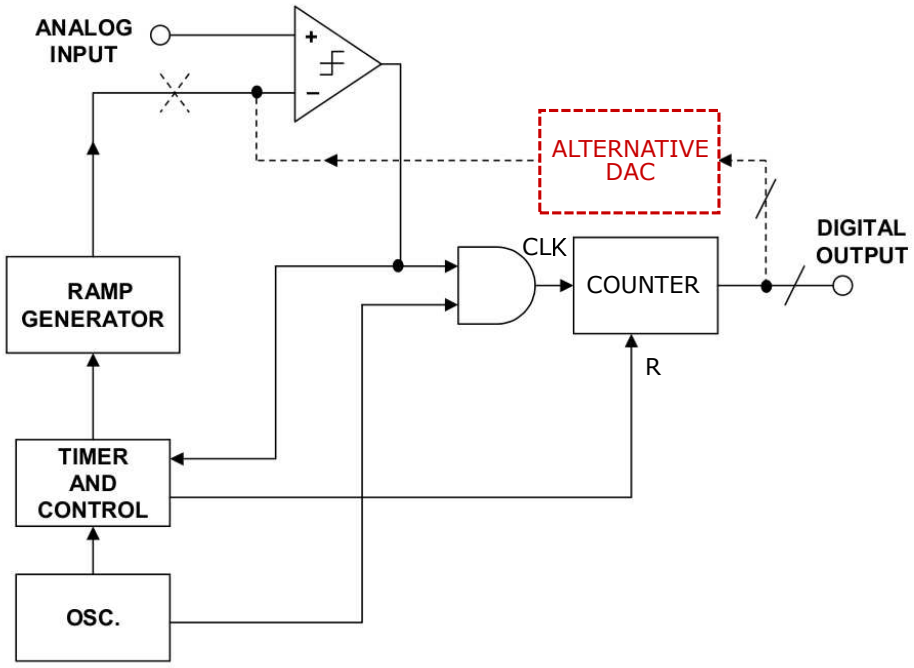
Um circuito alternativo (mostrado tracejado na Figura 2.18) é obtido ao troca o circuito gerador de rampa por um DAC, cuja entrada é o valor contido no contador. A vantagem do uso da rampa é que o ADC será sempre monotônico. No circuito alternativo em que é usado o DAC, essa característica será herdada do DAC, se esse é monotônico ou não.

Mesmo sendo um tipo de conversor bem preciso, ele tende a ser lento, devido ao número de pulsos de *clock* necessários para converter o sinal. A precisão do circuito depende então da precisão do gerador de rampa, ou DAC, bem como do oscilador utilizado para geração do *clock*.

2.6.5 CONVERSOR $\Sigma\Delta$

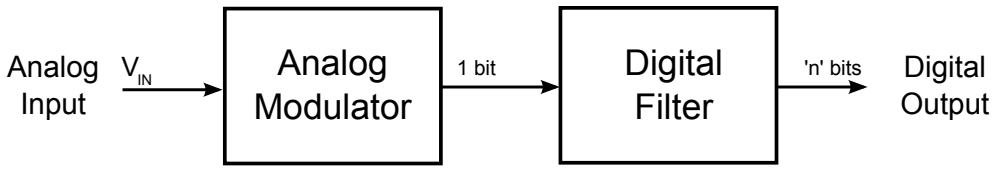
O conversor $\Sigma\Delta$ também é conhecido como ACD por sobreamostragem (*oversampling*). Ele recebe esse nome justamente por utilizar uma taxa de amostragem muito maior que a taxa de Nyquist

Figura 2.18: *Esquema do circuito ADC de Rampa (Figura adaptada de [ZUMBAHLEN, 2008])*



(64 vezes maior, por exemplo) [ZUMBAHLEN, 2008]. O circuito pode ser dividido em duas partes: um modulador analógico e um filtro digital, como mostrado na Figura 2.19.

Figura 2.19: *Esquema do circuito ADC* $\Sigma\Delta$.



O modulador é responsável por sobreamostrar o sinal. Ele funciona como um conversor AD de 1 bit que trabalha com uma frequência bem maior que a do sinal que está sendo amostrado. O que é visto na saída desse circuito ao logo do tempo é uma cadeia de bits que é a versão digital do sinal de entrada. Esse sinal serial é a entrada do filtro digital. O Filtro digital do ACD $\Sigma\Delta$ é um filtro passa baixas seguido de um decimador. O filtro passa-baixas retira o erro de quantização, que fica na fronteira da banda do espectro do sinal sobreamostrado. Após a retirada do erro, o decimador reduz a taxa de amostragem do sinal, gerando, então, o sinal de saída do circuito.

Como dito anteriormente, diferentes arquiteturas de circuitos ADC podem ser encontradas na literatura, incluindo variações das que foram mostradas neste capítulo. Com o uso de circuitos ADC é possível levar da eletrônica analógica para eletrônica digital o problema da geração de trens de

pulsos para alimentar SNNs de tempo real. Um sistema digital capaz de gerar esses trens de pulsos nas três classes de codificação descritas na literatura da neurociência é descrito no capítulo 4.

CAPÍTULO 3

REDES NEURAIS ARTIFICIAIS

3.1 INTRODUÇÃO

TENDO em vista o número de diferentes áreas do conhecimento que estão diretamente e indiretamente ligadas à Engenharia, o capítulo tenta situar este projeto entre essas áreas do conhecimento. Este processo será feito partindo da área do conhecimento da qual o projeto faz parte e enveredando por tópicos mais específicos. Primeiramente, definindo Inteligência Artificial e suas áreas de atuação, depois, discutindo uma importante parte desse campo que são as Redes Neurais Artificiais (RNAs). Será feita uma introdução cronológica das RNAs até chegar, por fim, a mais nova geração de Redes Neurais, as Redes Neurais Pulsadas. Este projeto de mestrado está inserido em um tópico particular das Redes Neurais Pulsadas, as Assembleias Neurais Pulsadas. Sendo a meta final deste capítulo elucidar o que são essas assembleias neurais e seus usos.

3.2 INTELIGÊNCIA ARTIFICIAL

De acordo com *Russell* e *Norvig*, Inteligência Artificial (IA) é uma ciência bastante recente, o começo dos estudos se deu após a Segunda Guerra Mundial e o nome propriamente dito foi criado apenas em 1956. A IA partilha conhecimentos com diversas outras ciências. Existem aplicações da IA em: aprendizagem e percepção, jogos, demonstração de teoremas matemáticos, engenharia de controle, reconhecimento de linguagem natural ou mesmo criação de poesias, músicas, etc. Mesmo estando ligada a várias outras ciências, a IA é estudada classicamente dentro das Ciências da Computação e é intimamente ligada a Engenharia e Matemática [RUSSELL and NORVIG, 2009].

Pode-se encontrar algumas diferentes definições para IA, mas a mais utilizada em engenharia, e a que este trabalho irá utilizar, é que "Inteligência Artificial é o estudo do projeto de agentes inteligentes" [POOLE et al., 1998]. Ou em melhor termos, é o estudo do projeto de **agentes** que **agem racionalmente** [RUSSELL and NORVIG, 2009]. O termo **agente** vem do latim "*agere*" e significa "algo que age". Então, um **agente racional** seria aquele que age para alcançar o melhor resultado possível ou, quando existe incerteza, o melhor resultado esperado. A IA tenta entender como pensamos e utilizar esse conhecimento para criar entidades inteligentes, trabalhando associada à Neurociência, Ciências Cognitivas, Psicologia, Fonética, etc.

Uma das abordagens da IA é a IA simbólica, ou cognitiva. Ela estuda os agentes reativos (reativos simples e baseados em modelos) e os agentes cognitivos (baseados em objetivos ou baseados em utilidades). Essa abordagem da IA estuda como os agentes raciocinam e como representam estados do mundo de modo a resolver problemas, podendo antecipar estados do mundo para tomada de decisão (para maiores detalhes ver [RUSSELL and NORVIG, 2009]). Uma outra abordagem é a IA conexionista, onde o objeto de estudo é o sistema nervoso, o neurônio e suas conexões.

Para a investigação dessas abordagens a IA utiliza-se de modelos, entre eles têm-se os Algoritmos Genéticos, Lógica Fuzzy, Sistemas Baseados em Conhecimento, Raciocínio Baseado em Casos, RNAs, etc. [RUSSELL and NORVIG, 2009], [ROJAS, 1996], [ROSS, 2010] .

Nas próximas seções serão explicadas as características das RNAs, que fazem parte da abordagem conexionista da IA, na qual este trabalho está inserido.

3.3 REDES NEURAIS ARTIFICIAIS

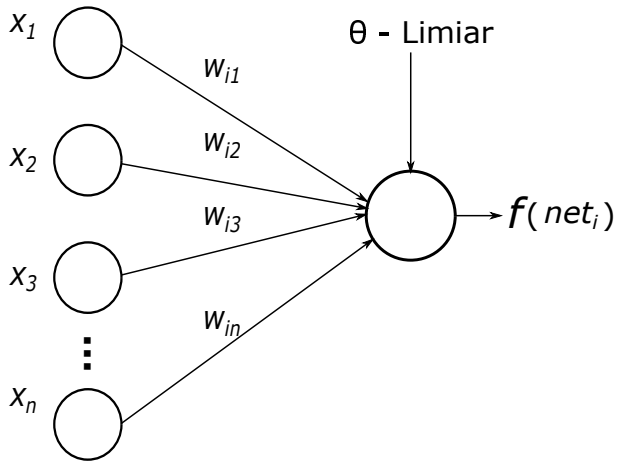
As RNAs partem da união entre estudos que englobam como o sistema nervoso funciona e o pensamento que, com esse conhecimento, é possível criar agentes inteligentes. Tentando definir de forma abrangente, as RNAs são sistemas conexionistas que tentam mimetizar o funcionamento das redes neurais naturais. Classicamente, elas partem de três características principais encontrados nas redes neurais naturais: como **neurônios computam**, como **suas conexões influenciam nisso** e **como o sistema nervoso aprende**. Essas três características têm sido estudadas tanto em conjunto como separadamente. Mas, de uma forma geral, os avanços em cada uma delas são compartilhados.

3.3.1 O NEURÔNIO ARTIFICIAL

As RNAs necessitam utilizar modelos artificiais dos neurônios biológicos. O primeiro modelo de neurônio artificial foi proposto por *MacCulloch* e *Pitts* (M&P) e era bastante "rústico", mas foi

a base para os demais modelos [VALENÇA, 2010]. A Figura 3.1 mostra o diagrama esquemático de um modelo M&P. No modelo as variáveis $x_1, x_2, x_3, ..., x_n$ indicam as entradas do neurônio 'i'. Elas representam a ligação do axônio de um neurônio pré-sináptico 'j' ao dendrito do neurônio pós-sináptico 'i'. O peso sináptico é dado por w_{ij}. Já o sinal net_i é dado pela equação 3.1, onde Θ representa o limiar de excitação. Este limiar influência a saída do neurônio e comporta-se como uma constate na equação 3.1.

Figura 3.1: *Neurônio do M&P com limiar explícito.*



$$net_i = \sum_{j=1}^{n} w_{ij} x_j - \Theta \tag{3.1}$$

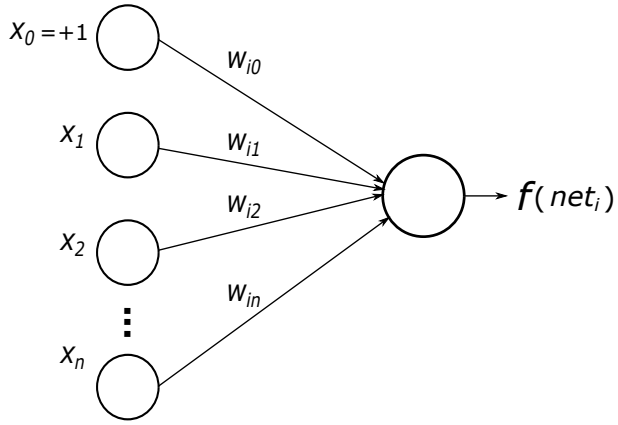
A função de ativação 'f' determina quando o neurônio vai "disparar". Para o modelo do M&P, 'f' é a função degrau mostrada na Equação 3.2. É fácil notar a simplificação em relação à resposta do neurônio biológico mostrada na seção 2.2. Enquanto os neurônios naturais emitem um pulso de duração fixa de aproximadamente 1ms, os neurônios de M&P geram uma saída digital que permanece ativa até a próxima iteração da rede.

$$f(net_i) = \begin{cases} 1 & \text{se} \quad net_i \geq 0 \\ 0 & \text{se} \quad net_i < 0 \end{cases} \tag{3.2}$$

No entanto, é possível modificar o modelo mostrado na Figura 3.1 pelo mostrado na Figura 3.2. No modelo da Figura 3.2 foi adicionada uma entrada x_0 que sempre está em nível lógico alto, '1'. Enquanto o peso sináptico $w_{i0} = -\Theta$. Com isso a equação 3.1 pode ser reescrita como a equação 3.3:

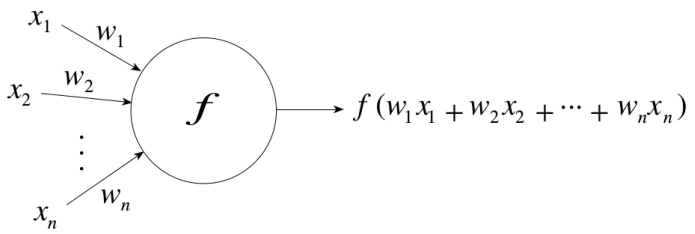
$$net_i = \sum_{j=0}^{n} w_{ij} x_j \tag{3.3}$$

Figura 3.2: *Neurônio do M&P com limiar implícito*



Um esquema mais geral de mostrar um neurônio artificial é o utilizado pelo *Rojas* [ROJAS, 1996], mostrado na Figura 3.3. Ele é composto de de entradas 'x_j' que indicam ramos dos dendritos que estão efetivamente ligadas a um axônio de um neurônio pré-sináptico. O peso sináptico é dado pela coleção de valores 'w_j', cada um relacionado ao respectivo 'x_j'. No "corpo celular" existe a função de ativação 'f' do neurônio, onde 'f' é função de $x_j w_j$ para todo j ligado ao neurônio. As mudanças dos modelos de neurônio basicamente surgem em modificações na função de ativação do neurônio. Modificações no modelo de neurônio ocorreram tanto para tornar possível a implantação de certos algoritmos de aprendizado, quanto para trazer maior plausibilidade biológica para o modelo [BRAGA et al., 2009].

Figura 3.3: *Esquema geral do neurônio artificial.*

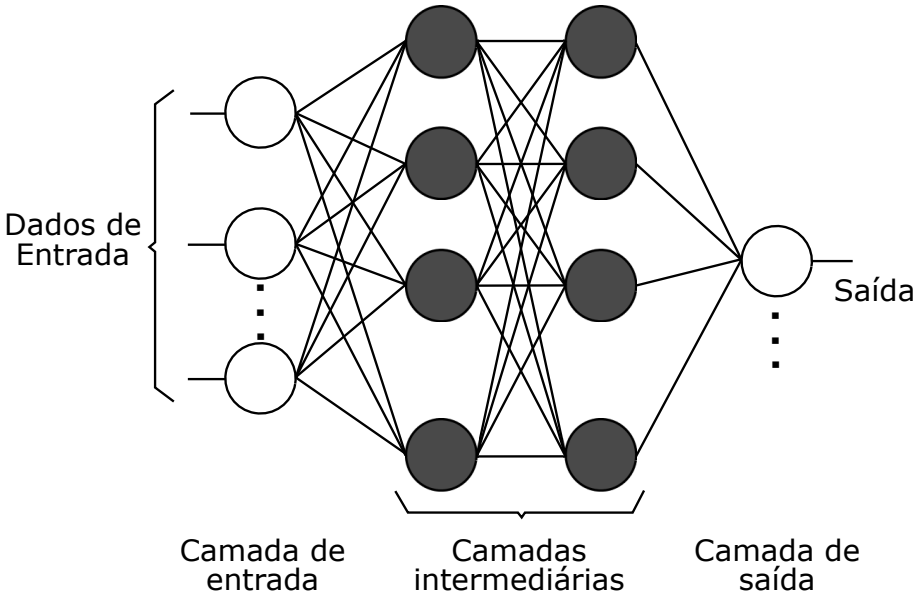


3.3.2 A Arquitetura da Rede

Outra característica herdada, ou imitada, das redes neurais biológicas é o paralelismo. Os neurônios artificiais mostrados na seção anterior necessitam ser interconectados para que funcionem como

uma rede propriamente dita. Classicamente as RNAs são interconectadas como mostrado na Figura 3.4, na qual a rede é dividida em camadas (*layers*) de neurônios: O *layer* de entrada, *layers* intermediários e o *layer* de saída. Cada neurônio é chamado de um nó da rede.

Figura 3.4: *Esquemático de uma RNA mostrando a arquitetura em camadas de neurônios.*



Os parâmetros que definem a arquitetura da rede são o **número de camadas da rede**, o **número de nós em cada camada**, **tipo de conexão entre os nós** e a **topologia da rede**, como a rede mostrada na Figura 3.4.

Em relação aos tipos de conexão entre os nós, exitem as **redes *feedforward***, também chamadas de redes acíclicas, onde a saída de qualquer nó de um *layer* apenas pode fazer conexões com a entrada de nós de *layers* posteriores. Ou seja, não é permitida conexões com nós de *layers* anteriores ou nós do próprio *layer*. Outro tipo de conexão possível é o encontrado nas **redes *feedback***, ou redes cíclicas. Nesse tipo de rede a limitação da ligação entre nós da rede *feedforward* não existe.

Quanto à conectividade da rede, é possível a divisão em dois tipos de rede: as **redes parcialmente conectadas** e as **redes completamente conectadas**. Como os nomes sugerem, nas redes parcialmente conectadas não existem ligações entre todos os neurônios. Já na rede completamente conectada, todos os neurônios estão interligados.

3.3.3 TIPOS DE APRENDIZADO EM RNAS

De uma forma geral, é possível definir aprendizagem em RNAs como: “Aprendizagem é o processo pelo qual os parâmetros de uma rede neural são ajustados através de uma forma continuada de

estímulo pelo ambiente no qual a rede está operando, sendo o tipo específico de aprendizagem realizada definido pela maneira particular que ocorrem os ajustes realizados nos parâmetros" [MENDEL and MCLAREN, 1970].

A mudança de parâmetros descrita no trabalho de *Mendel* e *McLaren* está relacionada à mudança dos pesos sinápticos das conexões entre os neurônios e/ou o ajuste das conexões [BRAGA et al., 2009]. Um conjunto de procedimentos bem-definidos para adaptar os parâmetros de uma RNA para que a mesma possa **aprender** uma determinada função é chamado de **algorítimo de aprendizado** [BRAGA et al., 2009]. Existem diversos algoritmos para aprendizado de RNAs que diferem basicamente nas regras de ajuste dos parâmetros. É possível classificar os tipos de aprendizado em **aprendizado supervisionado** e **aprendizado não-supervisionado** [ROJAS, 1996].

Aprendizado Supervisionado

No aprendizado supervisionado existe um supervisor, também chamado de professor. Este supervisor é externo à rede e tem o conhecimento das entradas e saídas desejadas para a rede. O aprendizado supervisionado é o mais comum nas RNAs, onde o ajuste se dá na tentativa de encontrar parâmetros para rede que façam os pares de entrada e saída serem obedecidos. Isso é feito disponibilizando as entradas na rede e verificando as saídas encontradas. A saída da rede é comparada com a saída ideal e é gerado um ajuste dos parâmetros da rede que visa à aproximação da saída da rede com a saída ideal. Esse processo de ajuste é feito gradualmente, em pequenos passos ao longo do treinamento, visando minimizar, geralmente, o erro médio quadrático entre as saídas encontradas e as desejadas. O processo é repetido até que o erro tenha valor menor ou igual ao desejado para função, quando possível (é uma boa prática configurar um número máximo de iterações). Dentre as várias regras de aprendizado, as mais conhecidas são a **regra delta** [WIDROW and HOFF, 1960] e o algoritmo de ***backpropagration*** [RUMELHART et al., 1988], que comporta-se como uma generalização da regra delta para redes de múltiplas camadas.

Aprendizado Não-Supervisionado

No aprendizado não-supervisionado não existe o papel do supervisor externo. Nesse sistema de aprendizado, existe apenas um conjunto de entradas que descreve o ambiente externo, essas entradas que serão apresentadas à rede. À medida que esses dados são mostrados à rede por repetidas vezes, a rede detém automaticamente uma representação interna dos dados, criando novas classes ou grupos. Para isso é necessário que no grupo de entradas exista alguma correlação entre subgrupos de dados,

não sendo necessário que sejam pré-conhecidos. Uma das qualidades desse tipo de aprendizado é justamente descobrir padrões em grupo de dados que seriam difíceis de serem encontrados.

O sistema utilizado para realizar o aprendizado não-supervisionado pode adquirir várias formas possíveis. Ele pode ser uma camada na entrada ou na saída da rede, ou conexões *feedforward* da entrada para saída, ou mesmo conexões laterais entre os neurônios da camada de saída. Em todos os casos, o processo de aprendizado consiste em modificar repetitivamente o peso sináptico de todas as conexões do sistema em resposta às entradas. Uma forma bem conhecida de aprendizado não-supervisionado é o aprendizado herbbiano, desenvolvido a partir dos trabalhos de *Hebb* [HEBB, 2002], onde a correlação temporal entre a saída do neurônio pré-sináptico e do neurônio pós-sináptico é utilizada para reforçar ou enfraquecer a ligação entre eles. Outro modelo de aprendizado é o de *Linsker* [LINSKER, 1988], baseado nos trabalhos de *Hebb* [HEBB, 2002], o modelo de *Linsker* tenta imitar os primeiros estágios do sistema visual de mamíferos [BRAGA et al., 2009].

3.4 UMA BREVE INTRODUÇÃO CRONOLÓGICA

É possível dividir os estudo em RNA em três gerações [MAASS, 1997]. Essa divisão proposta por *Maass* em 1997 leva em consideração o tipo de função de ativação que é utilizada no modelo do neurônio. Como dito na seção 3.3.1 deste capítulo, a função de ativação foi ao longos dos anos modificada, principalmente para se adequar a estratégias de aprendizado e usabilidade das redes, muitas vezes deixando de lado o rigor com a plausabilidade biológica. Com isso é possível, cronologicamente, dividir a história das RNAs nessas três gerações e descrever os avanços que se deram em cada geração.

3.4.1 O INÍCIO - PRIMEIRA GERAÇÃO DE RNAS

A história das RNAs tem início antes mesmo do termo Inteligência Artificial (IA) começar a ser utilizado. O início dos estudos em redes neurais está ligado com o início da IA, pois os estudos na área de RNAs, junto com o trabalho de *Alan Turing* de 1950, onde ele apresentou o teste de Turing, aprendizagem de máquina, algoritmos genéticos e aprendizagem por reforço, geraram a massa crítica necessária para desenvolvimento dessa nova área de pesquisa [RUSSELL and NORVIG, 2009].

Aliando os estudos em neurociência com o conceito de cérebro lógico, o neurofisiologista *Warren McCulloch* e o matemático *Walter Pitts* propuseram em 1943 uma arquitetura de máquina inspirada no cérebro humano [MCCULLOCH and PITTS, 1943], dando início a história das RNAs e da IA. O modelo de neurônio usado por *MacCulloch* e *Pitts* foi descrito na seção 3.3.1 e utilizava uma função

de ativação degrau, gerando como saída apenas ‘zero’ ou ‘um’.

Ainda na mesma década, o psicólogo *Donald Hebb* propôs uma regra de aprendizado para as RNAs que tem sido a base para os algoritmos de aprendizado: quando um neurônio recebe um estímulo de outro neurônio, e se ambos estão ativos (estão disparando), o peso entre estes deve ser fortalecido, caso contrário, deve ser enfraquecido [HEBB, 2002]. Em 1960 *Widrow* e *Holff* apresentaram a academia uma regra de aprendizado para extensão do *Perceptron* chamada de ADALINE (*ADAptive LInear NEuron*) [WIDROW and HOFF, 1960]. Esta regra de aprendizagem ficou conhecida como **regra delta** e é utilizada até os dias atuais. Na regra delta o conjunto de pesos são otimizados através da utilização de uma técnica conhecida como método do gradiente descendente. Neste método é usado como função custo a ser minimizada o erro médio quadrático [VALENÇA, 2010].

Durante as décadas de 1950 e 1960 houve muita pesquisa em RNAs e muitos avanços na sua usabilidade em reconhecimento de padrões, havendo muitos investimentos na área. Um dos pesquisadores mais promissores foi *Marvin Minsky*, ele voltou seu trabalho aos limites das RNAs e 1969 junto com *Papert* publicou um livro intitulado ***Perceptrons*** [MINSKY and PAPERT, 1969] que, involuntariamente, fez a comunidade científica da época, e principalmente os financiadores de pesquisas, virarem as costas para as RNAs. No trabalho de *Minsky* e *Perpert* foi explicado que um neurônio artificial só poderia separar linearmente dois tipos de entradas. Por exemplo, apenas um neurônio não conseguiria reproduzir a função **ou-exclusivo**. Esse fato trouxe uma descrença aos pesquisadores da área, que junto à tradição logicista consideravam o cérebro lógico. Logo, as RNAs, e nesse caso particular o neurônio artificial, deveriam ser capazes de efetuar funções lógicas [RUSSELL and NORVIG, 2009].

3.4.2 A RETOMADA - SEGUNDA GERAÇÃO DE RNAS

Houve uma baixa produtividade de trabalhos na área de RNAs na década de 1970, devido principalmente a falta de investimentos na área após os trabalhos de *Minsky*. Uma exceção foram os trabalhos de *Kohonen*, que estudou durante esta década o conceito de mapas auto-organizáveis, que utilizam aprendizado não supervisionado [KOHONEN, 1988].

Os estudos em RNAs aumentaram novamente apenas na década de 1980 com os trabalhos do físico *John Hopfield* [HOPFIELD, 1982] sobre as propriedades associativas das redes neurais, desenvolvendo topologias de redes que mais tarde levariam seu nome, as Redes de Hopfield. Ele deu início a um campo que animou novamente a área das RNAs, que são as Redes Neurais Recorrentes,

onde existe a realimentação de sinais entre os *layers* da rede. A realimentação gera sistemas com memória, possibilitando descrever sistemas dinâmicos. Outro trabalho que ajudou nessa volta das RNAs foi a invenção do algoritmo ***Backpropagation***, com estudos iniciados na década de 1970 na tese de *Paul Werbos* [WERBOS, 1974], que ganharam popularidade apenas em 1986 com o trabalho de *Rumelhart* e *McClelland* [RUMELHART et al., 1988].

3.5 REDES NEURAIS PULSADAS - A TERCEIRA GERAÇÃO DE RNAS

Desde os trabalhos com as redes ADALINE a função de ativação f utilizada não era mais o degrau unitário do modelo do M&P. Essas redes utilizam mapeamento linear da entrada para saída. Posteriormente funções não lineares foram utilizadas, como as sigmóides, mostradas na equação 3.4, que são limitadas entre dois valores (zero e um) e são monotônicas. Com funções de ativação que possuem saídas em número reais foi possível agregar maior funcionalidade para as RNAs, como aproximar funções matemáticas relacionando estados de entrada com os de saída [BRAGA et al., 2009].

$$y(x) = \frac{1}{1 + e^{-x/\tau}} \tag{3.4}$$

Uma possível interpretação para neurônios com números reais como saída, bem diferente do potencial de ação biológico, considera que a saída do neurônio artificial representa a taxa de *spikes* no tempo, noção bem aceita desde o início do século XX, devido as trabalhos mostrando a correlação entre a frequência de potenciais de ação em receptores musculares em relação a foça realizada pelos músculos [LEVINE, 2007], [GERSTNER and KISTLER, 2002].

No entanto, trabalhos na década de 1980 [THORPE et al., 2001], [PERRETT et al., 1982], [THORPE et al., 1996], com dados experimentais de macacos, mostraram que o sistema visual desses primatas possui um tempo de resposta para estímulos no córtex entre 100 e 150 ms, atravessando ao menos 10 estágios sinápticos da retina até o lobo frontal. Em seguida trabalhos publicados apontaram que áreas corticais envolvidas no processamento visual conseguem computar todo o processo que lhe cabe entre 20 e 30 ms [ROLLS and TOVEE, 1994], [ROLLS et al., 2006]. Porém, a taxa de disparos para os neurônios envolvidos nessas computações concentram-se em frequências menores que 100 Hz. O curto tempo entre a recepção dos sinais e a computação dentro do córtex necessária para que um estímulo de resposta seja detectado, torna improvável que, para esses sistemas, o conceito da taxa de disparos neuronais representando grandezas.

Mesmo ainda sendo um campo de estudo aberto na neurociência como o processamento de informações e computação neural ocorrem [BRETTE, 2012], evidências biológicas empíricas mostraram que o tempo preciso em que ocorre o potencial de ação desempenha um papel importante na representação e computação no sistema nervoso [VANRULLEN et al., 2005]. Como consequência dessas descobertas em neurociência, muitos pesquisadores se concentraram nas Redes Neurais Pulsadas (SNNs, do inglês *Spiking Neural Networks*), a terceira geração de RNAs [GHOSH-DASTIDAR and ADELI, 2009], [PAUGAM-MOISY and BOHTE, 2010]. SNN utiliza o momento em que ocorre o *spike*, *spike-timing*, para codificação e processamento de informações. A associação do *spike-timing* com o sincronismo intrínseco das redes biológicas, podem trazer paras SNNs um ambiente computacional rico, capaz de lidar com a seleção de entradas, consolidação e combinação de informações aprendidas, informações atribuídas a conjuntos de neurônios, entre outros [BUZSÁKI and DRAGUHN, 2004].

Uma vez que as SNNs tentam trazer maior plausibilidade biológica para as RNAs, os modelos neurais utilizados também trazem maior similaridade com o neurônio biológico. Ao longo dos anos diversos modelos têm sido propostos, onde o custo computacional e o comportamento dinâmico devem ser levados em consideração na escolha do modelo a ser utilizado. A tabela mostrada na Figura 3.5, retirada de [IZHIKEVICH, 2004], mostra vários tipos de modelos de neurônio utilizados em SNNs. Na tabela é mostrado o custo computacional (em FLOPS - *FLoating-point Operations Per Second*) e os tipos de resposta que cada neurônio consegue realizar em relação ao neurônio biológico (mostrado na Figura 2.3).

Dentro das SNNs existem diferentes abordagens de como os neurônios interagem entre si gerando sistemas computacionais. Umas dessas abordagens é a Computação de Reservatório, ou *Reservoir Computing* (*Liquid State Machine* [MAASS et al., 2002], *Echo State MAchine* [JAEGER, 2001]). De forma geral, essa abordagem trabalha como se o sistema nervoso fosse um grande reservatório. Neste reservatório, os sinais, *spikes* de neurônios, reverberam constantemente, não tendo o usuário acesso as conexões da rede ou ao fluxo de sinais internos. Apenas as camadas de neurônio de entrada e/ou saída são treinadas para extrair ou inserir dados no reservatório.

Outra abordagem é o conceito de Computação por Assembleias Neurais (NAC, do inglês *Neural Assembly Computing*) [RANHEL, 2012c], que será tratado em maiores detalhes na seção seguinte.

Figura 3.5: *Custo computacional para cada modelo de neurônio e tipos de resposta que cada modelo consegue reproduzir (Figura retirada de [IZHIKEVICH, 2004]).*

Models	biophysically meaningful	tonic spiking	phasic spiking	tonic bursting	phasic bursting	mixed mode	spike frequency adaptation	class 1 excitable	class 2 excitable	spike latency	subthreshold oscillations	resonator	integrator	rebound spike	rebound burst	threshold variability	bistability	DAP	accommodation	inhibition-induced spiking	inhibition-induced bursting	chaos	# of FLOPS
integrate-and-fire	-	+	-	-	-	-	-	+	-	-	-	-	+	-	-	-	-	-	-	-	-	-	5
integrate-and-fire with adapt.	-	+	-	-	-	-	+	+	-	-	-	-	+	-	-	-	-	+	-	-	-	-	10
integrate-and-fire-or-burst	-	+	+		+	-	+	+	-	-	-	-	+	+	+	-	+	+	-	-	-		13
resonate-and-fire	-	+	+	-	-	-	-	+	+	-	+	+	+	+	-	-	+	+	+	-	-	+	10
quadratic integrate-and-fire	-	+	-	-	-	-	-	+	-	+	-	-	+	-	-	+	+	-	-	-	-	-	7
Izhikevich (2003)	-	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	13
FitzHugh-Nagumo	-	+	+	-		-	-	+	-	+	+	+	-	+	-	+	+	-	+	+	-	-	72
Hindmarsh-Rose	-	+	+	+			+	+	+	+	+	+	+	+	+	+	+	+	+	+		+	120
Morris-Lecar	+	+	+	-		-	-	+	+	+	+	+	+	+		+	+	-	+	+	-	-	600
Wilson	-	+	+	+			+	+	+	+	+	+	+	+	+	+		+	+				180
Hodgkin-Huxley	+	+	+	+			+	+	+	+	+	+	+	+	+	+	+	+	+	+		+	1200

3.6 ASSEMBLEIAS NEURAIS PULSADAS

Em 1982 *Abeles* propôs que os neurônios podem desempenhar o papel de "detectores de coincidência" em vez de integradores de média de *spikes* [KÖNIG et al., 1996]. Neurônios que operam em sincronismo naturalmente dão origem a *cell assemblies*, ou assembléias de células, essa ideia foi proposta por *Hebb* em 1949 [HEBB, 2002]. Redes *feedforward* com grupo de neurônios trabalhando em conjunto foram propostas por *Abeles* [ABELES, 2009]. Nessas redes, esse grupo de neurônios disparavam sincronamente, o que foi chamado de "*synfire chains*". Posteriormente, foi proposta a noção de "*synfire braid*" [BIENENSTOCK, 1995], e mais tarde os grupos poli-síncronos [IZHIKEVICH, 2006]. Estes dois últimos conceitos consideram que assembleias podem gerar um padrão de disparos fixo que se repete no tempo. Neste caso, os neurônios não necessitam disparar todos ao mesmo tempo, eles podem repetir sempre o mesmo padrão de disparos e com isso também caracterizar uma atividade de *cell assemblies*.

Nesse sentido, foi proposta a abordagem de NAC [RANHEL, 2012c]. NAC é uma abordagem de SNN que investiga como assembleias neurais representam coisas e estados do mundo, como as interações entre os conjuntos de neurônios levam ao processamento de informação e como isso resulta

em computação e comportamento.

NAC é fortemente baseada na sincronicidade entre as coalizões (ou assembléias) e algumas das topologias de NAC já estudadas dependem de geradores de ritmo internos, uma abordagem bio-inspirada [BUZSÁKI, 2010], [BUZSÁKI and DRAGUHN, 2004]. A princípio, em NAC foram tratados casos onde os neurônios de uma assembleia operam em modo 'digital': ou todos os membros estão disparando ou todos os membros estão em silêncio. Ao usar o modo operacional 'tudo-ou-nada', foi demonstrado que os conjuntos neurais podem executar funções booleanas, implementar memórias biestáveis e manter laços rítmicos biestáveis, sem exigir qualquer mecanismo de plasticidade. Finalmente, associando estas funções, NAC pode executar máquina de estados finitos [RANHEL, 2013].

No entanto, é importante notar que os estímulos externos podem causar um conjunto de *spikes* de neurônios que não se comportam como 'tudo-ou-nada' de uma montagem digital. Em vez disso, estímulos externos podem causar neurônios disparando proporcionalmente à intensidade do estímulo (código por populações), ou podem causar a variação na taxa de disparos de uma assembleia (código por taxa de disparos), ou ainda determinar o momento específico em que acontece um *spike* (código temporal), etc. (Ver seção 2.4).

Como foi dito antes, a NAC é uma abordagem que tenta explicar como computação é realizada por conjuntos de células neurais. Basicamente, quando corretamente excitados, neurônios podem disparar conjuntamente, formando uma coalizão, ou assembleia. Tal conjunto, por sua vez, pode produzir um estímulo forte o suficiente para disparar ou inibir outros conjuntos de neurônios. A combinação de excitação e inibição entre esses conjuntos de neurônios formam os conceitos básicos da abordagem NAC. É possível facilmente identificar suas semelhanças com a lógica digital. De fato, tem sido mostrado como funções lógicas podem ser executadas utilizando os conceitos NAC [RANHEL, 2012c]. Como exemplo, suponha que um conjunto de neurônios A pode sozinho provocar o disparo em outro conjunto C. É possível denotar como $C \leftarrow A$ (é lido: C é causado por A). Isso significa que os *spikes* gerados pelos neurônios de A chegarão em C depois de algum tempo, alguns milissegundos. Esta notação significa que existe um atraso, Δt, de propagação entre evento A e ocorrer C. Considere o peso sináptico, $w_{j,k}$, entre todos os membros da assembleias A a todos os membros da assembleia C calculado pela Equação 3.5:

$$w_{j,k} = \frac{\Theta}{N} \tag{3.5}$$

Onde w é o peso sináptico entre neurônio k (pré-sináptico) para o j (pós-sinápticos) e N é o

número de membros em A. Sendo Θ o peso sináptico necessário para fazer um membro de C disparar, logo, conjuntamente, os membros de A geram um EPSP que faz com que cada membro de C dispare.

Agora, considere que dois conjuntos A e B estão ligados a C por $w = \Theta/2$ cada (metade do peso sináptico necessário para disparar um neurônio em C). Isto significa que nem A nem B, por si só, pode provocar um disparo de C. No entanto, se, por exemplo, tanto a assembleia A quando B dispararem de forma que os *spikes* de ambas cheguem a C ao mesmo tempo, neste caso, a assembleia C disparará. É fácil notar que é necessário que tanto A quanto B contribuam com *spikes* para que o EPSP em C seja suficientemente elevado para provocar um disparo de seus neurônios. Em outras palavras, o evento A e B são o eventos necessários para ocorrer o evento C, formando à função lógica **AND**:

$$C \leftarrow A \cdot B \tag{3.6}$$

Por outro lado, suponha que A e B, ambos ligados a C com pesos sinápticos $w = \Theta$. Isso significa que A pode disparar C isoladamente, mas B também pode fazê-lo. Em outras palavras, A ou B podem provocar C, realizando a função lógica **OR**:

$$C \leftarrow A + B \tag{3.7}$$

Ainda, como foi demonstrado em [RANHEL, 2012c], assembléias reverberando na rede podem executar memórias biestáveis, bem como ritmos internos que podem funcionar como sinais de *clock*. Um ciclo de assembléias reverberando é implementado quando um conjunto D provoca uma assembleia E que dispara uma F que dispara de volta o conjunto D como explícito na equação 3.8. Neste caso, o somatório dos tempos Δt' de propagação do sinal entre as assembleias determina o período o ciclo.

$$D \rightarrow E \rightarrow F \rightarrow D \tag{3.8}$$

Este ciclo pode ser ativo (ON) ou inativo (OFF), possibilitando a este *loop* memorizar um bit de informação, semelhante a um circuito flip-flop eletrônico. Assembléias biestáveis podem ser desmontadas por assembleias inibitórias gerando as funções lógicas **NOT**, **NOR** e **NAND**. Estas funções lógicas são os principais componentes utilizados para a construção de computadores. Logo, NAC pode implementar máquinas de estado finitas [RANHEL, 2013], um bloco recorrente em sistemas que computam.

CAPÍTULO 4

CODIFICADOR NEURAL E GERADOR DE *Spikes* PARA SNNS

4.1 INTRODUÇÃO

NESTE capítulo será descrita a arquitetura e o funcionamento de um codificador neural e gerador de potenciais de ação artificiais implementado em FPGA para uso em SNNs. O circuito proposto tem como finalidade alimentar SNNs com dados, físicos e/ou simulados, através de um hardware dedicado de baixo custo. Esse sistema é capaz de realizar três tarefas independentes e correlatas, operando com 1024 neurônios por módulo. O sistema é capaz de, individualmente, gerar *spikes* com uma frequência controlada externamente pelo sistema utilizador. Ele também pode converter dados de entradas digitais em trens de *spikes* em tempo real. Além disso, pode organizar conjuntos de neurônios que disparam conjuntamente, a fim de gerar os três códigos neurais mais importantes descritos na literatura da neurociência.

A partir de um vetor de números independentes, foi criado um circuito que gera *spikes* de forma proporcional a esse valor. Quando a relação é direta, simplesmente é gerada uma taxa de disparo proporcional ao valor numérico estipulado para aquele neurônio. Esse tipo de geração de *spikes* equivale ao que na literatura de neurociência é descrito como codificação por taxa de disparos.

Quando a relação passa por alguma correlação entre neurônios, a codificação do sinal de entrada também leva em consideração a função exercida por cada neurônio em relação ao grupo de neurônios. Um desses esquemas é a codificação por populações, no qual grupos de neurônios representam de

forma esparsa alguma grandeza. A relação pode ser direta: quanto maior o estímulo maior a quantidade de neurônios ativos dentro da população. O outro esquema de codificação descrito na literatura da neurociência é a codificação temporal. Esse tipo de codificação é muito mais rico que o anterior, no que concerne à quantidade de representações que pode gerar (Ver seção 2.4).

A descrição do circuito será dividida em camadas de abstração: primeiramente será mostrado o circuito em alto nível de abstração, visando explicar o sistema para alguém que irá utilizá-lo para codificar sinais que alimentarão SNNs. Numa segunda camada de abstração será descrita a arquitetura utilizada para que o circuito realize o trabalho a que se propõe. E na terceira camada, serão descritos a implementação do circuito, os testes do protótipo funcional e os resultados obtidos, compondo a camada de mais baixo nível de abstração.

4.2 CARACTERÍSTICAS DO SISTEMA

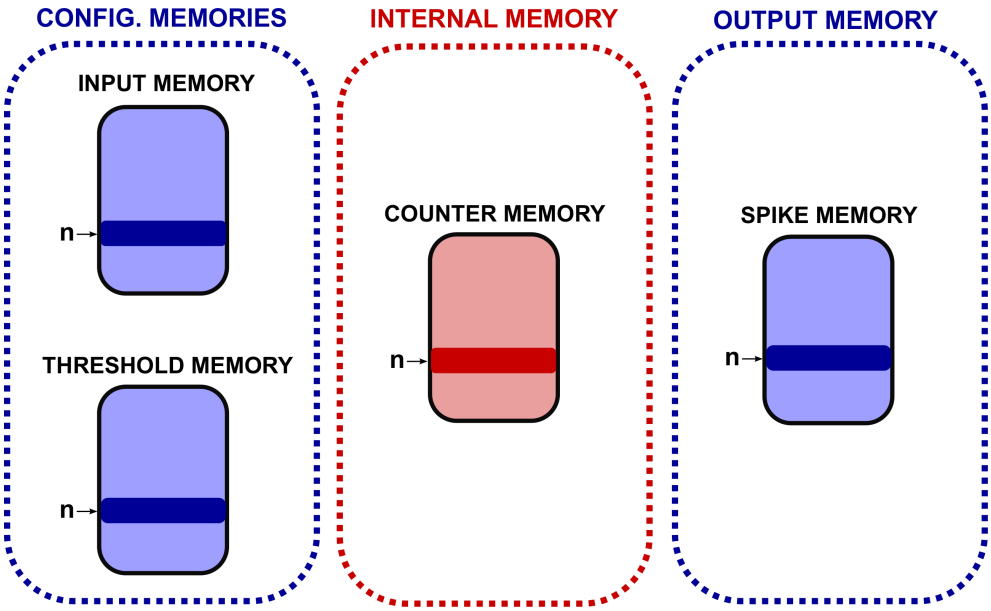
Traçando um paralelo, sistemas eletrônicos digitais necessitam que grandezas físicas sejam convertidas em sinais digitais (Ver seção 2.6). Esses sinais digitais são codificados em algum código que os tornem úteis aos sistemas computacionais a quem estão alimentando. Da mesma forma as SNNs de tempo real necessitam que grandezas físicas e sinais digitais sejam transformados em dados que possam ser computados pela rede. Esses sinais são os *spikes*, pulsos de duração de 1 ms, como descrito na seção 2.5. Esses pulsos indicam o momento em que aconteceu o potencial de ação em um neurônio. A taxa de disparos do neurônio e sua relação com outros neurônios são responsável por transmitir informação para rede.

Nesse trabalho é descrito o Gerador, Conversor e Codificador de Trens de *Spikes* (GCCst), que serve como interface entre dados numéricos e SNNs [OLIVEIRA NETO et al., 2015]. O GCCst faz a conversão de informação e/ou grandezas físicas para *spikes* de neurônios artificiais. Cada módulo GCCst simula até 1k (1024) neurônios trabalhando paralelamente, que podem ser configurados individualmente em relação a sua taxa de disparos e também agrupados para codificar uma informação (codificação por populações e temporal).

Para isso, o GCCst possui quatro blocos de memória: duas memórias de configuração, usadas para *setup*, uma memória interna, usada como contador, e uma memória de saída que disponibiliza o estado atual de cada neurônio (disparou (1) ou não disparou (0)). O esquema das memórias é mostrado na Figura 4.1. Nesta implementação, cada uma das memórias possui 1024 posições, uma dada posição ‘n’ em cada uma das memórias está relacionada ao mesmo neurônio. As memórias de configuração e de contagem possuem palavras de 8 bits, enquanto que na memória de saída as

palavras são de apenas 1 bit. As memórias de configuração são a *Input Memory* (IM) - Memória de Entrada - e a *Threshold Mémory* (TM) - Memória de Limiar. A memória interna é chamada de *Counter Memory* (CM) - Memória Contador - e a memória de saída é a *Spike Memory* (SM) - Memória de *Spikes*.

Figura 4.1: *Características do sistema GCCst destacando os blocos de memória*



Cada valor na IM é conectado a um contador que define a taxa de disparos de cada neurônio. Os contadores são implementados na CM. A última posição da IM possui um papel especial. Ela controla se todos os contadores são apagados simultaneamente ou individualmente. Qualquer valor diferente de zero nesta posição de memória faz com que todas as posições de CM sejam apagadas simultaneamente em função do número gravado nesta posição. Caso contrário, cada contador é apagado quando seu valor é igual ao valor no endereço correspondente na IM. O usuário controla quais são os valores salvos na IM assim como na TM. A TM controla a taxa mínima de disparo para que um neurônio funcione/dispare. A configuração dessas duas memórias (IM e TM) permite que o circuito configure trem de pulsos nas três codificações neurais previstas para o sistema. Como configurar as memórias será explicado detalhadamente nas seções seguintes.

4.2.1 Gerando Taxa de Disparos

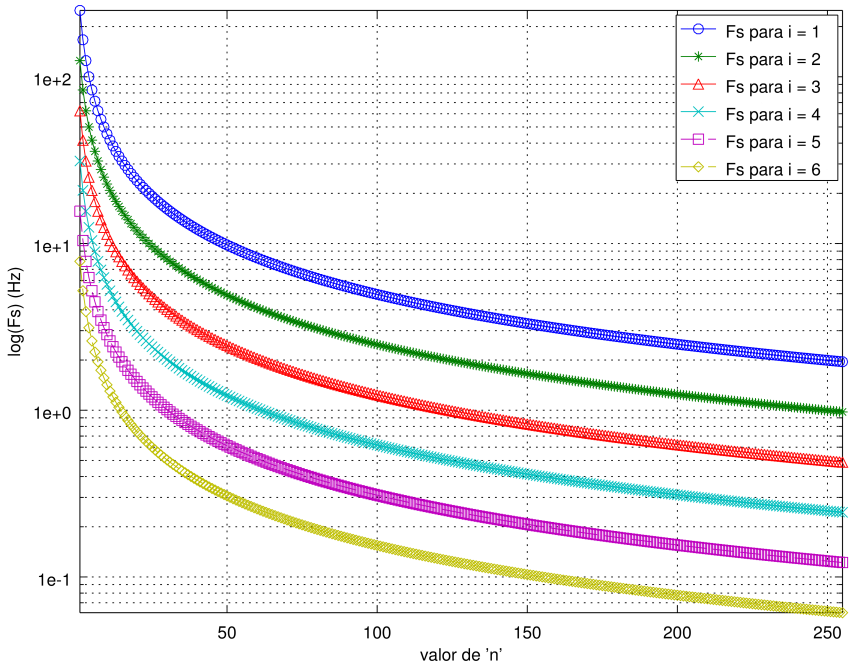
A configuração da taxa de disparos de cada neurônio é feita através da IM. Como dito anteriormente, cada posição de memória está atrelada à configuração de um neurônio. A última posição da IM tem uma função especial, e para trabalhar com *rate coding* deve ser deixada em branco, assim

como toda a memória TM. O valor guardado na IM define o período desejado para o disparo de cada neurônio, logo, a taxa de disparos é inversamente proporcional ao valor guardado na memória. Para um valor de n diferente de zero, a taxa de disparos é dada pela equação 4.1, onde F_S é dado em Hz (representando o número de *spikes* por segundo) e n é o número salvo na memória IM.

$$F_S = \frac{F_B}{n+1} \tag{4.1}$$

$$F_B = \frac{500}{2^i} \tag{4.2}$$

Figura 4.2: *Gráfico mostrando o logaritmo da frequência de disparo F_S como uma função do valor n guardado na posição de memória para cada um dos valores de i possíveis*



A frequência F_B é a frequência base dos contadores da CM. Ela tem por definição o valor de 500 Hz, mas pode ser configurada através do ultimo endereço da TM. O valor de F_B é definido pela equação 4.2, onde i é o número gravado no último endereço da TM. Os valores i podem variar de 0 a 5. Ou seja, a frequência de base máxima é $F_{B,MAX} = 500$ Hz e frequência de base mínima é $F_{B,MIN} = 15,65$ Hz. A modificação de F_B é interessante dependendo da faixa de frequência em que o usuário quer que os neurônios trabalhem. Na figura 4.2, é possível ver os valores de F_S em função do valor n para cada um dos valores de i. Para facilitar a visualização foi utilizado um gráfico

semilog, onde o eixo das ordenadas está em escala logarítmica.

4.2.2 USANDO CODIFICAÇÃO POR POPULAÇÕES

O uso da codificação por populações está atrelado à configuração da TM. Cada posição desta memória indica a taxa de disparos mínima para que um neurônio pertença a uma população. Em outras palavras, para uma mesma posição de memória, se um valor contido em IM é maior que o valor na TM, o neurônio não disparará. Caso contrário, o neurônio disparará com a taxa de disparos configurada na IM. Com isso é possível configurar assembleias com número variável de neurônios e em posições de memória também configuráveis. O uso de um limiar mínimo de disparo minimiza ainda que sinais espúrios interfiram na dinâmica da população.

4.2.3 USANDO CODIFICAÇÃO TEMPORAL

Como dito anteriormente, a ultima posição da IM possui uma característica particular. Ela é utilizada para gerar um sinal de sincronização para todos os demais neurônios, possibilitando assim a codificação temporal. Quando um valor diferente de zero é gravado nesta posição de memória indica que o circuito todo trabalhará tendo como variável adicional o sinal de sincronismo.

Na frequência em que foi configurada a ultima posição da IM é gerado um sinal de *reset* para todos os demais neurônios. Quando o sinal de *reset* ocorre, todos as posições da CM são apagadas. Esta taxa de *reset* segue a mesma lei da taxa de disparo para os neurônios, a equação 4.1. Neste caso, n é o valor gravado na ultima posição da IM, e F_S o valor da frequência de sincronismo.

Apenas dispararão os neurônios cuja taxa de disparos é maior que o sinal de sincronismo. Para os demais neurônios, a posição correspondente em SM sempre vai ser vista em nível lógico zero. Além disso, será observado um padrão de disparos em torno do sinal de sincronismo. Existe um grande número de possibilidades de configuração, mas elas são dependentes das escolhas feitas pelo usuário.

4.2.4 COMO TRANSFORMAR UMA INFORMAÇÃO EM *Spikes*

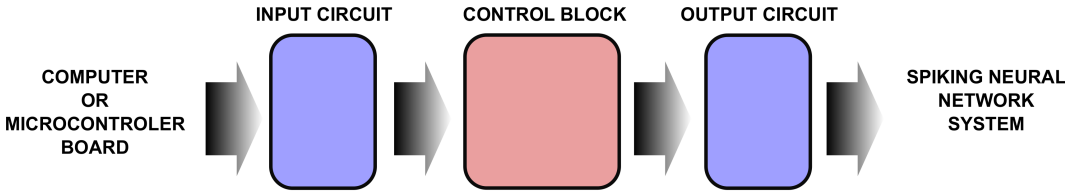
Os usuários devem considerar normalizar os dados, cujo domínio seria aplicável às equações 4.1 e 4.2. O sistema não codifica diretamente valores analógicos, então algum tipo de circuito ADC deve ser usado - uma boa opção são microcontroladores de baixo custo, como discutido nas Seções 2.5 e 2.6 do capítulo 2. O modo de codificação e o número de neurônios usados para representar alguma informação dependem das escolhas do usuário projetista e também do tipo de informação a ser codificada. Este estudo precede a incorporação dos dados à rede. Depois de feita estas decisões,

elas tornam-se regras particulares para o processamento desses sinais no sistema do usuário. Uma boa alternativa para escolha do tipo de codificação é procurar soluções biomiméticas. Isto significa, achar sistemas similares em organismos vivos e como a neurociência acredita que esses sistemas codificam informação. É uma interessante estratégia por se basear em sistemas que, obviamente, funcionam, além de trazer mais plausibilidade biológica para os sistemas artificiais, algo buscado, em maior ou menor grau, em toda as áreas da Inteligência Artificial.

4.3 ARQUITETURA DO SISTEMA

Como mostrado nas seções anteriores todo o controle do circuito GCCst é feito através da escrita e leitura de blocos de memória. Para facilitar essa escrita e leitura das memórias, a arquitetura adotada para o sistema foi pensada de forma modular. Como mostrado na Figura 4.3, o sistema é composto por três blocos que funcionam independentemente: o *Input Circuit* (InC) - Circuito de Entrada - , *Control Block* (CB) - Bloco de Controle - e o *Output Circuit* (OutC) - Circuito de Saída.

Figura 4.3: *Arquitetura modular do sistema, mostrando os três módulos que compõe o sistema: o Input Circuit (InC) - Circuito de Entrada - , Control Block (CB) - Bloco de Controle - e o Output Circuit (OutC) - Circuito de Saída.*



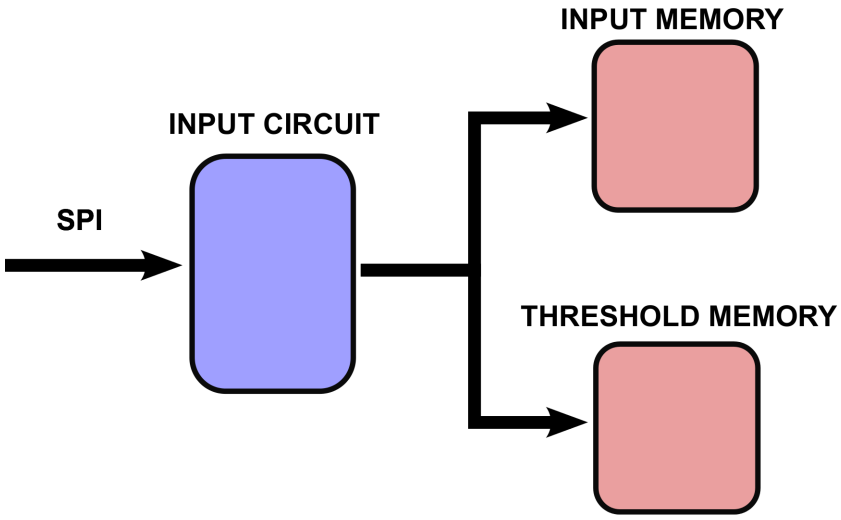
A arquitetura dividida em módulos foi escolhida para dissociar as interfaces com o mundo externo do núcleo de processamento do circuito, o CB. Isso facilita a implementação da computação necessária no CB, tornando-o independente do protocolo de comunicação utilizado nos circuitos de entrada e saída. Logo, tanto o InC quando o OutC podem ser modificados visando melhorias e adequações do sistema. Desde que eles mantenham a comunicação correta com o CB.

4.3.1 CIRCUITO DE ENTRADA

O InC é a porta de entrada dos sinais externos para o módulo GCCst. Ele escreve nos bancos de memória de configuração, sendo então responsável pela configuração do módulo e por determinar a taxa de disparos dos neurônios. Estas configurações podem ser mudadas em tempo de execução, promovendo melhor uso do sistema.

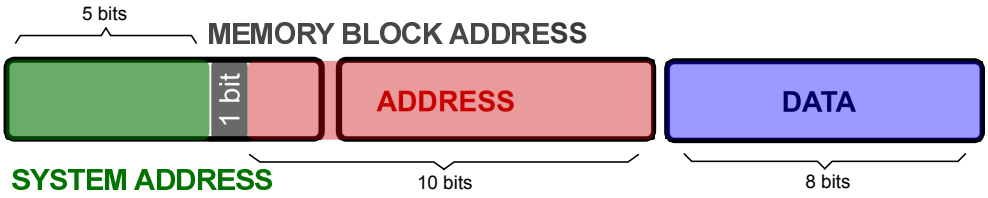
Será descrito nesta seção o circuito implementado no protótipo funcional. No circuito foi implementado um receptor de dados que utiliza o padrão SPI [LEENS, 2009]. Contudo, o sistema é modular, então é possível facilmente modificar o padrão de comunicação usado, desde que mantenha usabilidade do módulo GCCst. O InC recebe os dados para serem gravados na memória através de um pacote de dados que possui 3 bytes. Um esquema em blocos do InC pode ser visto na Figura 4.4.

Figura 4.4: *Esquema em blocos do InC.*



O pacote de dados de entrada é mostrado da Figura 4.5. O dois bytes mais significativos são utilizados para endereçamento da posição específica de uma das memórias, enquanto o byte menos significativo carrega o dado a ser gravado na memória. Os 10 bits menos significativos, dos 16 bits de endereçamento, são usados como endereço interno de cada memória. O bit 11 indica qual das memórias é acessada. Os outros 5 bits são reservados para endereço do módulo GCCst. Isto possibilita gerenciar até 32 módulos GCCst ligados a um mesmo barramento.

Figura 4.5: *Estrutura do pacote de dados de entrada.*



Uma máquina de estados é responsável por receber o pacote de dados e guardar em um registrador temporário. Após o recebimento completo do pacote, é verificado se o endereçamento do módulo está correto, em caso negativo, o conteúdo do registrador é descartado e a máquina de estado volta a esperar um novo pacote de dados. Se o endereçamento do módulo foi feito corretamente, o byte de

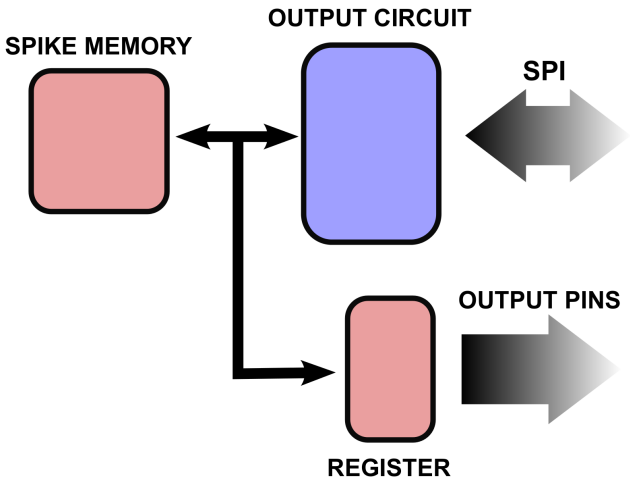
dados é gravado na memória e endereço correspondentes.

4.3.2 Circuito de Saída

Como na seção anterior, nessa seção será mostrado o OutC implementado no protótipo funcional, cujo esquema é mostrado na Figura 4.6. Contudo, qualquer variação no protocolo de comunicação, ou no circuito de acesso à SM, pode ser empregado, desde que garanta a utilidade dos dados.

O OutC é responsável por ler a memória SM e transmitir seu conteúdo para outros dispositivos digitais, provavelmente SNNs. A transmissão pode ser tanto serial como paralela. A SM mantém o estado atual (disparou ou não) de cada neurônio, sendo utilizado um bit para cada neurônio. O acesso SM se dá através de 128 palavras de 8 bits, facilitando a transmissão dos dados serialmente.

Figura 4.6: *Esquema em blocos do OutC.*



Um transmissor SPI mestre foi utilizado para transmitir os dados de SM. O início da transmissão é controlado pelo sinal de *enable*. O controle dele pode ser interno ou controlado por um circuito externo. No protótipo, este comando foi atribuído a um botão externo.

O circuito SPI mestre possui internamente uma máquina de estados que divide o *clock* do sistema por uma constante pré programada, garantindo um *baud rate* desejado. Ainda é possível programar o dado para estar estável na descida ou na subida do *clock*. Outra configuração possível é como o dado é enviado: se com o bit mais significativo primeiro, *big-endian*, ou com o menos significativo sendo enviado primeiro, *little-endian*. Com estes pontos em mente, o usuário pode configurar o módulo dependendo do sistema que irá receber os dados. Ainda, os últimos 8 bits lidos da SM são guardados

em um registrador interno e disponibilizados parlamente em pinos de saída, ajudando os testes e caracterização do circuito.

4.3.3 BLOCO DE CONTROLE

O CB é o "núcleo" de todo sistema. Ele é um circuito digital baseado em eventos do *clock* com uma resolução temporal de 1 ms. O CB executa operações aritméticas com o conteúdo de todas as posições das memórias de configuração e contadora. As decisões tomadas sobre o disparo dos neurônios é salva na SM e o processo atualização dos contadores da CM é então realizado.

O diagrama em blocos do CB é mostrado na Figura 4.7, onde é possível ver que o CB é composto de um núcleo central de controle e três blocos de auxiliares. Os blocos auxiliares são o *Clock Control* (ClkC), *Comparator Circuit* (CC) e o *Update Circuit* (UC).

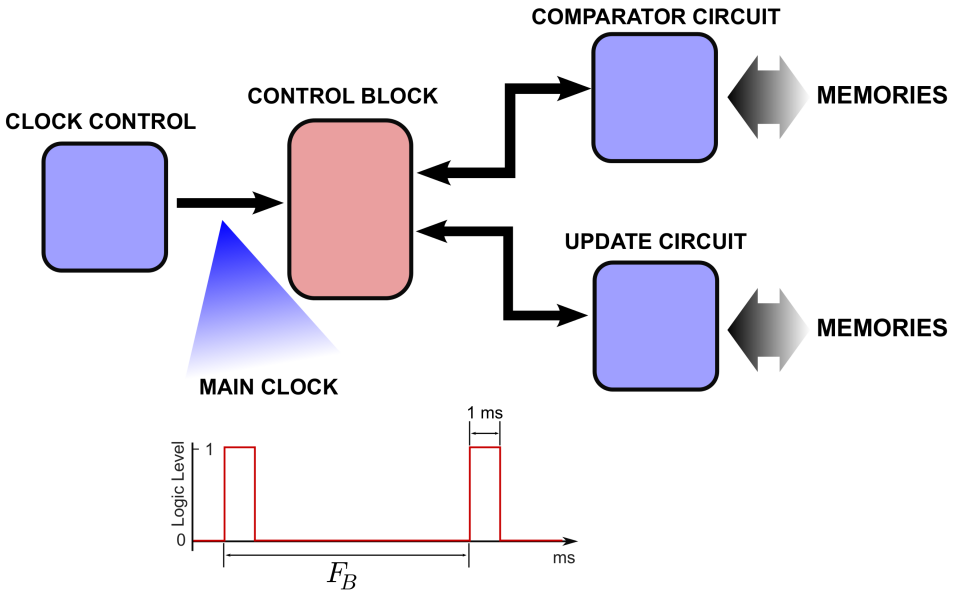
O ClkC é responsável por gerar o *clock* para todos os módulos do circuito. Além desse sinal, ele é responsável por gerar um sinal importante para o funcionamento do CB, o *Main Clock* (MCLK). O MCLK é um sinal configurável utilizado pelo CB para ativar os demais circuitos auxiliares. Este sinal é composto por pulsos de 1 ms que acontecem com a frequência configurada em F_B, como mostrado na Figura 4.7, configurada pela ultima posição da memória TM. Como dito anteriormente, a frequência F_B vem configurada para trabalhar em 500 Hz, podendo ser modificada através da escrita na ultima posição da TM, onde F_B definida pela equação 4.2, onde i pode adquirir valores entre 0 e 5.

No núcleo central é encontrado uma máquina de estados finitos de 5 estados, denominados S_I ao S_V, que controlam o CB. Na Figura 4.8 é possível ver o digrama de estados da máquina de estados interna do CB. O estado S_I é o estado inicial; a máquina permanece neste estado até receber uma borda de subida do sinal MCKL, então a máquina de estados muda para o estado S_{II}.

Um comando para comparar as memórias ocorre em S_{II}. Este sinal inicializa um circuito auxiliar, o CC, responsável por comparar as memórias IM, TM e CM, e atualizar os valores em SM dependendo dessas comparações. Para isso, o módulo CC conta com uma máquina de estado interna, inicializada pelo sinal enviado pela máquina de estados principal. Uma vez iniciada a máquina de estados do CC, o seguinte processo é repetido para cada endereço de memória: os conteúdos da IM, CM e TM são lidos e armazenados em registradores temporários. Esses valores são comparados combinacionalmente e é gerado um bit que será escrito na SM. Após a atualização da SM, o endereço de leitura é incrementado e o processo é repetido para próxima posição de memória.

A lógica utilizada para saber que dado vai ser escrito na SM verifica se o valor contido na IM

Figura 4.7: *Esquema em blocos do CB e seus blocos auxiliares, destacando a forma de onda do sinal MCLK.*



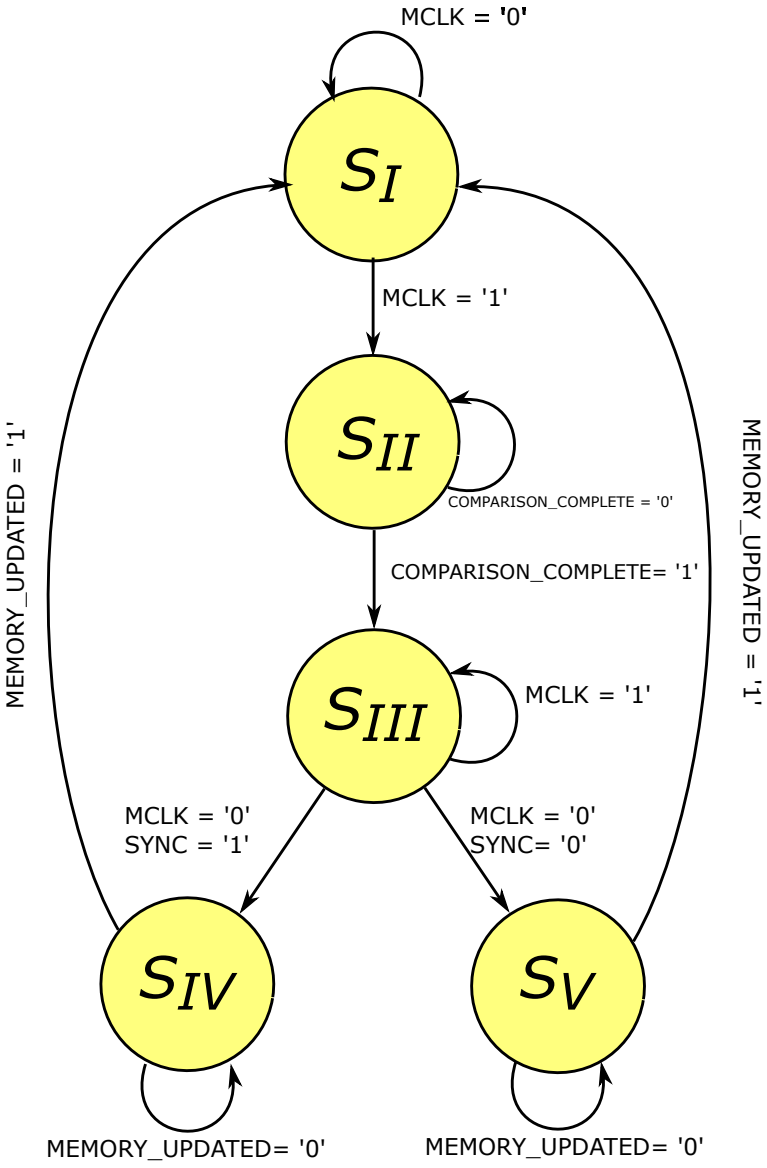
é diferente de zero e o compara com o valor contido na CM. Caso valor contido IM seja diferente de zero e menor ou igual ao da CM, um sinal de resposta vai a nível lógico '1'. Paralelamente, o valor TM é comparado com o valor da IM: se o valor de TM for igual a zero ou se for maior que o guardado na IM, um sinal de resposta vai para nível lógico '1'. Os dois sinais de resposta passam por uma porta lógica AND e o resultado é guardado na SM.

Após o processo ser repetido para todas as posições de memória, o CC força um sinal chamado `COMPARISON_COMPLETE` para o nível lógico '1', então a máquina de estados do CB vai para o estado S_{III}. Enquanto a do CC volta ao estado de espera, até que a máquina de estados principal volte ao estado S_{II}.

No estado S_{III}, o módulo espera um novo evento do MCLK, agora a borda de descida, que ocorre 1 ms após a borda de subida. Quando ela corre a máquina de estados muda para o estado seguinte, podendo ser S_{IV}, ou S_V. Ambos os estados habilitam o circuito auxiliar UC, que atualiza CM e a SM.

A escolha entre os estados S_{IV} ou S_V é feita por outra informação envida pelo CC, o sinal `SYNC`. Quando a ultima posição da memória é avaliada no CC, caso a IM seja diferente de zero e o valor da CM é maior que o guardado na IM, o CC envia um sinal para o CB que é armazenado em um flip-flop (`SYNC` $= 1$ para verdadeiro e `SYNC` $= 0$ para falso). Em caso verdadeiro significa que o

Figura 4.8: *Máquina de estados principal do sistema, interna ao CB.*



sistema está configurado para trabalhar em codificação temporal e que esse sinal de sincronismo foi alcançado. Nesse caso, do estado S_{III} a CB passa para o estado S_{IV}, caso contrário a máquina de estados principal segue para o estado S_V.

O circuito auxiliar UC também é munido de uma máquina de estados interna. Este circuito é responsável por atualizar tanto SM quando a CM. O funcionamento da máquina de estados interna independe do estado máquina de estados principal (S_{IV} ou S_V), apenas o meio de atuar nas memórias é que muda. Para cada endereço de memória o UC limpa a SM, gravando '0' na posição de memória, e atualiza o 'contador' CM.

No entanto, a atualização de CM é dependente do estado atual do CB. O valor atual da memória CM é lido, assim como o valor contido na IM. Paralelamente, o valor lido da CM é incrementado e guardado em um registrador temporário, enquanto os valores lidos da IM e CM são comparados. Se o valor em CM for maior ou igual ao valor de IM ou se o CB esteja no estado S_{IV}, é escrito '0' na posição de CM. Já se CM for menor que IM e a máquina de estados principal se encontre no estado S_V, o valor incrementado de CM é guardado em CM. Esse procedimento é repetido para todos os endereços de memória. Após a atualização, o módulo UC gera um sinal `MEMORY_UPDATED` e a máquina de estado volta para o estado S_I. No próximo evento do MCLK recomeça todo processo do CB.

Em resumo, este "núcleo" é responsável pela geração individual de *spikes* e pela atualização da SM quando ocorre o *spike*. É também responsável pela da atualização dos temporizadores individuais de cada neurônio dispostos na CM. E ainda responsável pela sincronização dos eventos que geram os códigos dependentes do tempo e os limiares de disparo. Sendo as configurações passíveis de modificação em tempo de execução.

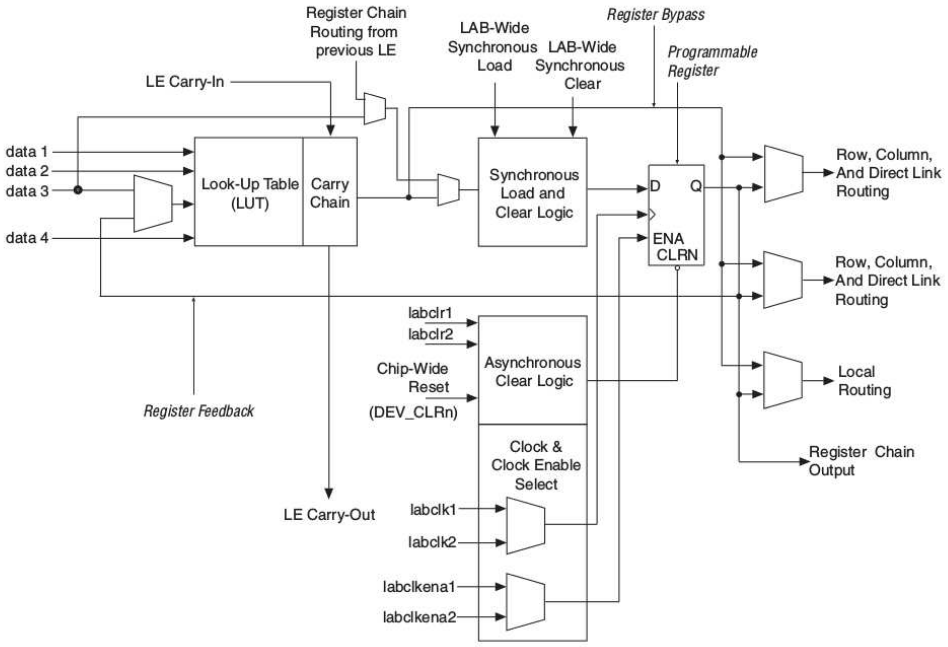
4.4 IMPLEMENTAÇÃO E RESULTADOS

Para implementação dos circuitos descritos nas seções anteriores foi utilizado uma FPGA (*Field Programmable Gate Array* ou Arranjo de Portas Programável em Campo) Altera da família *Cyclone* IV. Para descrição do circuito foi utilizada a linguagem de descrição de hardware Verilog [IEEE, 2006].

FPGAs são circuitos integrados programáveis que usam conjuntos de blocos lógicos, ou elementos lógicos (LE - *Logic Elements*), como unidades básicas de construção do dispositivo lógico programável. Na Figura 4.9 é mostrado o esquema do LE retirado do manual da família *Cyclone* IV da Altera [ALTERA, 2014]. Cada LE pode ser grosseiramente dividido em duas partes, uma parte

que implementa funções combinacionais e um registrador dedicado ao LE. Esse registrador dedicado é utilizado para construção de lógica sequencial e memórias. Os LEs são ligados em conjuntos chamados de Blocos de Matriz Lógica (LAB - *Logic Array Block*) [ALTERA, 2014]. Os LABs, por sua vez, se conectam entre si através de barramentos de dados, como mostrado na Figura 4.10.

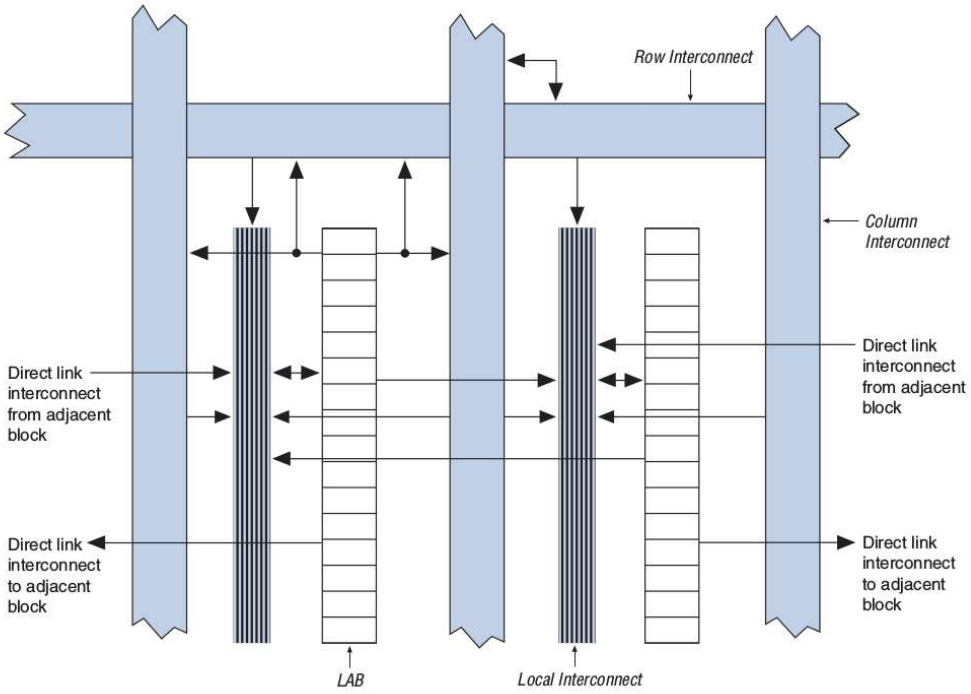
Figura 4.9: *Esquema de um LE da FPGA da família Cyclone IV da Altera (Retirado de [ALTERA, 2014]).*



Uma FPGA pode conter desde centenas a milhões de LE. A arquitetura, tanto dos LEs quanto dos LABs, difere muito entre diferentes marcas de FPGA, ou mesmo entre as famílias de dispositivos da mesma marca. Fica evidente a necessidade da assistência de um computador para realizar a configuração de todos os LEs. Para isso foram desenvolvidos ferramentas *Electronic Design Automation* (EDA). As EDAs fazem a configuração do dispositivo a partir das diretivas em alto nível de abstração passadas pelo projetista. O programa EDA da Altera, onde o usuário descreve o hardware de seu projeto, é o Quartus II.

O usuário pode configurar esses *chips* para implementar em hardware funcionalidades complexas e personalizadas. Para isso, ele deverá descrever o hardware desejado utilizando uma linguagem de descrição de hardware (HDL - *Hardware Description Language*). Essa descrição do circuito é introduzida no Quartus II. A EDA gera as configurações necessárias para que a FPGA tenha comportamento análogo ao circuito descrito, o que é chamado de síntese do hardware. As duas linguagens mais conhecidas para descrição de hardware são Verilog e VHDL. Como dito anteriormente, nesse projeto foi utilizada a linguagem Verilog.

Figura 4.10: *Esquema da disposição dos LABs interconectados por barramentos em uma FPGA da família Cyclone IV da Altera (Retirado de [ALTERA, 2014]).*



4.4.1 HARDWARE UTILIZADO

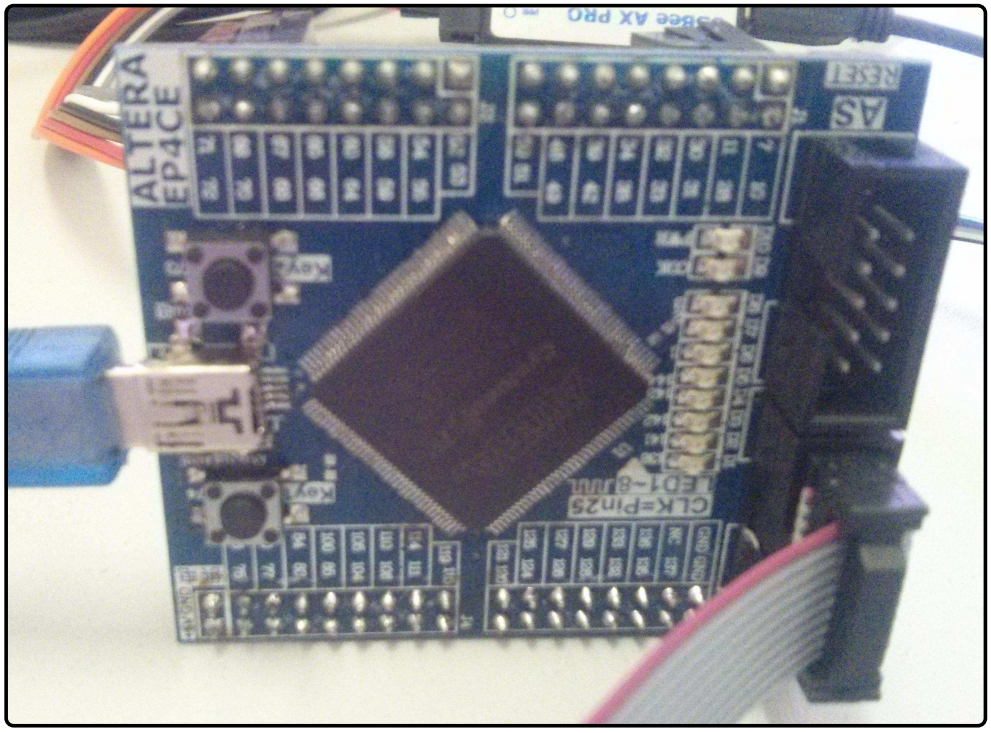
O circuito proposto foi sintetizado em uma FPGA *Cyclone* IV EP4CE6E22C8N da Altera, usando o software Quartus II 14.0 *subscription edition*. Para o protótipo, foi utilizada a FPGA contida na placa de desenvolvimento de baixo custo mostrada na Figura 4.11. A placa dispõe, além da FPGA, apenas do circuito de alimentação, circuito de gravação da FPGA, um cristal oscilador piezoelétrico de 25 MHz, 10 leds ligados a pinos da FPGA e os pinos de entrada e saída (IO) da placa disponíveis em conectores macho, como pode ser visto na foto.

A *Cyclone* IV EP4CE6E22C8N possui: $6,272$ LE , $276,480$ bits de memória RAM, 2 PLLs, 30 multiplicadores de 9 bits e 91 pinos de IO disponíveis para livre configuração. Neste projeto foi utilizado um PLL e parte da memória RAM interna para geração do *clock* do sistema e para implementação dos blocos de memória, respectivamente.

Gerando o *Clock* do Sistema

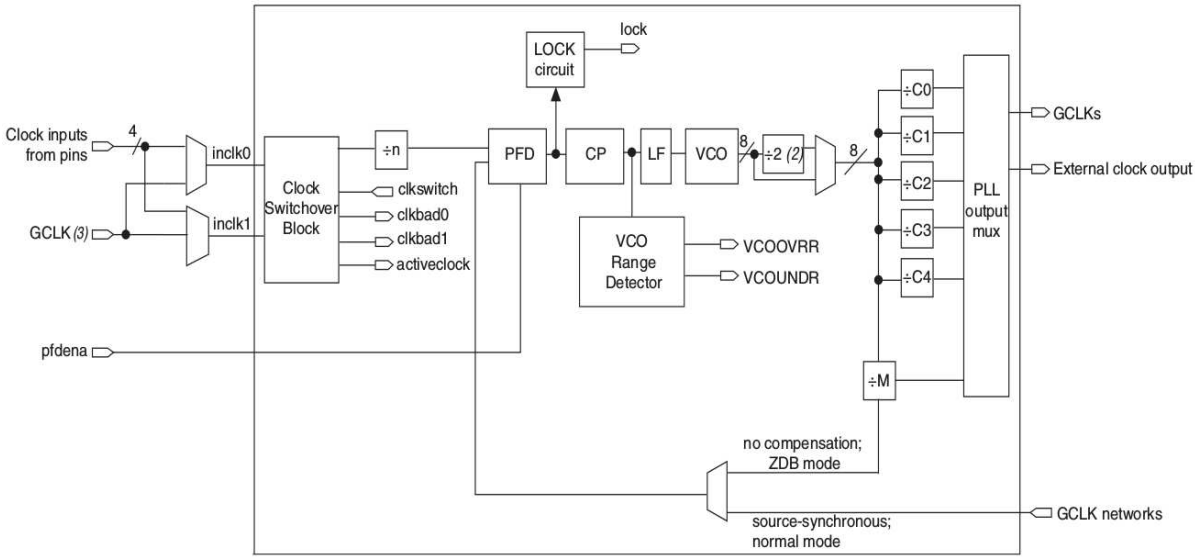
Como dito anteriormente, a placa de desenvolvimento utilizada possui um cristal oscilador de 25 MHz. No sistema foi utilizado duas fontes de *clock* disponibilizadas pelo ClkC, o *clock* que alimenta todos os módulos do sistema, e o MCLK que comanda os eventos do CB. Para garantir

Figura 4.11: *Foto da placa de desenvolvimento utilizada para implementação do protótipo funcional.*



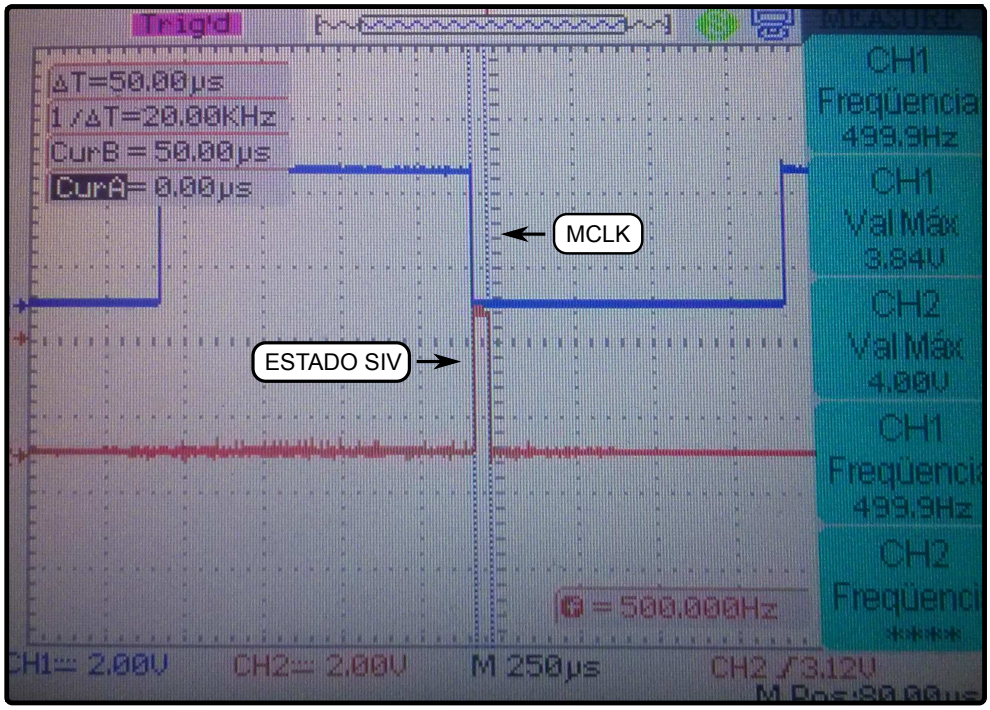
que os módulos internos processassem rápido o suficiente, de maneira a ser possível desconsiderar o desvido temporal entre neurônios, foi utilizado um módulo PLL para elevar o *clock* de 25 MHz para 100 MHz, utilizando este último como *clock* de alimentação dos módulos.

Figura 4.12: *Esquemático do circuito PLL encontrado em FPGAs da família Cyclone IV E da Altera (Retirado de [ALTERA, 2014]).*



O valor inicialmente elevado a 100 MHz pode ser dividido pelo PLL, gerando um segundo *clock* sincronizado com o primeiro. O *clock* de 100 MHz foi então dividido para uma saída mais próxima ao MCKL, que varia entre 500 Hz e $15,65$ Hz, como mostrado na seção 4.2.1. Como o PLL é um circuito que não trabalha em frequências tão baixas, foi gerada uma saída secundária com 8 kHz e utilizado um circuito interno ao ClkC para gerar o sinal configurável MCLK que é utilizado no sistema.

Figura 4.13: *Foto tirada da tela do osciloscópio durante testes de implementação. Acima, o canal CH1 em azul, é possível ver o MCLK. Enquanto que abaixo, o canal CH2, é possível ver um sinal indicativo que a máquina de estados principal se encontra no estado S_{IV}.*



Na Figura 4.13 é mostrada uma foto tirada da tela de um osciloscópio durantes os testes de implementação. Na parte superior da tela, em azul, é mostrado o sinal MCLK, com uma frequência aproximada de 500 Hz ($499,9$ Hz). Essa frequência foi calculada pelo osciloscópio em tempo de execução para o período mostrado na tela. Na parte de baixo da tela, em vermelho, é mostrado um sinal indicativo de que a máquina de estados principal encontra-se no estado S_{IV}. Esse sinal indica o tempo que o circuito leva para atualizar todas as posições de memória. Como mostrado na medição dos cursores, esse tempo é menor que 50 μs.

Acesso aos Blocos de Memória

Os quatro blocos de memória utilizados no sistema foram implementados utilizando bits de memória RAM disponíveis na EP4CE622C8N. A IM, TM e CM são idênticas, blocos de memórias com 1024 palavras de 8 bits. O acesso a elas é feito por interfaces de leitura e escrita dissociadas entre si. Esta escolha de configuração simplifica o controle da modularidade do sistema, pois, como o InC apenas se comunica com o CB através da escrita nas memórias de configuração e o barramento de escrita é dissociado do barramento de leitura, é garantido que o InC possa trabalhar independentemente do estado atual do CB. Os dados mais recentes gravados nas memórias de configuração são utilizados pelo CB, tirando o caso particular de quando uma posição específica de memória está sendo escrita e lida na mesma hora. Neste caso, o CB utilizará o valor anteriormente gravado na dada posição de memória e utilizará o novo valor apenas no próxima leitura da memória.

O mesmo ocorre com a SM. Ela é um bloco de memória com 1024 palavras de 1 bit. Neste caso, a SM foi configurada para trabalhar com dois barramentos de leitura e dois barramentos de escrita. O CB se comunica com a SM por um barramento de 1 bit de dado, com endereçamento bit a bit. Enquanto que o OutC tem acesso a SM por um barramento de 8 bits de dados através de 128 endereços.

O Protótipo em Números

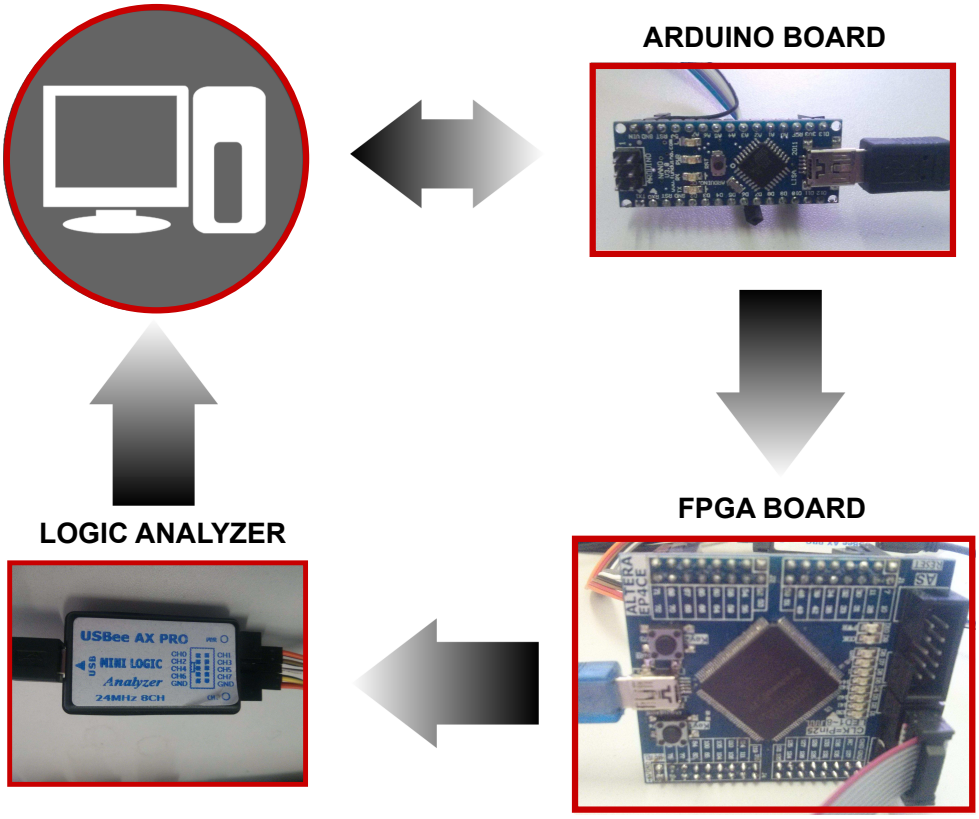
O projeto consumiu apenas 247 blocos lógicos, além disso, 25.600 bits de RAM foram utilizados, totalizando, respectivamente, 4% e 9% dos recursos da FPGA. A escolha de usar a memória RAM interna da FPGA para implementar as memórias e contadores garantiu um baixo consumo de blocos lógicos, sendo então possível sintetizar vários módulos GCCst idênticos em uma única FPGA de baixo custo, como a *Cyclone* IV utilizada.

4.4.2 ARRANJO EXPERIMENTAL

Para testes do protótipo funcional implementado foi utilizado o arranjo mostrado na Figura 4.14. Uma placa de desenvolvimento Arduino foi utilizada como uma ponte entre o computador e placa de desenvolvimento FPGA. As configurações da placa Arduino foram feitas através do protocolo USB CDC. Enquanto o Arduino alimentava a FPGA com dados através do protocolo SPI.

Os dados de saída gerados pelo GCCst foram capturados por um analisador lógico de baixo custo que possui 8 canais, podendo capturar dados com uma taxa de até 24 MHz. Os dados capturados foram analisados usando o software USBee Suite, cujos gráficos de fluxo de sinais são mostrados nas

Figura 4.14: *Esquema do Arranjo Experimental.*

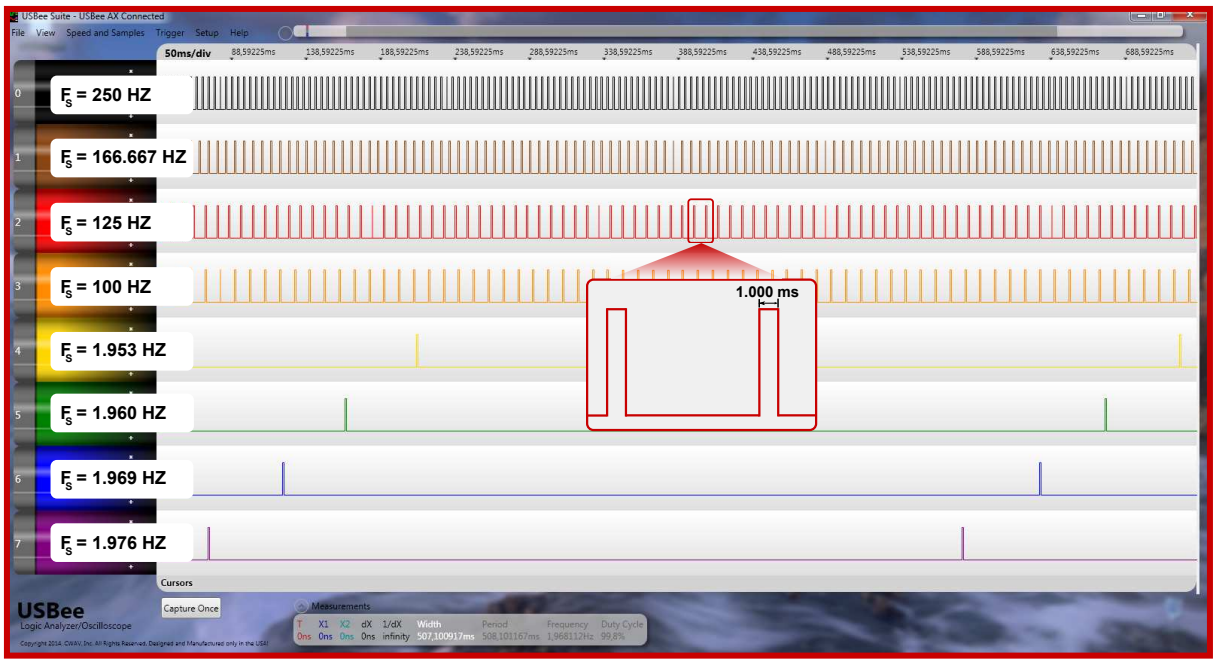


secções seguintes.

4.4.3 TESTE COM TAXA DE DISPARO

A Figura 4.15 mostra os *spikes* de oito neurônios trabalhando em codificação por taxa de disparos. Nos canais 0 ao 3 é mostrado as maiores frequências possíveis com o módulo, para $F_B = 500$ Hz. As posições da memória IM foram gravadas com os valores $n_{(0)} = 1$, $n_{(1)} = 2$, $n_{(2)} = 3$ e $n_{(3)} = 4$, respectivamente. Os canais 4 ao 7 mostram as menores frequências alcançadas para o módulo trabalhando com $F_B = 500$ Hz. Cuja as posições em IM foram gravadas com os valores $n_{(4)} = 255$, $n_{(5)} = 254$, $n_{(6)} = 253$ e $n_{(7)} = 252$, respectivamente.

Figura 4.15: *Tela do Analisador Lógico mostrando 8 neurônios disparando com taxas de disparos diferentes (Figura adaptada de [OLIVEIRA NETO et al., 2015]).*



Os resultados são mostrados na Tabela 4.1. O analisador lógico foi configurado para trabalhar a 12 Msps e o gráfico com o fluxo de sinais é mostrado na Figura 4.15, formado por 10 milhões de amostras. F_M é a frequência lida no software USBee Suite. Embora o software possua 6 casas de precisão, na Tabela 4.1 os números foram truncados na terceira casa decimal. O erro (ERROR) para alguns dados, (–*), foi aproximadamente zero e o erro devido ao analisador lógico não pode ser verificado por se tratar de um aparelho de baixo custo. Além disso, o *spike* é constante para todos os sinais e com uma duração de 1 ms, como desejado. No software USBee Suite esse valor foi encontrado para a terceira casa decimal. Para o maior valor de ERROR encontrado (para $n = 253$), o

desvio entre o sinal teórico e o medido foi na ordem de 10^{-4} s, para um sinal com período na ordem de 10^{-1} s. Já para os maiores valores de F_s, o desvio entre o sinal teórico e o mensurado ficou na ordem 10^{-9} s, para sinais cujo período são da ordem de 10^{-3} s.

Tabela 4.1: *Resultado dos testes para o sistema trabalhando por taxa de disparos. Onde F_s é a frequência calculada pela 4.1, e F_M a frequência medida (Tabela adaptada de [OLIVEIRA NETO et al., 2015]).*

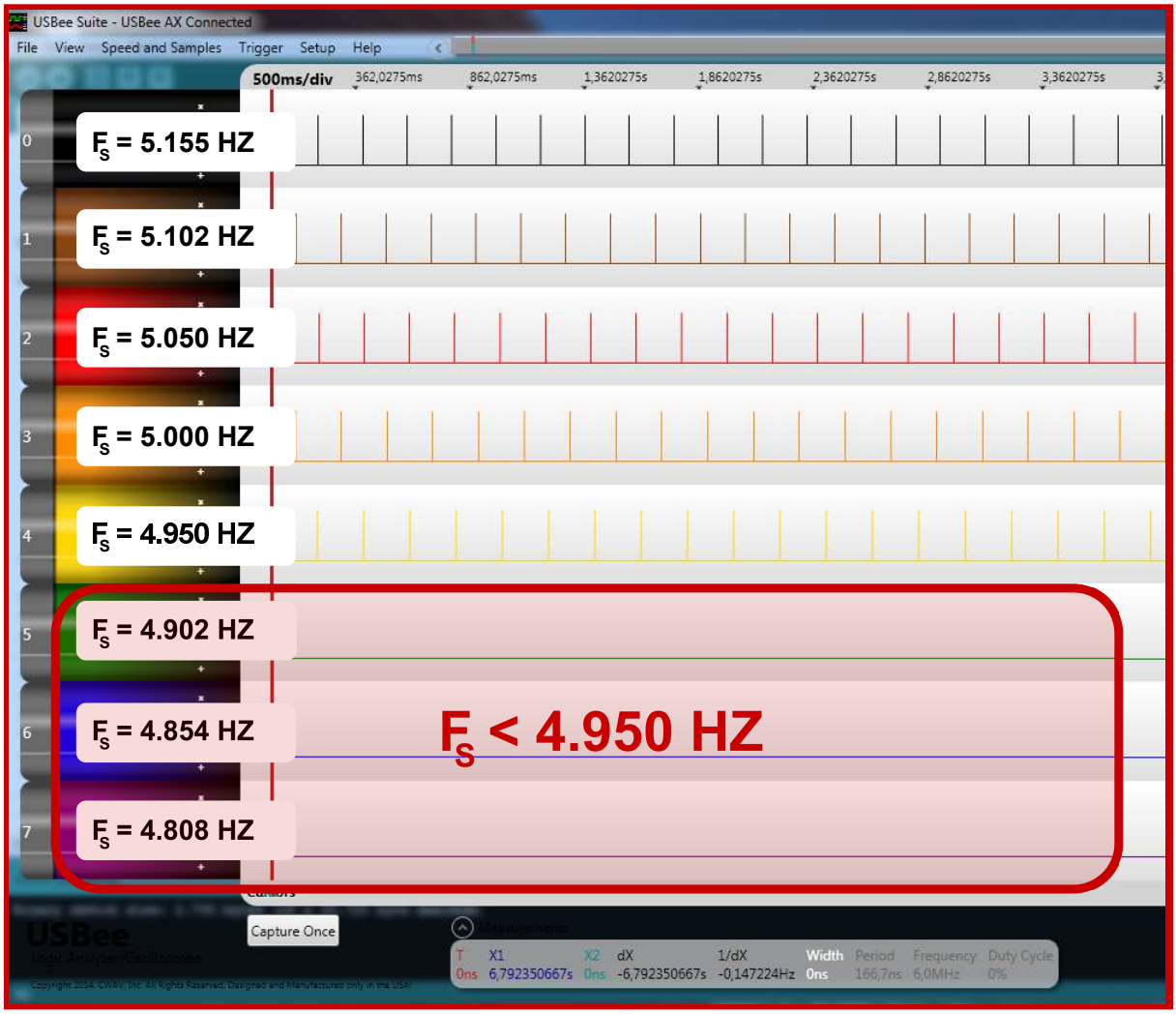
Channel	n	F_s (Hz)	F_M (Hz)	ERROR (%)
0	1	250.000	249.948	0.021
1	2	166.667	166.634	0.020
2	3	125.000	124.974	0.021
3	4	100.000	99.980	0.020
4	255	1.953	1.953	–*
5	254	1.960	1.960	–*
6	253	1.969	1.968	0.051
7	252	1.976	1.976	–*

–* Aproximadamente zero

4.4.4 Teste com Limiares de Disparo para Codificação por Populações

Para a Figura 4.16, foi configurada a TM com o valor $n = 100$, para todas as posições de memória e a IM com um n igual aos 8 bits menos significativos do endereço de cada posição de memória. A Figura 4.16 mostra os neurônios 96 ao 103, formando uma pequena população visível de 8 neurônios. Consequentemente, apenas os neurônios 96 ao 100 podem disparar nessa população (canais 0 ao 4). A medida que os neurônios tem suas taxas de disparos modificadas devido a um estímulo externo variável, os neurônios pertencentes a população podem passar a disparar ou deixar de disparar. O controle de TM, como dito anteriormente, é feito em cada neurônio e pode ser modificado em tempo de execução. Isso torna possível configurar populações com tamanhos variáveis de neurônios e com limiares apropriados para cada aplicação.

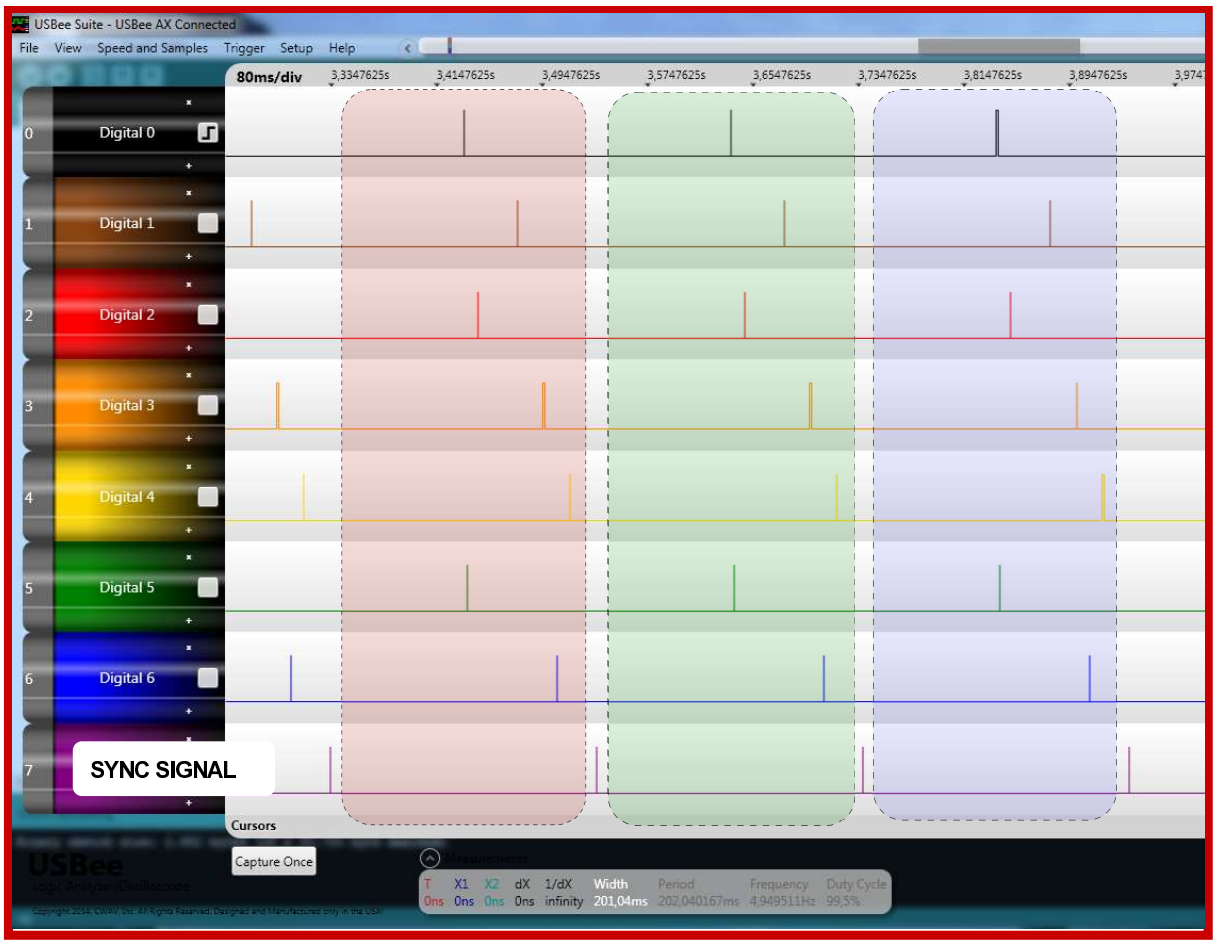
Figura 4.16: *Tela do Analisador Lógico mostrando 8 neurônios cujo limiar mínimo de disparo foi configurado para $F_s = 4.95$ Hz (Figura adaptada de [OLIVEIRA NETO et al., 2015]).*



4.4.5 TESTE COM SINAL DE SINCRONISMO PARA CODIFICAÇÃO TEMPORAL

A Figura 4.17 mostra os *spikes* de oito neurônios trabalhando em codificação temporal. O sinal de sincronismos é mostrado no canal 7. Ele foi configurado com uma frequência $F_S = 4,95$ HZ. É possível notar na Figura 4.17 a repetição de um padrão dos neurônios 1015 ao 1022 (canais Digital 0 ao 6), com sinal de sincronismo, neurônio 1023, apontado na figura. É possível notar que os neurônios mostrados nos canais 0 à 6 repetem um mesmo padrão no tempo para os 3 períodos mostrados na figura. Esse padrão tem como base o sinal de sincronismo mostrado no último canal da tela do analisador lógico. Com a escolha da frequência de operação do sinal de sincronismo e o número de neurônios participantes da codificação, é possível realizar diferentes tipos de codificação temporal com sistema proposto.

Figura 4.17: *Tela do Analisador Lógico mostrando 8 neurônios, onde o sinal de sincronismo é mostrando no canal 7, os demais neurônios formam um padrão em torno do sinal de sincronismo (Figura adaptada de [OLIVEIRA NETO et al., 2015]).*



Resumindo, através dos testes do protótipo foi possível verificar o funcionamento do sistema

proposto. Com ele foi possível gerar 1023 neurônios disparando independentemente com taxas de disparos configuráveis através de escritas nas memórias de configuração em tempo de execução. Com a utilização de um sinal de sincronismo (o neurônio 1023) e o uso de limiares mínimo de disparos configurados na TM, é possível utilizar as codificações temporal e por populações descritas na literatura da neurociência.

CAPÍTULO 5

CIRCUITOS AMOSTRADORES EM NAC

5.1 INTRODUÇÃO

O presente capítulo trata de esquemas de circuitos de entrada em SNN utilizando os conceitos de NAC. Quando dados do mundo são convertidos em *spikes* artificiais, ver capítulo 4, ou biológicos, capítulo 2, eles se encontram esparsos no tempo. Olhando exemplos na eletrônica, esse sinais comportam-se de forma contínua no tempo (sinais de tempo contínuo), ou mesmo como sinais sem sincronismo explícito (sinais de uma comunicação assíncrona). Em NAC, alguns tipos de processamento são realizados sincronamente [RANHEL, 2012c], [RANHEL, 2012a], [RANHEL, 2013]. Isso requer amostrar (discretizar) informações do mundo externo, obtendo exemplos de informações em momentos definidos. No entanto, utilizando o exemplo dos transdutores olfativos e gustativos, cada transdutor funciona independentemente. O conjunto de sinais neurais só é traduzido em informação no córtex, onde a população de sensores que disparam em determinado tempo "significa" um sabor ou cheiro (ver seção 2.3).

Num paralelo com a eletrônica, para transduzir sinais analógicos e contínuos no tempo em números digitais, circuitos ADC necessitam de uma etapa de captura de amostras do sinal. Essa etapa são os circuitos SAH mostrados no capítulo 2. Utilizando como inspiração circuitos eletrônicos e biológicos, serão apresentadas topologias de amostradores para SNNs utilizando os conceitos de NAC. Serão descritas duas topologias de amostradores: uma para trens de *spikes* em codificação por populações e outra para trens de *spikes* em codificação temporal. Após a descrição do circuito, se-

rão mostradas simulações de seu funcionamento feitas no software Matlab. Será também descrita a utilização dos dados amostrados em circuitos de tomada de decisão, testando a usabilidade desses dados.

No estágio atual das investigações em NAC a intenção é compreender os blocos funcionais que permitem criar algoritmos que interagem entre si. Atualmente, são desenhadas topologias, calibrando pesos sinápticos e atrasos temporais para que a rede execute uma função [RANHEL, 2012c], [RANHEL, 2012a], [RANHEL, 2013]. A fim de continuar os estudos de como funcionam tais blocos, este capítulo investiga como informações externas são codificadas e entregues a rede. De tal modo que a abordagem NAC consiga interpretar e realizar operações computacionais sobre tais informações, tendo como modelo os sistemas biológicos e as codificações neurais descritas na neurociência.

5.2 AMOSTRADOR EM POPULAÇÕES

Como dito anteriormente, na eletrônica, as grandezas físicas são geralmente convertidas em sinais elétricos que são analógicos e de tempo contínuo. Os sinais transduzidos pelos sensores biológicos também tem comportamento parecido. Na codificação por populações, o número de neurônios que dispara em dado momento representa a intensidade de uma grandeza, algo análogo a um sinal analógico cujo valor cresce e decresce informando o estado atual da grandeza. Por isso, essas populações de neurônios serão denominadas assembleias analógicas. Isso é diferente do que ocorre com as assembleias comumente trabalhadas em NAC [RANHEL, 2012c], cujo comportamento é que todos os elementos da grupo disparem em conjunto, sincronamente ou polisincronamente. Esse comportamento se assemelha a sinais digitais, por isso, será utilizado o nome de assembleias digitais para essas construções.

Outra característica já apontada dos sinais externos codificados em populações é que essas populações comportam-se semelhantes a sinais contínuos no tempo. Assim como a grandeza física, os neurônios pertencentes à população que representa essa grandeza herdam essa característica. Isso ocorre porque os neurônios pertencentes a população podem disparar com taxas de disparo diferentes entre se, isso gera uma probabilidade diferente de zero de que algum neurônio pertencente a população dispare em um dado instante escolhido. Algo que também difere dos sinais comumente computados em NAC, pois em NAC, o instante em que ocorre um evento é de suma importância para computação [RANHEL, 2012c]. Como as assembleias comumente usadas em NAC possuem instantes específicos em que seus neurônios disparam, essas assembleias assemelham-se a sinais de tempo discreto, enquanto que populações de neurônios podem possuir um comportamento mais semelhante

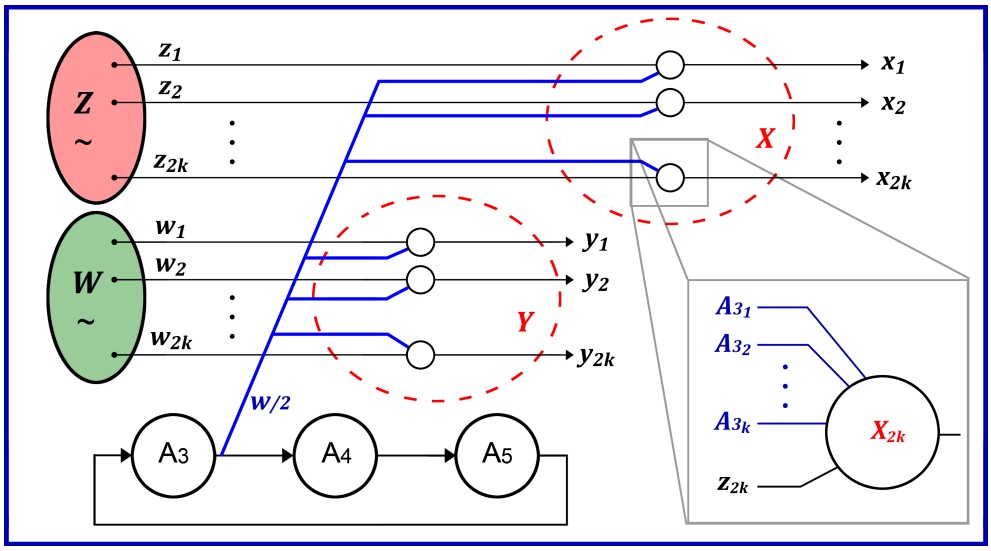
a sinais de tempo contínuo, como as grandezas que elas podem representar. Nesta seção será proposto um circuito amostrador de assembléias de tempo contínuo, gerando uma versão discretizada do sinal original. O circuito é capaz de discretizar uma população de entrada, transmitindo como saída apenas os neurônios que dispararem em instantes específicos.

Uma investigação da interação de assembléias analógicas com as assembléias digitais será abordado na seção 5.2.2

5.2.1 O Circuito Proposto

Na Figura 5.1 é mostrada a arquitetura proposta para o amostrador de populações. Considere duas assembleias Z e X, com igual número de membros. Cada membro i de Z está ligado a um único neurônio i de X. O peso sináptico para tal conexão é $\Theta/2$. Portanto, cada neurônio de Z não pode disparar seu neurônio correspondente em X por si só. É possível ver que cada neurônio de Z só pode causar metade da perturbação Θ necessária para disparar um neurônio de X. No entanto, se outro sinal contribuir simultaneamente com $\Theta/2$, neurônios em X são capazes de disparar. Na verdade, esse sinal adicional atuará como um '*enable*', que permite a X disparar apenas quando ele está ativo. Este sinal de habilitação é fornecido pela assembléia A_3 na rede mostrada na Figura 5.1.

Figura 5.1: *Topologia do circuito amostrador de populações (Figura retirada de [OLIVEIRA-NETO et al., 2014]).*



Ao utilizar a equação 3.5 para a ligação entre $A3$ e X, também é possível calcular o peso sináptico da ligação de cada neurônio de A_3 com cada um dos neurônios de X. Como A_3 deve contribuir com $\Theta/2$, cada neurônio de A_3 está ligado a X com peso:

$$w_i = \frac{\Theta}{2.N} \tag{5.1}$$

Onde N é o número de neurônios que formam a assembléia. Em resumo, a contribuição de todos os neurônios de A_3 disparando provoca um estímulo excitatório em todos os neurônios de X. Mas esta perturbação não é suficiente para provocar o disparo de nenhum dos neurônios de X. Exatamente quando os *spikes* de A_3 estão chegando aos neurônios de X, os *spikes* de Z podem contribuir com $\Theta/2$ para o seu par correspondente em X. Como consequência, os únicos neurônios de X que disparam são aqueles cujo neurônio equivalente em Z está disparando no momento em que A_3 está ativo.

O mesmo raciocínio pode ser aplicado para os neurônios de W e Y. Isso significa que as amostras de X e Y são feitas com período $T = 3\Delta t$. Δt é o tempo que os *spikes* de A_3 levam para chegar em A_4 e é igual ao tempo que os *spikes* de A_4 demoram para chegar em A_5 e os A_5 para chegarem em A_3, formando o ritmo $A_3 \rightarrow A_4 \rightarrow A_5 \rightarrow A_3$ mostrado na Figura 5.1.

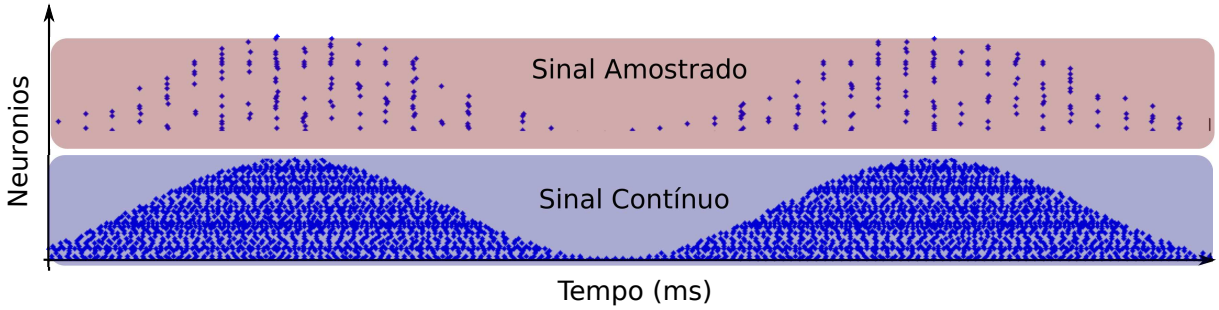
O resultado da simulação da arquitetura mostrada na Figura 5.1 pode ser observado na Figura 5.2. Nela é possível ver uma assembléia analógica de tempo contínuo simulada na parte inferior do gráfico. Para sua simulação foi utilizado uma distribuição de *Poisson* [SOUZA, 2013] para gerar uma taxa de disparos "randômica" constante para cada neurônio da população. Ao mesmo tempo, o sistema modula uma função seno sobre a população, retirando alguns neurônios de atividade e deixando-os voltar a disparar na medida em que a senóide (estímulo externo) aumenta ou diminui, simulando uma população de neurônios cujo número de neurônios ativos muda ao longo do tempo, o sinal mostrado na parte de baixo da Figura 5.2.

Na parte de cima no gráfico é possível ver a assembléia analógica discretizada no tempo. Em razão da taxa de disparos aleatória atribuída a cada neurônio da população, alguns neurônios, embora disparem, nunca são observados na assembléia resultante. Mesmo com "falhas" na assembléia resultante é possível notar que a assembleia amostrada acompanha o sinal descrito pela assembléia contínua.

5.2.2 COMPARANDO MAGNITUDES

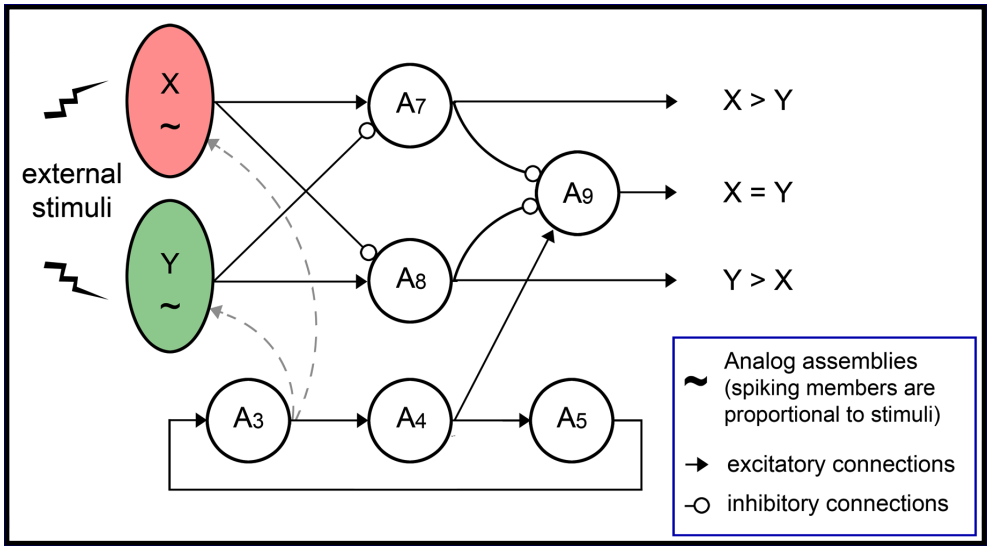
Nessa seção será abordado a interação entre assembleias analógicas e digitais através da descrição de um circuito comparador de magnitudes em NAC. Este circuito compara duas amostras de populações de neurônios. Ele consegue retornar, através de assembleias digitais, qual das duas amostras tem o maior número de neurônios ativos, logo, maior intensidade. A topologia adotada para esse

Figura 5.2: *Simulação mostrando a assembleia de tempo continuo (abaixo) e a assembleia discretizada (acima).*



circuito é mostrada na Figura 5.3.

Figura 5.3: *Esquema do circuito Comparador de Magnitude (Figura retirada de [OLIVEIRA-NETO et al., 2014]).*



Analisando o circuito mostrado na Figura 5.3 é possível determinar a 'perturbação' (o potencial excitatório ou inibitório resultante) nas assembléias A_7 e A_8 gerada pelas assembleias X, Y e A_3. A Figura 5.3 mostra que estas assembléias são ligadas por conexões excitatórias $+w$ e inibitórias $-w$, dadas pelas equações:

$$\Theta_7 = (N_X.w - N_Y.w) \tag{5.2}$$

$$\Theta_8 = (N_Y.w - N_X.w) \tag{5.3}$$

Ou ainda:

$$\Theta_7 = (N_X - N_Y)w \tag{5.4}$$

$$\Theta_8 = (N_Y - N_X)w \tag{5.5}$$

Lembrando que N_X e N_Y indicam o número de neurônios das assembleias X e Y, respectivamente, que estão disparando em um dado instante e w é o peso sináptico da ligação de cada neurônio pré-sináptico com cada neurônio da assembleia pós-sináptica. Já Θ_7 e Θ_8 representam o valor da 'pertubação' que chega às assembleias A_7 e A_8, respectivamente.

Isso significa que as perturbações nos neurônios de A_7 dependem da diferença entre o número de disparos excitatórios vindos de X e o número de disparos inibitórios de Y. O oposto ocorre em A_8. Considere o peso w multiplicado por uma constante α (com $2 \leq \alpha \leq 10$). O fator α permite controlar o quão grande a diferença entre X e Y deve ser. Uma pequena diferença entre o número de estímulos de cada assembléia (N_X e N_Y) multiplicada por um grande número ($w.\alpha$) pode causar a perturbação necessária para ocorrer o disparo de A_7 ou A_8. O valor de α foi variado durante as simulações e, na simulação descrita neste capítulo, foi utilizado $\alpha = 4$. Como descrito acima, A_7 e A_8 são assembléias digitais cujos neurônios dispararam todos juntos no tempo, utilizando o modo *synfire*. Portanto, A_7 poderá disparar quando X for ligeiramente maior que Y, e A_8 poderá disparar quando Y for ligeiramente maior que X.

A assembleia A_9 deve disparar quando a diferença de membros de X e Y que disparam num instante não é suficiente para disparar a A_7 ou A_8. A maneira mais simples de criar uma rede que resolva este problema é a utilização de A_7 e A_8 como estímulos inibitórios para um circuito que é sempre ativo. No entanto, as respostas vindas de A_7 e A_8 demoram algum tempo para serem obtidas. Após disparar A_3 (amostragem de X e Y) a rede leva Δt para os *spikes* de X e Y alcançarem A_7 e A_8. Em seguida, é preciso outro tempo Δt para os *spikes* de A_7 e A_8 chegarem a A_9. Simultaneamente A_3 dispara A_4, e uma vez que A_4 seja ligado a A_9, consequentemente, *spikes* de A_4 alcançarão A_9, ao mesmo tempo que os *spikes* de A_7 e A_8. Podendo ser denotado pela equação 5.6:

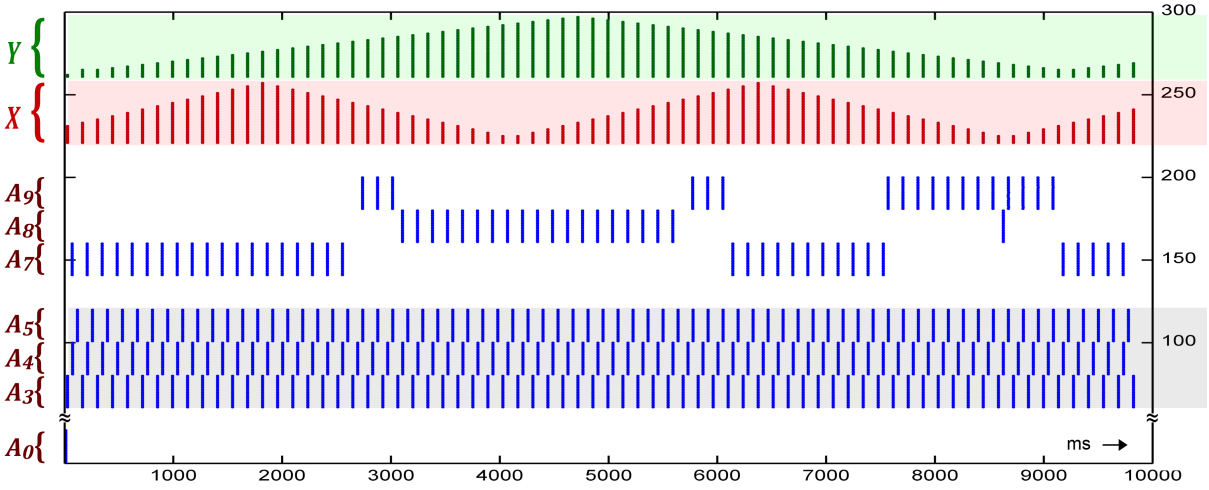
$$\Theta_9 = (N_{A_4}.w_{A_4} - N_{A_7}.w_{A_7} - N_{A_8}.w_{A_8}) \tag{5.6}$$

Analisando a equação 5.6 é possível notar que o disparo de A_9 está sempre sendo provocado por A_4. Mas se A_7 ou A_8 dispararem, a pertubação resultante (Θ_9) não será suficiente para fazer a assembléia A_9 disparar. Portanto, A_9 só é acionada quando nem A_7 nem A_8 estão ativos.

5.2.3 SIMULAÇÕES

A fim de simular a topologia proposta foi utilizado o software Matlab [MATHWORKS, 2015]. Os resultados podem ser vistos na Figura 5.4. A Figura mostra o *raster plot* com a distribuição de neurônios disparando ao longo do tempo. Cada ponto cheio no gráfico significa que um neurônio 'k' disparou no instante 't' ms, onde 'k' é o eixo das ordenadas e o 't' o eixo das abscissas. Como um meio para facilitar a visualização, cada assembléia tem seus membros sequencialmente atribuídos. Por exemplo, neurônios de 1 a N pertencem a assembléia A_0, a partir de $N+1$ a $2N$ pertencem a A_1 e assim por diante.

Figura 5.4: *Resultado da simulação do Comparador de Magnitude (Figura retirada de [OLIVEIRA-NETO et al., 2014]).*



Nesta simulação, as assembléias foram compostas por vinte neurônios cada. As exceções são as assembléias analógicos X e Y, com quarenta neurônios cada. Para todas elas, modelo de neurônio utilizado foi o *Izhikevich's Simple Model* (IZH) [IZHIKEVICH, 2003], [IZHIKEVICH, 2004]. Assembléias operam em modo *synfire*. Na Figura 5.4, o efeito *synfire* pode ser facilmente observado porque os neurônios pertencentes a uma mesma assembléia disparam todos ao mesmo tempo.

Olhando para a simulação, a assembléia A_0 simplesmente inicia a SNN e A_1 e A_2 não são utilizadas. Com o disparo de A_0 é iniciado o ciclo rítmico formado pelas assembleias A_3, A_4 e A_5. Como mencionado anteriormente, este *clock* é responsável por sincronizar os outros processos e se repete indefinidamente.

Na parte superior da Figura 5.4, é possível ver as assembléias X e Y. Elas variam o número de neurônios disparando ao passar do tempo. Nota-se que X e Y possuem "frequências" diferentes (o número de neurônios ativos cresce e decresce em frequências distintas para X e Y) de modo a

abranger todos os possíveis resultados do comparador.

Quando X é maior do que Y, é possível observar que apenas a assembléia A_7 dispara. Por outro lado, somente a assembléia A_8 dispara quando Y é maior do que X. Existem 'hiatos', em que o circuito não conseguem responder se $X > Y$ ou para $Y > X$. Esse intervalo pode ser aceitável e, às vezes, até desejável. Isso mostra que a rede indica apenas quem é maior com um grau de certeza configurado pela constante multiplicativa α. A igualdade é indicada pela assembléia A_9, que é esperado que dispare apenas quando nem A_7 quanto A_8 estão disparando.

5.3 AMOSTRADOR UTILIZANDO *Bursts* PARA CODIFICAÇÃO TEMPORAL

Os amostradores investigados na seção anterior dependem da coincidência de *spikes* e da correlação apropriada de pesos sinápticos. Neles, as janelas temporais podem ser tão pequenas quanto 1ms (*synfire*). Um amostrador para codificação temporal, ilustrado na Figura 5.6, trabalha como uma "porta" (*gate*). Esta porta fica aberta por uma janela de tempo necessária para que um código de *spikes* deslocados no tempo possa passar (ver seção 2.4).

A arquitetura proposta requer um sinal de *burst* para comandar a abertura de cada neurônio individualmente. Isso porque este tipo de amostrador requer que os neurônios fiquem "abertos" a estímulos durante um intervalo temporal maior. Um tempo maior de excitação possibilita arranjos temporais de neurônios disparando codificando informações. Em outras palavras, enquanto a janela temporal para codificação por populações pode durar apenas o tempo que assembleia está ativa, a janela temporal para codificação temporal deve abranger uma escala temporal maior. Durante este intervalo maior, os *spikes* de todos os neurônios devem passar. *Layers* subsequentes de neurônios cuidam da interpretação dos *spikes* que passaram pela porta. Há situações que a leitura e interpretação dos *spikes* podem acontecer na própria porta, como será visto no exemplo de circuito de tomada de decisão descrito na seção 5.3.6.

O comportamento de *burst*, ou rajadas de *spikes*, é encontrado em neurônios biológicos, como foi mostrado na Figura 2.3. Será necessário que os modelos de neurônio artificial repliquem esse comportamento. O que torna interessante aprofundar mais no comportamento dos modelos de neurônios artificiais. Em razão disso, a seção seguinte irá explanar sobre o comportamento dos modelos de neurônio utilizados nas simulações deste trabalho.

5.3.1 MODELOS DE NEURÔNIOS UTILIZADOS

Nas simulações mostradas nesse capítulo foram utilizados dois modelos de neurônios artificiais: o *Leaky Integrated and Fire* (LI&F) e o IZH [IZHIKEVICH, 2003], [IZHIKEVICH, 2004]. Na rede LI&F é utilizada a equação 5.7, onde v é o potencial interno da membrana do neurônio, τ_m é a constante de tempo da membrana, R_m é a resistência da membrana, v_e é o potencial de repouso e I é o vetor de corrente de entrada. Todos os valores são escalonados em mV e ms. Foi usado $\tau_m = 5$, $R_m = 3,4\ \Omega$, $v_e = -65$ mV e um potencial limiar de -50 mV.

$$\tau_m v' = -[v + v_e] + R_m I \tag{5.7}$$

Já o IZH é descrito pelas equações 5.8 e 5.9:

$$v' = 0,04v^2 + 5v + 140 - u + I \tag{5.8}$$

$$u' = a(bv - u) \tag{5.9}$$

Onde a regra de pós disparo é:

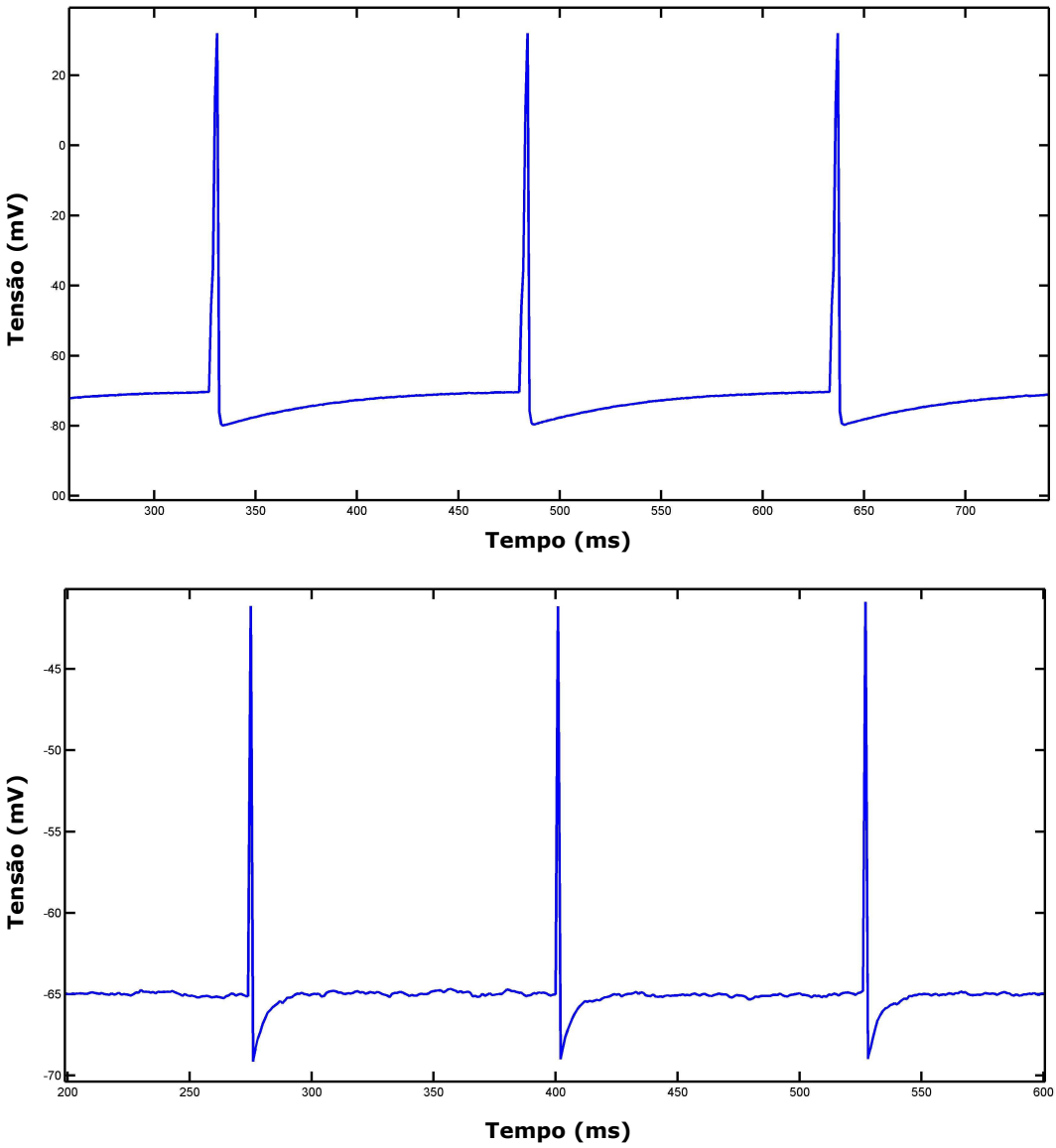
$$\text{se} \quad v \geq 30mV \quad \begin{cases} v & \leftarrow c \\ u & \leftarrow u + d \end{cases} \tag{5.10}$$

Para as equações 5.8, 5.9 e 5.10, v é potencial interno da membrana do neurônio, u é o parâmetro de recuperação da membrana e a, b, c e d são parâmetros que mudam as características dinâmicas do neurônios. Os parâmetros para cada tipo do neurônio podem ser vistos em Izhikevich [IZHIKEVICH, 2003]. Note que, para ambas as equações dos modelos, I é uma corrente excitatória ou inibitória que chega ao neurônio. Através desse vetor de entrada é possível injetar ruído na rede, como será mostrado em seções seguintes. Na Figura 5.5 é mostrada a curva da tensão em função do tempo para os modelos de neurônio IZH (acima) e LI&F (abaixo). Os parâmetros utilizados nesta figura para o modelo do IZH foram o do *Tonic Spiking* [IZHIKEVICH, 2004].

5.3.2 GERANDO *Bursts*

Bursts não acontecem espontaneamente no modelo LI&F, porque este neurônio naturalmente não consegue disparar em *bursts*. Por outro lado, o IZH consegue simular neurônios que respondem em rajadas de *spikes* dependendo dos parâmetros a, b, c e d escolhidos [IZHIKEVICH, 2003].

Figura 5.5: *Tensão da membrana dos neurônios em função do tempo: modelo IZH acima e modelo LI&F abaixo.*



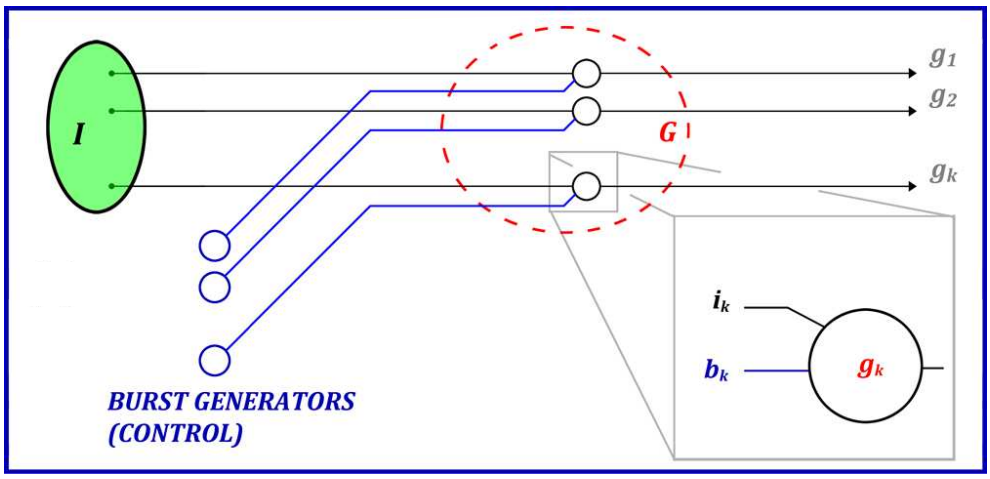
Para resolver o problema da inexistência de *bursts* de *spikes* no LI&F, *spikes* foram forçados para imitar *bursts* em todas as simulações que utilizam o LI&F. Foi levado em consideração a quantidade de *spikes* por *burst* e o intervalo entre os *spikes*, com estes fatores é possível controlar o tempo de duração da rajada de *spikes*. Já para o IZH foi possível utilizar a geração natural de *bursts* que ele possui. No entanto, nos testes realizados também foram simuladas redes com o IZH com a estratégia adotada com o LI&F. Os tipos de IZH utilizados foram o *Tonic Bursting, Phasic Bursting,* e o *Bursting Rebound*.

Os *bursts* são aplicados diretamente nos neurônios que atuam como *gate*. O peso sináptico desempenha um papel importante, ele também é uma variável que controla o tempo de excitação em que a porta ficará aberta. Ainda, EPSPs muito intensos podem disparar o *gate* sem a existência de um estímulo a ser copiado. Assim como o caso contrário é um problema de igual dimensão: EPSPs muito fracos não deixariam os neurônios do *gate* excitados o suficiente para que um estímulo externo gerasse um *spike*. A investigação desses limites será explicada nas seções seguintes. Mas antes será mais bem explicada topologia do amostrador por *bursts*

5.3.3 O Circuito Proposto

A Figura 5.7 ilustra a simulação dos neurônios ligados como circuito descrito na Figura 5.6 para testar as portas controladas por *bursts*.

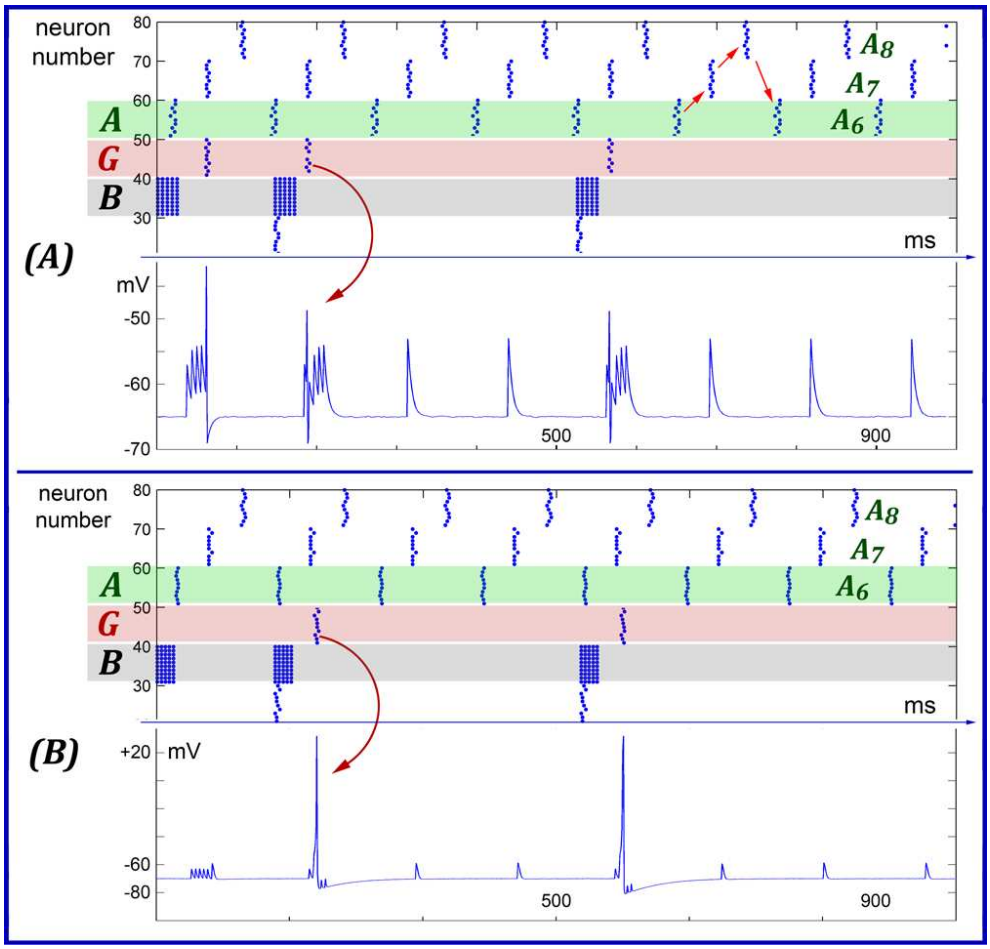
Figura 5.6: *Circuito Esquemático do Amostrador por Bursts.*



Enquanto a Figura 5.7A mostra o *raster plot* para rede que usa o modelo LI&F, a Figura 5.7B mostra o mesmo circuito com o modelo IZH. NAC usa codificação de neurônios esparsa e paralela para representar algo. Essa redundância aumenta a confiabilidade e a robustez do sistema. Assim,

por premissa, NAC em vez de usar um único neurônio para a criação de um *gate* foi utilizado um conjunto de células. No entanto, é possível ter um único neurônio funcionando como *gate*. Em razão disso, foi pego um único neurônio para analisar o que acontece no potencial de sua membrana. No entanto, cada assembleia (A, B e G) mostrada na Figura 5.7 tem um número N de neurônios. Os testes foram realizados com $N = 10$, $N = 25$ e $N = 100$. Funcionalmente, a rede sempre apresentou o mesmo resultado.

Figura 5.7: *Simulação dos gates com bursts. Em* ***(A)*** *foi utilizado o modelo LI&F e em* ***(B)*** *o modelo IZH. Para os dois casos, pode-se notar que, enquanto o burst B excita os neurônios da porta G os spikes da assembleia aferente A passam pelo gate. Quando o burst B desaparece, os spikes da assembleia A não passam pelo gate G, mesmo quando os neurônios de A disparam.*



A funcionalidade é independente do número de neurônios da assembleia por causa da maneira como os pesos sinápticos são calculados através da equação 3.5. No entanto, para a porta controlada por *bursts* é usada uma forma diferente para criar a topologia mostrada na Figura 5.6. As ligações são feitas com cada neurônio A_i de A está ligado a um único neurônio G_i de G por um peso sináptico de

Θ/X_A. Por outro lado, cada neurônio B_i é ligado a um único neurônio correspondente G_i por um peso igual a Θ/X_B. Onde os *bursts* são simulados pela assembléia B. Então, quando os neurônios A_i disparam cada neurônio irá contribuir com um potencial excitatório ou inibitório de Θ/X_A para os neurônios correspondentes da porta G.

O *burst* gerado por cada neurônio B_i mantém uma contribuição média de EPSPs para cada neurônios G_i correspondente. Então G permanece parcialmente excitado durante o tempo em que os *spikes* do *burst* estão chegando. Caso um dado neurônio de A_i dispare durante esse tempo o neurônio G_i correspondente poderá disparar. Isto é ilustrado na Figura 5.7. É fácil notar que a remoção do *burst* fecha o porta e disparos dos neurônios aferentes A_i não conseguem excitar neurônios de G o suficiente para que ele dispare, também mostrado na Figura 5.7. Nessas simulações as assembleias utilizadas são grupos poli-síncronos, como é possível observar.

5.3.4 RESULTADOS

A primeira coisa que foi ajustada para a simulação da rede foi o valor dos pesos sinápticos (Θ) para as assembleias que marcam o ritmo ($A_6 \rightarrow A_7 \rightarrow A_8 \rightarrow A_6$), mostradas na Figura 5.7. Este procedimento foi feito ajustando o valor de Θ de forma que este *loop* rítmico operasse em regime estável, determinando um Θ_{min} estável para o funcionamento da rede.

Ao alternar o valor de Θ/X_A, a contribuição que o sinal que cada neurônio aferente A_i aplica ao neurônio G_i correspondente é modificada. Assim como é alterada a contribuição do sinal de *burst* que cada neurônio B_i aplica ao neurônio G_i correspondente se o valor de Θ/X_B for modificado. Os limites máximos e mínimos para X_B, fixando o valor $X_A = 2$, são mostrados na 5.1.

Após serem realizadas as simulações pode-se compreender as diferenças básicas entre o *gates* simulados como o modelo LI&F e com o modelo IZH. Como mostrado anteriormente, a Figura 5.5 mostra as formas de ondas típicas da tensão de membrana para os dois modelos de neurônio. A forma de onda do LI&F é bem típica de um circuito RC. Já a forma de onda do IZH não é tão fácil de analisar pelo fato do modelo do neurônio representar um sistema dinâmico. Nas simulações com IZH, foi utilizado primeiramente o neurônio tipo *Tonic Spiking*, mas também a rede foi testada e calibrada (ajuste do peso sináptico descrito anteriormente) para funcionar com os neurônios do tipo *Phasic Spiking*, *Mixed Mode* e *Spike Frequency Adaptation*, cujo comportamento pode ser visto na Figura 2.3. Funcionalmente, não houve diferença na operação da rede utilizando os diferentes tipos de neurônio e os valores de Θ_{min} encontrados para os diferentes tipos de neurônios utilizados são mostrados na Tabela 5.1, mostrada na seção seguinte.

5.3.5 ADICIONANDO RUÍDO

Os resultados foram analisados baseando-se na resposta funcional da rede. Ou seja, foi avaliado se a rede está realizando a tarefa determinística para qual foi projetada. Contudo, o ambiente de redes neurais biológicas é ruidoso. Para testar o desempenho do circuito em situações adversas, foi injetado um sinal de ruído randômico no vetor de entrada I dos neurônios. A função $rand(.)$ do Matlab foi utilizada para gerar uma distribuição normal de números pseudo-randômicos. A corrente de entrada foi inicializada através da equação 5.11:

$$I(t) = \epsilon.rand(nN, 1) \tag{5.11}$$

Onde t é o tempo de uma dada iteração, ϵ é o fator de ruído e nN é o número total de neurônios na rede.

Os resultados da rede com ruído e sem ruído estão apresentados na Tabela 5.1. Nela é possível ver o valor de Θ usado para cada tipo de neurônio. Esse valor foi alcançado de forma individual através de testes. É possível ver que para o neurônio tipo *Phasic Spiking* o valor de Θ é bem baixo e que é o neurônio cujos dados mais destoam dos demais para toda a tabela. No restante da tabela existe uma homogeneidade dos dados. Os valores de Θ foram encontrados para o circuito sem ruído e depois adicionando o ruído e encontrando os valores mostrados na quarta coluna de dados.

Tabela 5.1: *Resultado do Amostrador por Bursts sob ruído para diferentes tipos de neurônio.*

Tipo de Neurônio	Θ_{min}	$X_{B,MIN}$	$X_{B,MAX}$	ϵ_{MAX}
LI&F	37	2,22	4,4	1,7
Tonic Spiking (IZH)	25	1,90	4,6	1,0*
Phasic Spiking (IZH)	7	3,50	11,6	0,8*
Mixed Mode (IZH)	25	1,91	4,6	1,1*
Spiking Freq. Adaptation (IZH)	25	1,99	4,3	0,7*

$\star$ Valor de ruído para qual o circuito gerador de ritmo para de funcionar.

Na segunda e terceira colunas da Tabela 5.1 são mostrados os valores máximos e mínimos para X_B. Dentro dessa faixa de valores o circuito funciona corretamente como *gate* desejado. Para valores menores que o $X_{B,MIN}$ o *burst* sozinho é capaz de disparar os neurônios do *gate*, inviabilizando o circuito. Já o valor $X_{B,MAX}$ é o valor máximo que é possível usar para X_B de forma que a rede consiga funcionar como *gate*. Esse valor foi alcançado para um sinal aferente com peso de $\Theta/2$, ou

seja, $X_A = 2$.

Como mencionado acima, na quarta coluna da Tabela 5.1 é mostrado o valor encontrado para o fator de ruído ϵ que o circuito consegue receber e ainda continuar funcionando como um *gate*. Para esse estudo foi usado um valor médio de X_B entre os valores mostrados na tabela, variando então para cada tipo de neurônio. Mas o peso sináptico para cada tipo de neurônio ficou em torno de 35 $\%$ do valor de Θ. Algo interessante é que o LI&F é menos sensível a ruído do que o IZH. Com isso foi possível encontrar um valor para o qual o circuito com o LI&F para de funcionar como *gate* devido apenas ao ruído. No entanto, para o IZH ocorreu algo inusitado, o circuito gerador de ritmo das assembléias ($A_7 \rightarrow A_8 \rightarrow A_9 \rightarrow A_7$) deixa de funcionar antes que o *gate* apresente qualquer "defeito". Então, na tabela são mostrados os valores para os quais o circuito deixou de funcionar como um todo ($\epsilon^\star$) e não devido ao *gate*. Dado o valor encontrado de ruído para o LI&F e o ocorrido com as redes com o IZH , o *gate* proposto é um arquitetura resistente ao ruído na rede, garantindo sua robustez.

5.3.6 TIME-TO-FIRST SPIKE

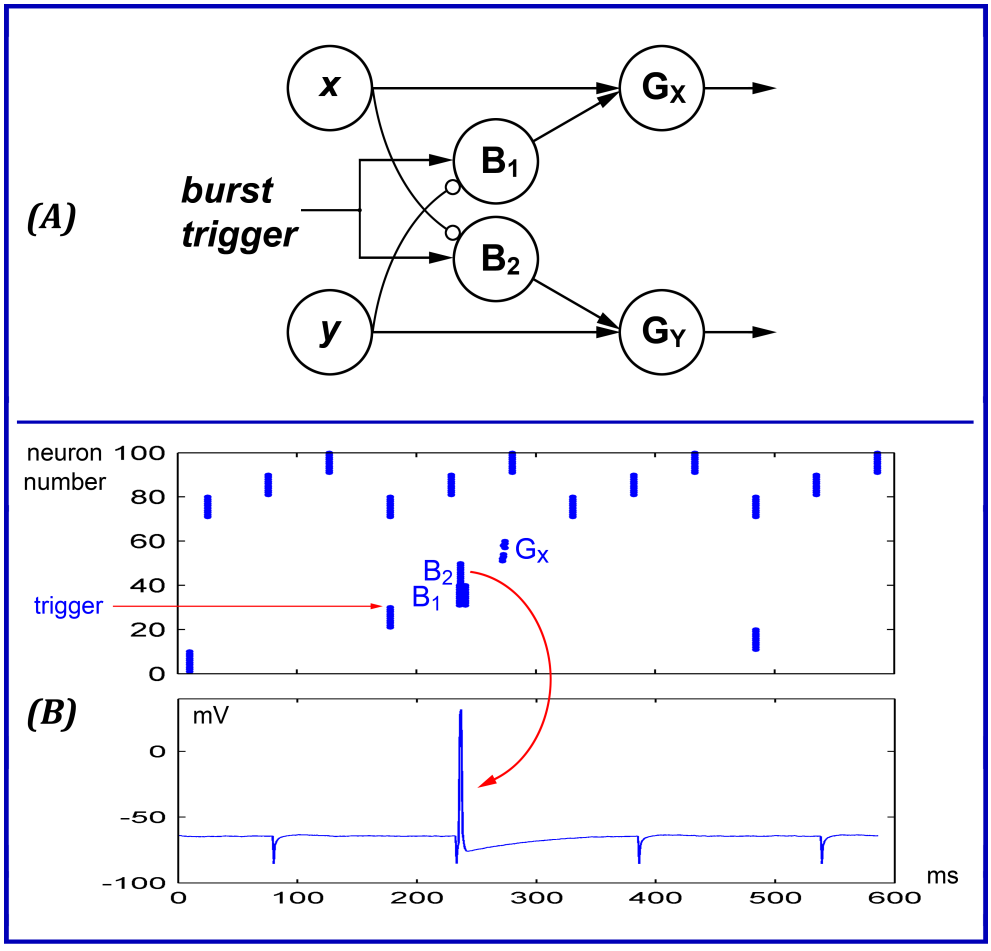
Fazendo pequenas alterações na topologia do amostrador apresentado na seção 5.3, é possível incorporar ao circuito a tomada de decisão entre sinais de entrada codificados temporalmente. O circuito mostrado na Figura 5.8A identifica apenas o primeiro *spike* a chegar, dentro de uma população de neurônios ligados ao *gate*. Esse tipo de codificação temporal é chamado na literatura da neurociência de codificação por latência, ou *time-to-first spike*. Esse tipo de codificação é encontrada nos sistemas somatossensorias, auditivo, olfativo entre outros [VANRULLEN et al., 2005].

A Figura 5.8B mostra o *raster plot* de uma simulação do circuito. Note que o circuito mostrado na Figura 5.8 além de *gate*, está executando a função de comparação entre dois grupos de sinais, elegendo a assembleia cujos *spikes* chegaram primeiro.

O circuito tem como entrada duas assembleias aferentes X e Y ligadas aos *gates* G_X e G_Y, respectivamente. Onde, cada neurônio X_i de X está ligado ao neurônio $G_{X,i}$ correspondente em G_X. Sendo a ligação de Y a G_Y análoga.

Assim como ocorre no amostrador descrito anteriormente, um sinal de *burst* é usado para deixar a porta aberta por uma janela de tempo determinada. Esse sinal de *burst* é gerado pelas assembleias B_1 e B_2. Onde, B_1 tem seus neurônio ligados individualmente aos neurônios de G_X, enquanto os neurônios de B_2 são ligados aos neurônios de G_Y. Ambas as ligações obedecem a estrutura descrita para X e G_Y e Y e G_Y.

Figura 5.8: *Esquemático do circuito amostrador e codificador em Time-to-First Spike*



É fácil notar que o circuito descrito até agora é idêntico ao *gate* mostrado anteriormente. No entanto, a ativação das assembleias B_1 e B_2 difere do circuito apenas amostrador. Além de serem excitadas para que gerem o sinal de *burst* em instantes de tempo pré estabelecidos, elas possuem ligações inibitórias vindas das assembleias aferentes. No caso, X está ligada de forma inibitória a B_2, enquanto Y está ligada a B_1, também com ligação inibitória.

Outra diferença no novo circuito é o tempo de propagação do sinal entre as assembleias: Os *bursts* gerados por B_1 e B_2 levam um tempo Δt para alcançarem as assembleias G_X e G_y, respectivamente. Enquanto que os sinais aferentes vindos de X e Y demoram um tempo $2.\Delta t$ para chegarem em G_X e G_Y. Ao mesmo tempo, esses sinais vindos de X e Y levam um tempo Δt para inibirem B_1 e B_2.

O comportamento do circuito no tempo é mostrado na Figura 5.8B. O sinal de *trigger* das assembleias B_1 e B_2 acontece sincronamente, fazendo com que elas gerem *bursts* ao mesmo tempo. No entanto os sinais aferentes de X e Y são capazes inibir os *burts* fechando os *gates* G_X e G_Y. Observando o que ocorre na Figura 5.8B, foi utilizada a mesma assembleia para excitar tanto B_1 quanto B_2. Utilizando a assembleia A_7, foram simuladas as duas assembleias X e Y de tal modo que os *spikes* de X chegam ao circuito de *gate* alguns milissegundos antes dos de Y. Logo, X inibe a assembleia B_2 fazendo com que o *gate* G_Y esteja fechado quando os *spikes* de Y cheguem a G_Y. Portanto, o circuito deixa passar apenas a assembleias cujos *spikes* chegaram primeiro, dentro da janela temporal do *burst*.

Para ser possível ativar e inibir os *bursts* através de assembleias, foi utilizado o modelo de neurônio IZH com os parâmetros dos tipos *Tonic Bursting* e *Phasic Bursting* nas assembleias B_1 e B_2, não havendo diferença no comportamento do circuito para os dois tipos de neurônio. Para as demais assembleias foi utilizado o tipo *Tonic Spiking*.

Como explicado anteriormente, o passo seguinte à transdução de *spikes* para as SNNs em tempo real é que grupos de *spikes* precisam ser coletados como amostras discretas para que se tornem representações internas na abordagem NAC. As topologias desenvolvidas para esse fim funcionam como '*gates*' ou portas que podem ser abertas de duas formas: uma síncrona, por meio de assembleias neurais, por exemplo, e a outra como uma janela temporal, por meio de *bursts*, já que é característico dos *bursts* permanecerem disparando por um tempo. Mesmo sob efeito de ruído as topologias de amostradores funcionaram satisfatoriamente dentro dos padrões desejados. Com os sistemas de tomada de decisão descritos foi possível verificar a adequação de sinais codificados temporalmente em populações às topologias anteriormente estudadas em NAC [RANHEL, 2012c], [RANHEL, 2012a], [RANHEL, 2013].

CAPÍTULO 6

DISCUSSÕES E CONCLUSÕES

O objetivo deste capítulo é discutir os resultados alcançados durante este trabalho, assim como a que conclusões eles levaram. Primeiramente, é descrito a analise dos resultados de cada parte do projeto, seguida pelas conclusões do trabalho como um todo. Ainda é conteúdo desde capítulo mostrar as publicações relacionadas ao trabalho e indicações de trabalhos futuros.

6.1 DISCUSSÕES

No trabalho dissertado foram mostradas duas partes singulares da construção de um sistema que controle um agente através de SNNs de tempo real. Por se tratar de um projeto com vários desafios particulares que devem ser superados individualmente, foram escolhidos como foco dessa dissertação: como gerar *spikes* para SNNs que operam em tempo real e a investigação de topologias de amostradores de *spikes* (o *input circuit* e os *samplers* mostrados na Figura 1.1).

Por se tratarem de objetivos interligados devido ao projeto, mas que possuem resultados independentes, suas discussões serão feitas em seções separadas.

6.1.1 DISCUSSÕES: CODIFICADOR NEURAL E GERADOR DE *Spikes*

No capítulo 4 foi descrito o GCCst, um sistema digital que converte dados em trens de *spikes* utilizando uma FPGA e baixo custo (custo do chip FPGA $< \$12$ [DIGIKEY, 2015]). O protótipo de teste teve exito em gerar 1023 neurônios podendo ser configurado em tempo de execução para trabalhar com os três tipos de codificação descritos na literatura da neurociência. Levando em consideração que simulações em SNNs, que utilizam a escala temporal dos circuitos biológicos, não tem mos-

trado capacidade de resolução na faixa dos microssegundos [RANHEL, 2012c], [OLIVEIRA-NETO et al., 2014], [VIEVILLE and CRAHAY, 2004], e, analisando as medidas da Tabela 4.1, apenas um pequeno desvio entre os dados mensurados e os dados esperados foi encontrado, menor que $0,06$ %, que está dentro da faixa de microssegundos, o que reforça que o sistema é preciso o suficiente para gerar *spikes* para SNNs que trabalhem em tempo real. Além de que, o software USBee Suite, utilizado nas medições, calcula as frequência período por período, não sendo verificada mudança na frequência para diferentes períodos de um mesmo sinal ou para diferentes dados coletados.

Devido ao baixo consumo de LEs (*Logic Elements*), é possível replicar vários módulos dentro de uma mesma FPGA. Como mostrado previamente, a interface de entrada pode endereçar até 32 módulos GCCst, utilizando a interface de comunicação adotada no protótipo de testes. Ainda, como o InC trabalha com o *clock* do sistema de 100 MHz, taxas elevadas de comunicação podem ser utilizadas. Cada módulo GCCst ocupa 4 % dos blocos lógico e 9 % da memória RAM da FPGA utilizada. Sendo possível, em tese, sintetizar até 10 módulos nessa FPGA de baixo custo, gerando 10.240 neurônios. Este número ainda está longe dos sistemas biológicos, pois, o sistema aferente de um organismo biológico (por exemplo um mamífero) pode ter alguns milhões de transdutores [BEAR et al., 2007]. Contudo, o sistema proposto neste trabalho pode perfeitamente ser utilizado em um robô (agente) que seja controlado por SNNs em tempo real.

Para alcançar duas ordens de magnitude, atingindo aproximadamente 1 milhão de neurônios, pode-se utilizar FPGAs com maior número de recursos, ou mesmo várias FPGAs de baixo custo trabalhando paralelamente. Ainda assim, haverá um limitante que é a grande quantidade de entradas necessárias. No entanto, em um sistema visual, uma câmera CMOS de baixo custo, com 640x480 pixels [OMNIVISION, 2011], resulta em aproximadamente 1 Mbytes por quadro a serem convertido para *spikes* (desconsiderando compressões). Ampliado para versões em múltiplas FPGAs e/ou FPGAs de maior desempenho, o sistema GCCst pode ser utilizado para gerar as entradas do sistema visual de um robô controlado por SNNs de tempo real.

Com respeito às saídas paralelas, o sistema permite configurar saídas individuais que emulam axônios de neurônios, como as 8 saídas utilizadas para testes do protótipo. Neste caso, a limitação do número de saídas paralelas é devido ao número de pinos de saída disponíveis na FPGA escolhida. Existem FPGAs com grande número de pinos de saída, no entanto, a que foi usada permite configurar até 48 pinos como axônios de neurônios disparando paralelamente. Outra possibilidade para expandir o número de axônios é usar CPLDs de baixo custo apenas para expandir o número de pinos.

Uma comparação direta com sistemas similares não foi possível até o momento. Na literatura fo-

ram encontrados poucos exemplos de SNNs trabalhando em tempo real e mesmo esses poucos exemplos não especificam como transformar sinais digitais e/ou analógicos em *spikes* para a rede [BENJAMIN et al., 2014], [NEFTCI et al., 2013], [POON and ZHOU, 2011], [CHEUNG et al., 2012], [WANG et al., 2014], [WANG et al., 2013], [RICE et al., 2009], [CASSIDY et al., 2007], [SHAYANI et al., 2008]. Este é um campo de pesquisa relativamente recente e apenas poucos sistemas estão em operação. Por outro lado, sistemas comerciais neuromórficos são protegidos por patentes e o sistema de entrada de dados de cada rede não é claramente descrito, por exemplo, o *Brain Power project* da IBM [IBM, 2014] e o *Qualcomm Zeroth Processors* [QUALCOMM, 2014].

Uma comparação com um exemplo analógico do circuito sensor foi mostrada no capítulo 2. Dentre as vantagens do trabalho apresentado nessa dissertação é possível citar a modularidade de flexibilidade do sistema, podendo ser facilmente adaptado para funcionar com os diferentes tipos de sensores comerciais, além de reduzir massivamente à quantidade de fios necessários para transmissão do sinal, pois utiliza uma interface de comunicação serial. Como contras, é possível destacar a necessidade de um bloco conversor Analógico para Digital dos sinais. No entanto, um módulo de adequação de sinais também é necessário na alternativa analógica, caso sejam utilizados sensores comerciais cujas frequências, intensidades ou outras características não sejam compatíveis com o neurônio sensor analógicos.

Mesmo que o sistema tenha sido desenvolvido para SNNs de tempo real, é possível usar o sistema como hardware dedicado para gerar sinais para qualquer tipo de SNN. Como ilustração, é possível converter um conjunto de dados de computador em grupos de *spikes*, desde que se faça a devida adaptação para salvar os *spikes* gerados em arquivos que serão posteriormente aplicados aos simuladores de SNNs. Devido a esta modularidade, baixo custo e consumo, o hardware pode ser facilmente adaptado para uso como módulo de entrada em sistemas embarcados controlados por SNNs.

Por outro lado, por trabalhar com a base de tempo dos sistemas biológicos, uma interação entre esses sistemas biológicos e o proposto nesse trabalho não está descartada. No entanto, além de maiores estudos sobre codificação neural num enfoque mais específico para cada aplicação e como também a construção de um módulo de adequação dos sinais para os níveis de tensão e corrente dos sinais biológicos são necessários para essa interação ser possível.

6.1.2 DISCUSSÕES: CIRCUITOS AMOSTRADORES

No capítulo 5 foi investigado como trens de *spikes* podem ser amostrados dentro da SNN, em particular, utilizando a abordagem NAC. O capítulo 5 pode ser dividido em duas partes: *spikes* codi-

ficados em populações e *spikes* em codificação temporal.

Na primeira parte foi proposta uma topologia de amostrador de populações, no qual um conjunto de neurônios faz a função de uma porta, *gate*. Esse amostrador deixa passar *spikes* de uma população de neurônios em curtos intervalos de tempo pré-programados. Os neurônios ativos da população formam um subconjunto que representa (ou quantifica) algo. Isso significa que a rede pulsada memoriza um *sample* da população. Por exemplo, obtidos *samples* das populações X e Y representando grandezas diferentes, ou mesmo uma mesma grandeza como uma diferença temporal entre as amostras, é possível decidir se $X > Y$, ou se $Y > X$, como mostrado na seção 5.2.2. Esta operação de tomada de decisão a partir de dados de entrada é importante para computação em NAC, tendo como sinais de saída informações entregues a processos subsequentes dentro da rede.

Na segunda parte do capítulo 5 foi descrita a topologia de um amostrador que colhe amostras de dados codificados em codificação temporal. Para isso foi utilizado uma estratégia bioinspirada em circuitos neurais, onde existem classes de neurônios que respondem a estímulos através de rajadas de *spikes* (*bursts*) [KANDEL et al., 2000], [IZHIKEVICH, 2004]. Nesse amostrador, o *burst* funciona gerando um EPSP médio em um nerônio que funciona como porta. Essa excitação cria uma janela de tempo em que a porta está "aberta" para receber estímulos aferentes que serão amostrados. Trabalhando ainda com codificação temporal, foi criado um circuito de tomada de decisões que utiliza a codificação por latência, ou *time-to-first-spike*. Esta porta temporal realiza uma função que, em certo modo, é semelhante ao circuito de comparação entre populações X e Y. Admitindo que um neurônio aferente seja excitado por um estímulo excitatório, parece lógico supor que quanto mais intenso for tal estímulo mais rápido um neurônio sensor responde [PAUGAM-MOISY and BOHTE, 2010].

O tempo de resposta, da abertura do *gate* até a tomada de decisão, é igual para os dois circuitos porque as operações são sincronizadas com o temporizador interno que controla quando as assembleias disparam. O amostrador em codificação por populações exige pelo menos três assembleias com N neurônios cada: a assembleia aferente , a assembleia que habilita a porta, e a assembleia que funciona como porta. O circuito em codificação temporal através de *bursts* também exige esta configuração. Como qualquer das assembleias pode ter um número variável de neurônios, não há vantagem de um tipo em relação ao outro que possa ser destacada em relação ao número de neurônios utilizados no *gate*. No entanto, como foi dito anteriormente, os *gates* para codificação temporal podem trabalhar com assembleias de um único neurônio, enquanto que o número de neurônios de uma população na codificação por população tem uma relação direta com a resolução do sinal transduzido. Além do mais, por se tratarem de codificações diferentes, os circuitos subsequentes aos

amostradores lidarão com sinais de natureza diferente, logo, o estudo de ambos é interessante, assim como dos circuitos de tomada de decisão abordados.

De uma forma geral, as saídas de ambos os circuitos podem ser usadas para disparar (excitar) ou desmantelar (inibir) alguma assembleia, que por sua vez pode ser parte de uma máquina de estado ou de um algoritmo interno da rede. O que torna as topologias de circuitos descritas úteis dentro dos circuito trabalhados em NAC até o momento.

6.2 CONCLUSÕES

Com o objetivo principal de investigar como gerar trens de *spikes* em tempo real para excitação de SNNs que trabalham também em tempo real, foi proposto um sistema implementado em FPGA. O sistema consegue, com um baixo custo e consumo de hardware, ser uma solução para o problema apontado. Menos de 10 % de uma FPGA de baixo custo foi usada para este trabalho, gerando 1023 neurônios artificiais disparando independentemente em tempo real para alimentar SNNs. Este sistema codifica sinais nos três maiores tipos de codificação neural: codificação em taxa de disparos, codificação temporal e codificação por populações. O usuário pode controlar em tempo real a taxa de disparos e o tipo de codificação que o sistema está gerando através da gravação de dados em posições específicas da memória do módulo. Possuindo ainda arquitetura modular e por ter sido descrito em HDL, vários módulos idênticos podem ser implementados num mesmo dispositivo lógico programável, podendo gerar o número de neurônios sensores desejado para uma dada aplicação.

Consonante a esse sistema foram estudadas e propostas topologias de amostradores de trens de *spikes* para SNN, utilizando a abordagem NAC. Foi investigado como uma SNN pode realizar operações de amostragem de informações em trens de *spikes* que entram numa rede. Tanto foi abordado a mostragem de *spikes* e circuitos de tomada de decisão para uma população de neurônios (codificação por populações). Quanto foram simuladas portas cujo sinal de controle é controlado por *bursts*, gerando amostradores de neurônios codificados em codificação temporal. Colhidas as amostras, foi demonstrado como é possível, dentro dos conceitos da NAC, utilizar esses dados para tomada de decisão.

Foi criado um circuito comparador de magnitude para populações de neurônios, podendo responder entre populações de neurônios qual representa uma grandeza maior ou menor, através dos números de neurônios ativos em um determinado instante. Também foi criado um circuito que faz uma tomada de decisão aliada a amostragem de um padrão de entrada configurados na codificação *Time-to-first-spike*. Este subsistema pode realizar a tomada de decisão sobre qual entrada em um

vetor tem maior excitação. Estes circuitos de tomada de decisão são estágios iniciais para sistemas de tomada de decisão mais complexos.

6.2.1 Publicações Relacionadas e Trabalhos Futuros

As seguintes publicações são decorrentes de investigações que aconteceram durante este trabalho de mestrado:

- OLIVEIRA-NETO, J. R., BEFORT, F. D., CAVALCANTI-NETO, R., and RAHEL, J.. Magnitude Comparison in Analog Spiking Neural Assemblies, **2014 International Joint Conference on Neural Networks (IJCNN)**, Beijing, p. 3186, 2014.
- RAHEL, J., OLIVEIRA-NETO, J. R., and CAJUEIRO, J. P. C.. "Gerador, Conversor e Codificador de Potenciais de Ação Artificiais para Redes Neurais Pulsadas" **Patente: Privilégio de Inovação. Número do registro: BR102015004644, data de depósito: 02/03/2015, Instituição de registro: INPI - Instituto Nacional da Propriedade Industrial**, Brasil, 2015.
- OLIVEIRA-NETO, J. R., CAJUEIRO, J. P. C., and RAHEL, J., .Neural Encoding and Spike Generation for Spiking Neural Networks implemented in FPGA, **2015 International Conference on Electronics, Communications and Computers (CONIELECOMP)**, Cholula, v. 1, p. 55-61, 2015.

Trabalhos Futuros

Os próximos passos em direção à criação de uma máquina para computar SNN em tempo real que seguem a esse trabalho é a implementação de modelos de neurônios em hardware. Além de ser um passo mais próximo da máquina, é também possível utilizar os modelos de neurônios, junto ao sistema gerador de trens de *spikes* desse trabalho, para testar as topologias de amostradores propostas funcionando no hardware em tempo real. Esses testes corroborariam com os testes apresentados neste trabalho.

O *output circuit* da Figura 1.1, sistema simétrico ao *input circuit* abordado neste trabalho, também é um desafio a ser superado. Esse sistema deve ser capaz de converter trens de *spikes* em sinais que alimentem os atuadores em tempo real para o controle do agente.

Ainda em relação ao sistema de entrada da rede, é interessante a investigação de uma solução híbrida entre a alternativa digital apresentada e a solução analógicas para o neurônio sensor [PUMARICA et al., 2007], [PUMARICA and DEL-MORAL-HERNÁNDEZ, 2007], visando ser possível aliar as vantagens das duas abordagens.

Paralelo a construção da máquina de computação de SNNs, as topologias de amostradores abrem perspectivas para investigações sobre novos tipos de códigos possíveis de serem processados dentro das SNNs e dentro da abordagem de NAC. Os próximos passos apontam para investigações sobre *rank order coding*, *Phase coding*, entre outros.

REFERÊNCIAS

[ABELES, 2009] ABELES, M. (2009). Synfire chains. *Scholarpedia*, 4(7):1441.

[ADRIAN and ZOTTERMAN, 1926] ADRIAN, E. D. and ZOTTERMAN, Y. (1926). The impulses produced by sensory nerve endings: Part ii: The response of a single end organ. *J Physiol*, (61(2)):151–171.

[ALTERA, 2014] ALTERA (2014). *Cyclone IV Device Handbook*. Altera Corporation, 101 Innovation Drive, San Jose, CA 95134.

[ANALOG, 2014] ANALOG (2014). *AD7960 - 18-Bit, 5 MSPS PulSAR Differential ADC*. Analog Devices, Norwood, MA 02062-9106, U.S.A.

[ATMEL, 2013] ATMEL (2013). *Atmel AT42QT1011 Single-key QTouch Touch Sensor IC*. Atmel Corporation. Rev.: 9542I–AT42–05/2013.

[BEAR et al., 2007] BEAR, M. F., CONNORS, B. W., and PARADISO, M. A. (2007). *Neuroscience - Exploring the Brain*. Lippincott Williams & Wilkins, 3rd edition.

[BENJAMIN et al., 2014] BENJAMIN, B., GAO, P., MCQUINN, E., CHOUDHARY, S., CHANDRASEKARAN, A., BUSSAT, J.-M., ALVAREZ-ICAZA, R., ARTHUR, J., MEROLLA, P., and BOAHEN, K. (2014). Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations. *Proceedings of the IEEE*, 102(5):699–716.

[BIENENSTOCK, 1995] BIENENSTOCK, E. (1995). A model of neocortex. *Network: Computation in Neural Systems*, 6(1):179–224.

[BRAGA et al., 2009] BRAGA, A. D. P., CARVALHO, A. P. D. L. F. D., and LUDERMIR, T. B. (2009). *Redes Neurais Artificiais. Teoria E Aplicações (Em Portuguese do Brasil)*. LTC.

[BRETTE, 2012] BRETTE, R. (2012). Computing with neural synchrony. *PLoS Comput Biol*, 8(6):e1002561.

[BROWN et al., 2004] BROWN, E. N., KASS, R. E., and MITRA, P. P. (2004). Multiple neural spike train data analysis: state-of-the-art and future challenges. *Nature Neuroscience*, 7(5):461.

[BURACAS and ALBRIGHT, 2014] BURACAS, G. T. and ALBRIGHT, T. D. (2014). Gauging sensory representations in the brain. *Trends in Neurosciences*, 22(7):302 – 309.

[BUZSÁKI, 2010] BUZSÁKI, G. (2010). Neural syntax: Cell assemblies, synapsembles, and readers. *Neuron*, 68(3):362–385.

[BUZSÁKI and DRAGUHN, 2004] BUZSÁKI, G. and DRAGUHN, A. (2004). Neuronal oscillations in cortical networks. *Science*, 304(5679):1926–1929.

[CASSIDY et al., 2007] CASSIDY, A., DENHAM, S., KANOLD, P., and ANDREOU, A. (2007). Fpga based silicon spiking neural array. In *Biomedical Circuits and Systems Conference, 2007. BIOCAS 2007. IEEE*, pages 75–78.

[CHEUNG et al., 2012] CHEUNG, K., SCHULTZ, S., and LUK, W. (2012). *A Large-Scale Spiking Neural Network Accelerator for FPGA Systems*, volume 7552 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg.

[CUI, 2013] CUI (2013). *CMA-4544PF-W - Electret Condenser Microphone*. CUI INC.

[DAYAN and ABBOTT, 2001] DAYAN, P. and ABBOTT, L. F. (2001). *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. Massachusetts Institute of Technology Press.

[DIGIKEY, 2015] DIGIKEY (2015). Digikey product-searsh. accessed on ago/2015.

[ELIT, 2015] ELIT (2015). Elit brand electrochemical sensors and computer-based instrumentation. accessed on jun/2015.

[EPCOS, 2014] EPCOS (2014). *PTC thermistors as limit temperature sensors*. EPCOS TDK.

[EVERLIGHT, 2012] EVERLIGHT (2012). *ALS-PT19-315C/L177/TR8 - Ambient Light Sensor Surface - Mount*. Everlight. Rev. 5.

[FARQUHAR and HASLER, 2004] FARQUHAR, E. and HASLER, P. (2004). A bio-physically inspired silicon neuron. In *Circuits and Systems, 2004. ISCAS '04. Proceedings of the 2004 International Symposium on*, volume 1, pages I–309–I–312 Vol.1.

[FERNANDES et al., 2001] FERNANDES, J. C. B., KUBOTA, L. T., and NETO, G. D. O. (2001). Eletrodos Íon-seletivos: Histórico, mecanismo de resposta, seletividade e revisão dos conceitos. *Química Nova*, 24(1):120–130. ISSN 1678-7064.

[FREESCALE, 2008] FREESCALE (2008). *S12 Product & Module Overview*. Freescale Semiconductor, Inc.

[FREESCALE, 2012] FREESCALE (2012). *Kinetis MCUs: The Big Picture*. Freescale Semiconductor, Inc.

[GERSTNER and KISTLER, 2002] GERSTNER, W. and KISTLER, W. M. (2002). *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press.

[GERSTNER et al., 1997] GERSTNER, W., KREITER, A. K., MARKRAM, H., and HERZ, A. V. M. (1997). Neural codes: Firing rates and beyond. *Proceedings of the National Academy of Sciences*, 94(24):12740–12741.

[GHOSH-DASTIDAR and ADELI, 2009] GHOSH-DASTIDAR, S. and ADELI, H. (2009). Spiking neural networks. *International Journal of Neural Systems*, 19(04):295–308.

[HANWEI, 2015] HANWEI (2015). *MQ-3 Gas Sensor*. HANWEI Eletronics CO., LTD.

[HEBB, 2002] HEBB, D. O. (2002). *The Organization of Behavior: A Neuropsychological Theory*. Lawrence Erlbaum Associated, Inc, Mahwah, NJ. Originally published: New York: Wiley, 1949.

[HERCULANO-HOUZEL, 2009] HERCULANO-HOUZEL, S. (2009). The human brain in numbers: A linearly scaled-up primate brain. *Frontiers in Human Neuroscience*.

[HONEYWELL, 2009] HONEYWELL (2009). *Infrared Components - Ceramic Discrete Surface Mount Emitters and Detectors*. Honeywell Inc.

[HONEYWELL, 2011] HONEYWELL (2011). *GMS-10 RVS Series - Oxygen Sensors*. Honeywell Inc.

[HONEYWELL, 2012] HONEYWELL (2012). *SD5491 - Silicon Phototransistor*. Honeywell Inc.

[HOPFIELD, 1982] HOPFIELD, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proc. Nat. Acad. Sci.*, 79:2554–2558.

[HYNNA and BOAHEN, 2006] HYNNA, K. and BOAHEN, K. (2006). Neuronal ion-channel dynamics in silicon. In *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, pages 4 pp.–.

[IBM, 2014] IBM (2014). Brain power - scientists at ibm research unveil a brain-inspired computer and ecosystem. http://researchweb.watson.ibm.com/cognitive-computing/brainpower/. accessed on dec/2014.

[IEEE, 2006] IEEE (2006). Ieee standard for verilog hardware description language. *IEEE Std 1364-2005 (Revision of IEEE Std 1364-2001)*, pages 1–560.

[IZHIKEVICH, 2003] IZHIKEVICH, E. M. (2003). Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, 14(6):1569–1572.

[IZHIKEVICH, 2004] IZHIKEVICH, E. M. (2004). Which model to use for cortical spiking neurons? *Neural Networks, IEEE Transactions on*, 15(5):1063–1070.

[IZHIKEVICH, 2006] IZHIKEVICH, E. M. (2006). Polychronization: computation with spikes. *Neural Computation*, 18(2):245–282.

[JAEGER, 2001] JAEGER, H. (2001). The ‘‘echo state’’ approach to analysing and training recurrent neural networks - with an erratum note. *German National Research Center for Information Technology*.

[KAMEDA and YAGI, 2006] KAMEDA, S. and YAGI, T. (2006). An analog silicon retina with multichip configuration. *Neural Networks, IEEE Transactions on*, 17(1):197–210.

[KANDEL et al., 2000] KANDEL, E. R., SCHWARTZ, J. H., and JESSEL, T. M. (2000). *Principles of Neural Science*. McGrall-Hill Health Prof. Division, New York, NY, 4th ed edition.

[KNOWLES, 2005a] KNOWLES (2005a). *BJ-21590-000 - Microphone Performance Specification*. Knowles Electronics. Itasca, Illinois U.S.A.

[KNOWLES, 2005b] KNOWLES (2005b). *BL Series Microphones*. Knowles Acoustics.

[KOHONEN, 1988] KOHONEN, T. (1988). Neurocomputing: Foundations of research. chapter Self-organized Formation of Topologically Correct Feature Maps, pages 509–521. MIT Press, Cambridge, MA, USA.

[KOICKAL et al., 2006] KOICKAL, T., HAMILTON, A., PEARCE, T., TAN, S., COVINGTON, J., and GARDNER, J. (2006). Analog vlsi design of an adaptive neuromorphic chip for olfactory systems. In *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, pages 4 pp.–4550.

[KÖNIG et al., 1996] KÖNIG, P., ENGEL, A. K., and SINGER, W. (1996). Integrator or coincidence detector? the role of the cortical neuron revisited. *Trends in Neurosciences*, 19(4):130–137.

[KOSTAL et al., 2007] KOSTAL, L., LANSKY, P., and ROSPARS, J.-P. (2007). Review article: Neuronal coding and spiking randomness. *European Journal of Neuroscience*, 26(10):2693–2701.

[KUMAR et al., 2010] KUMAR, A., ROTTER, S., and AERTSEN, A. (2010). Spiking activity propagation in neuronal networks: reconciling different perspectives on neural coding. *Nat Rev Neurosci*, 11:615–627.

[LEENS, 2009] LEENS, F. (2009). An introduction to i2c and spi protocols. *Instrumentation Measurement Magazine, IEEE*, 12(1):8–13.

[LEVINE, 2007] LEVINE, D. N. (2007). Sherringtons the integrative action of the nervous system: Acentennial appraisal. *Journal of the Neurological Sciences*, 1:1–6.

[LINEAR, 2005] LINEAR (2005). *LTC2485 - 24-Bit ADC with Easy Drive Input Current Cancellation and I2C Interface*. Linear Thechnology Corporation, McCarthy Blvd., Milpitas, CA.

[LINSKER, 1988] LINSKER, R. (1988). Self-organization in a perceptual network. *Computer*, 21(3):105–117.

[MAASS, 1997] MAASS, W. (1997). Networks of spiking neurons: the third generation of neural network models. *IEEE Trans. Neural Networks*, 10(9):1659–1671.

[MAASS et al., 2002] MAASS, W., NATSCHLÄGER, T., and MARKRAM, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–2560.

[MATHWORKS, 2015] MATHWORKS (2015). Matlab: The language of technical computing. accessed on ago/2015.

[MCCULLOCH and PITTS, 1943] MCCULLOCH, W. S. and PITTS, W. (1943). A logical calculus of the ideas immanent in nervous activity. *ulletin of Mathematical Biophysics*, 5:115–133.

[MENDEL and MCLAREN, 1970] MENDEL, M. and MCLAREN, R. W. (1970). *Adaptive, learning, and pattern recognition systems; theory and applications, Volume 66 (Mathematics in Science and Engineering)*. Academic Press.

[MICROCHIP, 2012] MICROCHIP (2012). *PIC32 Microcontroller Families*. Microchip Technology Inc.

[MICROCHIP, 2013] MICROCHIP (2013). *16-bit Embedded Control Solutions*. Microchip Technology Inc.

[MINSKY and PAPERT, 1969] MINSKY, M. L. and PAPERT, S. A. (1969). *Perceptrons*. MIT Press, Cambridge, Massachussets.

[MURATA, 2014] MURATA (2014). *NTC Termitors*. muRata - Innovator in Eletectronics.

[NEFTCI et al., 2013] NEFTCI, E., BINAS, J., RUTISHAUSER, U., CHICCA, E., INDIVERI, G., and DOUGLAS, R. J. (2013). Synthesizing cognition in neuromorphic electronic systems. *Proceedings of the National Academy of Sciences*, 110(37):E3468–E3476.

[NXP, 2011a] NXP (2011a). *Cortex-M0 MCUs with lowest active power superior code density*. NXP Semiconductors.

[NXP, 2011b] NXP (2011b). *First asymmetrical, dual-core digital signal controller featuring Cortex-M4 & Cortex-M0*. NXP Semiconductors.

[OLIVEIRA-NETO et al., 2014] OLIVEIRA-NETO, J. R., BELFORT, F. D., CAVALCANTI-NETO, R., and RANHEL, J. (2014). Magnitude comparison in analog spiking neural assemblies. In *Neural Networks (IJCNN), 2014 International Joint Conference on*, pages 3186–3191.

[OLIVEIRA NETO et al., 2015] OLIVEIRA NETO, J. R., CAJUEIRO, J. P. C., and RANHEL, J. (2015). Neural encoding and spike generation for spiking neural networks implemented in fpga. In *Electronics, Communications and Computers (CONIELECOMP), 2015 International Conference on*, pages 55–61.

[OMNIVISION, 2011] OMNIVISION (2011). *OV7675 - VGA Product brief*. OmniVision Technologies, Burton Drive, Santa Clara, CA.

[OMRON, 2011] OMRON (2011). *Temperature Sensor E52*. OMRON Corporation - Industrial Automation Company.

[ON, 2013] ON (2013). *LC717A00AR - Capacitance-Digital-Converter LSI for Electrostatic Capacitive Touch Sensors*. ON Semiconductor.

[OPPENHEIM et al., 1999] OPPENHEIM, A. V., SCHAFER, R. W., and BUCk, J. R. (1999). *Discrete-time Signal Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2 edition.

[PANASONIC, 2005] PANASONIC (2005). *GP-KX121 Series - Color Board Cameras*. Panasonic - Matsushita Electric.

[PAPROCKI et al., 2013] PAPROCKI, B., SZCZEPANSKI, J., and KOLBUK, D. (2013). Information transmission efficiency in neuronal communication systems. *BMC Neuroscience*, 14(Suppl 1):P217.

[PAUGAM-MOISY and BOHTE, 2010] PAUGAM-MOISY, H. and BOHTE, S. (2010). Computing with spiking neuron networks. In Rozenberg, G., Bäck, T., and Kok, J. N., editors, *Handbook of Natural Computing. 1st Edition*, volume 1, pages 1–47. Springer-Verlag, Heidelberg, Germany.

[PERRETT et al., 1982] PERRETT, D. I., ROLLS, E. T., and CAAN, W. (1982). Visual neurones responsive to faces in the monkey temporal cortex. *Experimental Brain Research*, 47(3):329–342.

[PLASSCHE, 2003] PLASSCHE, R. V. D. (2003). *CMOS Integrated Analog-to-Digital and Digital-to-Analog Converters*. 742. Kluwer Academic Publishers, 2 edition.

[POOLE et al., 1998] POOLE, D., MACKWORTH, A., and GOEBEL, R. (1998). *Computational intelligence: A logical approach*. Oxford University Press.

[POON and ZHOU, 2011] POON, C.-S. and ZHOU, K. (2011). Neuromorphic silicon neurons and large-scale neural networks: challenges and opportunities. *Frontiers in Neuroscience*, 5(108).

[PUMARICA and DEL-MORAL-HERNÁNDEZ, 2007] PUMARICA, J. S. and DEL-MORAL-HERNÁNDEZ, E. (2007). Codificador cmos orientado al reconocimiento de patrones con independencia de escala. In S.R.L., E. H., editor, *In: XIII Workshop Ierchip, 2007, Lima. XIII WORKSHOP IBERCHIP*, volume 8.

[PUMARICA et al., 2007] PUMARICA, J. S., HERNANDEZ, E. D. M., and CÁRDENAS, C. S. (2007). CMOS encoder for scale-independent pattern recognition. In Petraglia, A., Pedroni, V. A., and Cauwenberghs, G., editors, *Proceedings of the 20th Annual Symposium on Integrated Circuits and Systems Design, SBCCI 2007, Copacabana, Rio de Janeiro, Brazil, September 3-6, 2007*, pages 241–244. ACM.

[QUALCOMM, 2014] QUALCOMM (2014). Introducing qualcomm zeroth processors: Brain-inspired computing. accessed on dec/2014.

[RANHEL, 2012a] RANHEL, J. (2012a). Computação em assembleias neurais pulsadas. *CBEB2012 - XXIII Congresso Brasileiro em Engenharia Biomédica*, 1:1697–1701.

[RANHEL, 2012b] RANHEL, J. (2012b). *Computação por Assembleias Neurais em Redes Neurais Pulsadas*. PhD thesis, Escola Politécnica da Universidade de São Paulo, São Paulo.

[RANHEL, 2012c] RANHEL, J. (2012c). Neural assembly computing. *IEEE Trans. Neural Networks*, 23(6):916–927.

[RANHEL, 2013] RANHEL, J. A. (2013). Neural assemblies and finite state automata. In *Proceedings of 1st BRICS Countries Congress (BRICS-CCI) and 11th Brazilian Congress (CBIC) on Computational Intelligence*, volume 1, pages 1–6.

[RICE et al., 2009] RICE, K. L., BHUIYAN, M. A., TAHA, T. M., VUTSINAS, C. N., and SMITH, M. C. (2009). Fpga implementation of izhikevich spiking neural networks for character recognition. In *Reconfigurable Computing and FPGAs, 2009. ReConFig '09. International Conference on*, pages 451–456.

[ROJAS, 1996] ROJAS, R. (1996). *Neural Networks: a Systematic Introduction*. Springer, Berlin.

[ROLLS et al., 2006] ROLLS, E. T., FRANCO, L., AGGELOPOULOS, N. C., and JEREZ, J. M. (2006). Information in the first spike, the order of spikes, and the number of spikes provided by neurons in the inferior temporal visual cortex. *Vision Research*, 46(25):4193 – 4205.

[ROLLS and TOVEE, 1994] ROLLS, E. T. and TOVEE, M. J. (1994). Processing speed in the cerebral cortex and the neurophysiology of visual masking. *Proceedings of the Royal Society of London B: Biological Sciences*, 257(1348):9–15.

[ROSS, 2010] ROSS, T. J. (2010). *Fuzzy Logic with Engineering Applications*. John Wiley and Sons, University of New Mexico, USA, 3rd edition.

[RUMELHART et al., 1988] RUMELHART, D. E., HINTON, G. E., and WILLIAMS, R. J. (1988). Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, Cambridge, MA, USA.

[RUSSELL and NORVIG, 2009] RUSSELL, S. and NORVIG, P. (2009). *Artificial Intelligence: A Modern Approach (3rd Edition)*. Pearson.

[SEMICONDUCTOR, 2013] SEMICONDUCTOR (2013). *NOIV2SN1300A - VITA 1300 1.3 Megapixel 150 FPS Global Shutter CMOS Image Sensor*. Semiconductor Components Industries. Rev. 9.

[SHAYANI et al., 2008] SHAYANI, H., BENTLEY, P. J., and TYRRELL, A. M. (2008). Hardware implementation of a bio-plausible neuron model for evolution and growth of spiking neural networks on fpga. In *Adaptive Hardware and Systems, 2008. AHS '08. NASA/ESA Conference on*, pages 236–243.

[SOUZA, 2013] SOUZA, F. M. C. D. (2013). *Probabilidade, Estatística e Processos Estocásticos*, volume 3. Fernando Menezes Campello de Souza, Recife, PE, 22 edition.

[SPECTRUM, 2009] SPECTRUM (2009). *SS Style High Temp Sensor*. Spectrum Sensors & Products Operation INC.

[TEXAS, 2011] TEXAS (2011). *DDC264 - 64-Channel, Current-Input Analog-to-Digital Converter*. Texas Instruments, Dallas, Texas U.S.A.

[TEXAS, 2013a] TEXAS (2013a). *C2000 Real-Time Microcontrollers*. Texas Instruments, Dallas, Texas U.S.A.

[TEXAS, 2013b] TEXAS (2013b). *MSP430 - Ultra-Low-Power Microcontrollers*. Texas Instruments, Dallas, Texas U.S.A.

[TEXAS, 2013c] TEXAS (2013c). *Tiva C Series ARM Microcontrollers*. Texas Instruments, Dallas, Texas U.S.A.

[TEXAS, 2015] TEXAS (2015). *LM35 Precision Centigrade Temperature Sensors*. Texas Instruments, Dallas, Texas U.S.A.

[THORPE et al., 2001] THORPE, S., DELORME, A., and VANRULLEN, R. (2001). Spike-based strategies for rapid processing. *Neural Networks*, 14(6-7):715–725.

[THORPE et al., 1996] THORPE, S., FIZE, D., and MARLOT, C. (1996). Speed of processing in the human visual syste. *Nature*, (381(6582)):520–522.

[VALENÇA, 2010] VALENÇA, M. J. S. (2010). *Fundamentos das Redes Neurais: exemplos em Java*. Livro Rápido - Elógica, Olinda, PE, Brazil, 2nd edition.

[VANRULLEN et al., 2005] VANRULLEN, R., GUYONNEAU, R., and THORPE, S. J. (2005). Spike times make sense. *Trends in Neurosciences*, 28(1):1–4.

[VIEVILLE and CRAHAY, 2004] VIEVILLE, T. and CRAHAY, S. (2004). Using an Hebbian Learning Rule for Multi-Class SVM Classifiers. *Journal of Computational Neuroscience*, 17(3):271+.

[VISHAY, 2008] VISHAY (2008). *265 V PTC Thermistors For Overload Protection*. Vishay BCcomponents. Document Number: 2908, Revision: 18-Aug-09.

[VISHAY, 2010] VISHAY (2010). *General Purpose Strain Gages - Linear Pattern*. Micro-Measurements - Vishay Precision Group. Document Number: 11312, Revision: 28-Jan-10.

[WANG et al., 2014] WANG, R., HAMILTON, T., TAPSON, J., and VAN SCHAIK, A. (2014). An fpga design framework for large-scale spiking neural networks. In *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on*, pages 457–460.

[WANG et al., 2013] WANG, R. M., COHEN, G., STIEFEL, K. M., HAMILTON, T. J., TAPSON, J. C., and VAN SCHAIK, A. (2013). An fpga implementation of a polychronous spiking neural network with delay adaptation. *Frontiers in Neuroscience*, 7(14).

[WEN and BOAHEN, 2003] WEN, B. and BOAHEN, K. (2003). A linear cochlear model with active bi-directional coupling. In *Engineering in Medicine and Biology Society, 2003. Proceedings of the 25th Annual International Conference of the IEEE*, volume 3, pages 2013–2016 Vol.3.

[WERBOS, 1974] WERBOS, P. (1974). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, Cambridge, MA.

[WIDROW and HOFF, 1960] WIDROW, B. and HOFF, M. E. (1960). *Adaptive Switching Circuits*.

[WU et al., 2002] WU, S., AMARI, S., and NAKAHARA, H. (2002). Population coding and decoding in a neural field: a computational study. *Neural Computation*, 14(5):999–1026.

[XING et al., 2012] XING, J., BERGER, T., and SEJNOWSKI, T. J. (2012). A berger-levy energy efficient neuron model with unequal synaptic weights. In *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on*, pages 2964–2968.

[ZUMBAHLEN, 2008] ZUMBAHLEN, H., editor (2008). *Basic Linear Design*. Analog Devices.

APÊNDICE A

AQUIVOS VERILOG DO PROJETO CONVERSOR/GERADOR NEURAL

Em razão do grande numero de arquivos de descrição de hardware utilizados no projeto do GCCst e o tamanho de cada arquivo, apenas a instância principal sera reproduzida neste apêndice:

```
//-----------------------------------------------------------------------
// IN_SPIKES:
// Descricao: Implementacao de um hardware que transforma sinais de sensores
//            (analogico ou digital) em Spikes;
//
// Autor:  Jose Rodrigues de Oliveira Neto
// E-mail: neto.rodrigues.14@gmail.com
//-----------------------------------------------------------------------
// Arquivo:   in_spikes.v;
// Descricao: Instancia top que chama os demais blocos do sistema;
// Versao:    0.1 - Versao inicial:
//            0.2 - Modificacoes para Rodar na Placa Chinesa EP clock de
//                  25MHz;
//            0.3 - Modificacoes para poder configurar as frequencias de
//                  trabalho da maquina principal F_m =  500/2^n, onde:
//                  F_m = frequencia de trabalho maquina de estado principal;
//                  n = numero escrito na ultima posicao da MEM_SPIKES;
//
//-----------------------------------------------------------------------
module in_spikes(
```

```
   input           clk_25MHz,
input           reset,

//CIRCUITO DE RECEPCAO:
input           clk_in,
input           stb_in,
input           data_in,

//sinais SPI Saida
input           spi_en,
output          spi_sclk_o,
output          spi_mosi_o,
input           spi_miso_i,

//SAIDAS TESTE
output wire [ 7:0] LEDS,
output             clk_500Hz_t,
output      [ 7:0] data_out_0,
output      [ 7:0] data_out_1
);

//-------SINAIS E REGISTRADORES-----------------------//

//MEM_IN (RAM DUAL PORT):
wire [ 9:0] mem_i_wr_add;
wire [ 9:0] mem_i_rd_add;
wire [ 7:0] mem_i_wr_data;
wire        mem_i_wr_en;
wire [ 7:0] mem_i_rd_data;

//MEM_COMP (RAM DUAL PORT):
wire [ 9:0] mem_c_wr_add;
wire [ 9:0] mem_c_rd_add;
wire [ 7:0] mem_c_wr_data;
wire        mem_c_wr_en;
wire [ 7:0] mem_c_rd_data;

//MEM_LIMIAR (RAM DUAL PORT):
wire [ 9:0] mem_l_wr_add;
wire [ 9:0] mem_l_rd_add;
wire [ 7:0] mem_l_wr_data;
wire        mem_l_wr_en;
wire [ 7:0] mem_l_rd_data;

//MEM_SPIKES (RAM DUAL PORT):
wire        mem_s_wr_en;
```

```
wire [ 9:0] mem_s_wr_add;
wire        mem_s_wr_data;
wire [ 9:0] mem_s_rd_add;
wire        mem_s_rd_data;

wire [ 6:0] mem_add_b;
wire [ 7:0] mem_data_b;

//MAQUINA_AMC
wire [ 9:0] mem_add_amc;
wire        mem_s_wr_data_amc;

//MAQUINA_CM
wire [ 9:0] mem_add_cm;
wire        mem_s_wr_data_cm;
wire        mem_s_wr_en_cm;

//CLOCKS
wire        clk_100MHz;
wire        clk_8kHz; //Vai passar pelo 'gera_clocks' para gerar
wire        clk_500Hz;//500Hz
wire        clk_config;
wire        en_clk_config;

//MAQUINA GERAL
wire        incremento_mem_c;
wire        reset_mem_c;
wire [ 2:0] estado_atual;
wire        comparador_en;
wire        incremento_mem_c_en;
wire        reset_mem_c_en;
wire        comparador_c;
wire        sincronismo;

//CONTROLADOR SAIDA E SPI
wire        enable;
wire        start;
wire        idle;
wire        last_edge;
wire [ 6:0] mem_s_add;
reg         controlador_saida_en;
wire        reset_spi;

// saidas de teste
wire        reset_n;
wire [ 2:0] estado_atual_cs;
```

```
wire [ 2:0] estado_atual_cm;
wire [ 1:0] estado_atual_cr;
reg  [ 7:0] data_out_0_reg;
reg  [ 7:0] data_out_1_reg;

//----------------MODULOS-----------------------------//

//MAQUINA DE ESTADOS GERAL:
in_spikes_maquina in_spikes_maquina_Inst(
   .clk                 (clk_100MHz),
.reset               (reset_n),
//Entradas da lista de sensibilidade da maquina de estados:
//.clk_500Hz           (clk_500Hz),
.clk_config          (clk_config),
.mem_c_rd_add        (mem_add_cm),
.comparador_c        (comparador_c),
.sincronismo         (sincronismo),
.incremento_mem_c    (incremento_mem_c),   // Quando '1' indica que o
                                           // incrmento de MEM_COMP foi
                                           // concluido;
.reset_mem_c         (reset_mem_c),        // Quando '1' indica que a
                                           // MEM_COMP foi resetada;
//Saidas de controle:
.comparador_en       (comparador_en),
.incremento_mem_c_en (incremento_mem_c_en),
.reset_mem_c_en      (reset_mem_c_en),
.estado_atual_out    (estado_atual)        // Sinal que copia o estado
                                           // atual para servir de sinal
                                           // de controle para os demais
                                           // modulos
);

//MAQUINA DE ESTADOS DO COMPARADOR:
comparador_maquina comparador_maquina_Inst(
   .clk                (clk_100MHz),
.reset              (reset_n),
//Entradas da lista de sensibilidade da maquina de estados:
.comparador_en      (comparador_en),
.mem_i_data         (mem_i_rd_data),
.mem_c_data         (mem_c_rd_data),
.mem_l_data         (mem_l_rd_data),
//Saidas de controle:
.mem_add            (mem_add_cm),
.comparador_c       (comparador_c),
.sincronismo        (sincronismo),
.mem_s_wr_data      (mem_s_wr_data_cm),
```

```
.mem_s_wr_en       (mem_s_wr_en_cm)
);

//MAQUINA DE ESTADOS QUE ATUALIZA A MEM_COMP:
atualiza_mem_c_maquina atualiza_mem_c_maquina_Inst(
   .clk                   (clk_100MHz),
.reset                 (reset_n),
//Entradas da lista de sensibilidade da maquina de estados:
.incremento_mem_c_en    (incremento_mem_c_en),
.reset_mem_c_en         (reset_mem_c_en),
.mem_i_data             (mem_i_rd_data),
.mem_c_data             (mem_c_rd_data),
//Saidas de controle:
.incremento_mem_c       (incremento_mem_c),
.reset_mem_c            (reset_mem_c),
.mem_add                (mem_add_amc),
.mem_s_wr_data          (mem_s_wr_data_amc),
.mem_c_wr_data          (mem_c_wr_data),
.mem_c_wr_en            (mem_c_wr_en)
);

//CIRCUITO DE RECEPCAO:
circuito_recepcao circuito_recepcao_Inst(
   .clk             (clk_100MHz),
   .reset           (reset_n),
.clk_in          (clk_in),
.stb_in          (stb_in),
.data_in         (data_in),

.en_mem_in       (mem_i_wr_en),
.en_mem_limiar   (mem_l_wr_en),
.en_clk_config   (en_clk_config),

.add_out         (mem_i_wr_add),
.data_out        (mem_i_wr_data)
);

//MEM_IN:
ram_2_port MEM_IN(
.data         (mem_i_wr_data),//(8'hF0),//
.rdaddress    (mem_i_rd_add),
.rdclock      (clk_100MHz),
.wraddress    (mem_i_wr_add),
.wrclock      (clk_100MHz),
.wren         (mem_i_wr_en),
.q            (mem_i_rd_data)
```

```
);

assign mem_i_rd_add =(comparador_en)? mem_add_cm: mem_add_amc;

//MEM_COMP:
ram_2_port MEM_COMP(
.data         (mem_c_wr_data),
.rdaddress    (mem_c_rd_add),
.rdclock      (clk_100MHz),
.wraddress    (mem_c_wr_add),
.wrclock      (clk_100MHz),
.wren         (mem_c_wr_en),
.q            (mem_c_rd_data)
);

assign mem_c_wr_add = mem_add_amc;
assign mem_c_rd_add =(comparador_en)? mem_add_cm: mem_add_amc;

//MEM_LIMIAR:
ram_2_port MEM_LIMIAR(
.data         (mem_l_wr_data),
.rdaddress    (mem_l_rd_add),
.rdclock      (clk_100MHz),
.wraddress    (mem_l_wr_add),
.wrclock      (clk_100MHz),
.wren         (mem_l_wr_en),
.q            (mem_l_rd_data)
);

assign mem_l_wr_data = mem_i_wr_data;
assign mem_l_wr_add = mem_i_wr_add;
assign mem_l_rd_add =(comparador_en)? mem_add_cm: mem_add_amc;

//RAM QUE GUARDA OS SPIKES
ram_spikes MEM_SPIKES(
.address_a    (mem_s_rd_add),
.address_b    (mem_add_b),
.data_a       (mem_s_wr_data),
.data_b       (),
.inclock      (clk_100MHz),
.outclock     (clk_100MHz),
.wren_a       (mem_s_wr_en),
.wren_b       (),
.q_a          (mem_s_rd_data),
.q_b          (mem_data_b)
);
```

```
assign mem_s_wr_en =  (comparador_en)? mem_s_wr_en_cm: mem_c_wr_en;
assign mem_s_wr_data = (comparador_en)? mem_s_wr_data_cm : mem_s_wr_data_amc;
assign mem_s_rd_add = (comparador_en)? mem_add_cm: mem_add_amc;
assign mem_add_b =  mem_s_add; //7'h00;

always @(posedge reset_n, posedge clk_100MHz)
begin
  if(reset_n)
    data_out_0_reg <= 8'h00;
  else
    data_out_0_reg <= mem_data_b;
end

assign data_out_0 = data_out_0_reg;

always @(posedge reset_n, posedge mem_i_wr_en)
begin
  if(reset_n)
    data_out_1_reg <= 8'h00;
  else
    data_out_1_reg <= mem_i_wr_data;
end

assign data_out_1 =  data_out_1_reg;

controlador_saida controlador_saida_Inst(
    .reset                  (reset_n),
    .clk                    (clk_100MHz),       // clk do sistema
    //Entrada e saida paralela de dados
 .enable                 (enable),
    .start                  (start),
 .reset_spi              (reset_spi),
 .mem_s_rd_add           (mem_s_add),
 .estado_atual           (estado_atual_cs),
    .controlador_saida_en   (controlador_saida_en),
    .idle                   (idle),
    .last_edge              (last_edge)
);

always @(posedge reset_n or posedge spi_en or negedge clk_25MHz)
begin
  if(reset_n)
    controlador_saida_en <= 1'b0;
  else if( ~clk_25MHz)
    controlador_saida_en <= 1'b0;
```

```
  else if(spi_en)
    controlador_saida_en <= 1'b1;
  else
    controlador_saida_en <= 1'b0;
end


// SPI - SAIDA DOS SPIKES
SPI_Register SPI_SAIDA_Inst(
    .reset         (reset_spi),
    .clk           (clk_100MHz),
    .enable        (enable),
    //Entrada e saiÂda paralela de dados
    .start         (start),
    .data_in       (mem_data_b),
    .data_out      (),
    .idle          (idle),
    .last_edge     (last_edge),
    //sinais SPI
    .spi_sclk_o    (spi_sclk_o),
    .spi_mosi_o    (spi_mosi_o),
    .spi_miso_i    (spi_miso_i)
);


//PLL QUE GERA OS CLOCKS:
clocks clocks_Inst(
.areset       (reset_n),
.inclk0       (clk_25MHz),
.c0           (clk_100MHz),
.c1           (clk_8kHz),
.locked       ()
);

//assign clk_100MHz = clk_25MHz;
//GERA 500Hz de 8kHz
gera_clocks gera_clocks_Inst(
    .clk_8kHz      (clk_8kHz),
    .reset         (reset_n),
.en_clk_config (en_clk_config) ,
.data_in       (mem_i_wr_data[4:0]),
.clk_500Hz     (clk_500Hz),
.clk_config_o  (clk_config)
);

//testes
```

```
assign reset_n = (reset)? 1'b0: 1'b1;
assign clk_500Hz_t = clk_500Hz;
assign LEDS = data_out_1_reg;

endmodule
```

APÊNDICE B

AQUIVOS MATLAB DAS TOPOLOGIAS DE AMOSTRADORES

As simulações das topologias de redes descritas neste trabalho utilizam como arquivo base o *script* Matlab publicado em [RANHEL, 2012b]. Esse arquivo base foi modificado para implementar as topologias de amostradores e circuitos de tomada de decisão. Por conter a maior parte das mudanças necessárias durante as simulações, será transcrito neste apêndice o arquivo de simulação da topologia do amostrador de sinais em codificação temporal utilizando o modelo de neurônio LI&F:

```
% NAC Advanced, Analog & Decision : J.Ranhel - Rev: 2014_01
% MATLAB code for dmeonstrating Neural Assembly Computing (NAC) basics,
% cells sharing time, neurons participating on multiple assemblies,
% as well as % for discussing the Analog x Digital operation.
% This version can construct ANALOG GATES - the topology is different from
% the previous versions.
% The assemblies K0, A1, A2, A3, A4, A5, A6, and A7 are RESERVED system assemblies:
%   K0 - triggers all the other processes - occurs only once.
%   A1 - force inputs, sync to A5, controlled by the input string "InStr1"
%   A2 - force inputs, sync to A5, controlled by the input string "InStr2"
%   A3 + A4 - reserved for simulating analog input 1
%   A5 + A6 - reserved for simulating analog input 2
%   A7 - first assembly in the system pacemaker
%
% A Neural Assembly (NA) is a set of 'nNa' neuroids.
% The total number of assemblies must be set in 'numAssb'.
% The total number of neuroids is calculated as follows: nN=nNa*numAssb.
%
% All neuroids are interconnected by: (1) axonal propagation delays matrix D[],
% and (2) by synaptic weights matrix W[]. The program fills in these matrices
% using data from T[], which contains vectors for creating assemblies.
% Therefore, the network topology is described in T{} and the connections
% and Propagation delays are defined into the matrices W[] and PD{} respectively.
%
% User can DESIGN a topology by defining the NA relations as follows:
% T=[-i1 i1 -j1 j1 s1 d1 ct1 ip1; -12 i2 -j2 j2 s2 d2 ct2 ip2; ... -in in -jn jn sn dn ctn ipn];
% where: -i_ = lower part (x.nNa+1 <= neuron < x.nNa+nNa/2) of pre-synaptic assembly;
%         i_ = higher part (x.nNa+nNa/2 <= neuron < (x+1)*nNa) of pre-synaptic assembly,
%        -j_ = lower part (x.nNa+1 <= neuron < x.nNa+nNa/2) of post-synaptic assembly;
```

```
%         j_ = higher part (x.nNa+nNa/2 <= neuron < (x+1)*nNa) of post-synaptic assembly,
% (consider 'x' is the assembly number Kx)
%        s_ = synaptic weight among i_ -> j_ neuroids,
%        d_ = axonal propagation delay from i_ -> j_,
%        ct_ connections type: 0=Normal 1=Analog Gate.
%        ip_ denote the j_ neuroid type (used only for Izhikevich's model - not used here )

%% -- 1. SETTING PROGRAM VARIABLES -----
clear; clc;                % restart Matlab vars / clear the comand screen

% -- 1.CONFIG parameters -- %
% INPUT vectors with characters 1's and 0's to be applied to the FSM. In both cases,
% stimuli '1' inputs synchronously to NA A7 (when neuroids 7*nNa+1...8*nNa are firing)
% This vector controls when an INPUT '1' appears in the NA A1 (nNa+1...2*nNa)
% OBS: the lenght of these strings determine the simulation time:
%      Total simulation time (ms) ~= tINev + tBD * size(InStr1)
InStr1=[0 0 0 0 0 0 0 1];
% This vector controls when an INPUT '1' appears on NA A2 (2*nNa+1...3*nNa)
InStr2=[0 1 0 0 1 0 0 0];

tBD=40;                    % time Basis for DELAY between assemblies (i)->(j)(ms)
maxPD=2*tBD+20;             % max Propagation Delay
simMode=1;                  % MODE: simMode=0 fixed delay (synfire chain)
                           %       simMode=1 randomic delays (polychronous groups)
teta=37.00;%42.7;       % post-synaptic PERTURBATION (EPSP/IPSP) that causes a spike
                            % teta_min = 37.000001;
lburst=5;                  % how many spikes a busrt must fire
cburst=lburst;
dtburst=5;                  % delay in ms among spikes into the burst
ctick=dtburst;
ni=0;

% -- SET: number of NAs, number of members/assb, and NOISE injection -- %
numAssb=11;                 % SET: number of NAs in this simulation
nNa=10;                     % SET: number of nrds per NA: 3<=nNa<=200 (use EVEN numbers)
noiseFactor=0.1;           % SET: noise factor for synaptic input current
% the program calculates total num of neurons and default synaptic weights
if nNa<4; nNa=4; elseif nNa>200; nNa=200; end;
nN=nNa*numAssb;             % total neuroids in this simulation
w=teta/nNa;                 % default synaptic weights for all connections
nTarget=7*nNa+5;            % neuron target for memorizing membrane potential (analog)

% -- SET: timing for INPUT stimuli -- %
tINev=10; TIEv=tINev*ones(nNa,1);  % time for SYNfiring the STARTER event (ms)
dispFactorPG=2;             % scattering factor when dealing with polychronous groups
if simMode==1               % if simulation MODE=1 -> polychronized groups
  TmpTIEv=dispFactorPG*randn(nNa,1); % create a temporary randomized timing vector
  TIEv=TmpTIEv+TIEv;        % update the timing vector for firing the STARTER nrds
  TmpTIEv=int32(TmpTIEv).'; % timing vector / for stimuli (PG)
  TmpIEv=int32(dispFactorPG*randn(numAssb,nNa)); % timing inter-assemblies (PG)
  TmpIEv(1,:)=TmpTIEv;
end
TIEv=int32(TIEv);           % formated timing vector for STARTER stimuli int32()

% -- defining the TOPOLOGY -- %
% design here the relations among the pre- (i) and the pos-synaptic(j) assemblies
% consider not to change the relations from A0 ... A9 - reserved for system functions
T=[0 0 7 7 w tINev 0 0;  % starting PACEMAKER 1
   0 0 1 1 w tINev 0 0;  % force '1' in case the 1sr string event is '1'
   0 0 2 2 w tINev 0 0;  % force '1' in case the 2nd string event is '1'
   % A3&A4 = analog input X        A5&A6 = Analog input Y
   % BNA (System PACEMAKER) that maintains the rhythm generator (A7, A8, and A9)
```

```
   -7 7 -8 8 w tBD 0 0; -8 8 -9 9 w tBD 0 0; -9 9 -7 7 w tBD 0 0;
   -9 9 -1 1 w tBD 0 0; -9 9 -2 2 w tBD 0 0;
   -3 3 -4 4 teta/3 tBD 1 0; -7 7 -4 4 teta/2 tBD 1 0;
   %-3 3 -5 5 teta/3 tBD 1 0; -9 9 -5 5 teta/2 tBD 1 0;
  ];
% e.g: in the first line, the lower half and the higher half of NA K0 (starter event)
% is connected to the lower half and the higher half of NA A7, by a synaptic weight w,
% it is triggered after tINev (ms), the connection is normal (0=normal, full connection)
% and the type of neuron in A5 is defined by 0.
% OBS: For connecting K0 user must initialize the vector as [0 0 -x x ...)
% -- end of CONFIG -- %

%% -- 2. NETWORK STRUCTURE CREATION --
% Initializing the two main matrices on topology and delay:
W=zeros(nN);              % Matrix for Synaptic Weights NA(i) -> NA(j)
PD=int32(zeros(nN));      % Matriz for Propagation Delay in axons from NA(i) -> NA(j)

ve=-65;                   % Membrane equilibrium (rest) potential
V=ve*ones(nN,1);          % Membrane Voltage
vf=-50;                   % membrane Voltage for checking if neuron fired
vthres=vf;                % Membrane potential for threshold
vr=-70;                   % V reset for recovering potential after fired
tau=5;                    % membrane time constant (dimensionless)
r=3.4;                      % resistor in the LIF circuit

I=zeros(nN,1);                        % I[] is a vector for current inputs
Spks=[];                              % save the spiking history
Vout=[];                              % Vector for analog membrane voltage
tci=0;                                % time for current iteraction
cINev=1;                              % index for string bit IN stimuli
fINev=0;                              % flag to control string IN event

% The fist nrds (1,2,3...nNa)(=NA(K0)) are used for STARTER stimuli The
% next nrds (nNa+1 ... 2*nNa)=NAs(A1 and A2) are used for input string
% signals The next nrds (3*nNa+1 ... 5*nNa)=NAs(A3 and A4) are ANALOG input
% 1 The next nrds (5*nNa+1 ... 7*nNa)=NAs(A5 and A6) are ANALOG input 2
% Here the network STRUCTURE is created based on T[] matrix -- constructing
% the interlaced assemblies
for x=1:size(T,1)
  i1=T(x,1); i2=T(x,2); j1=T(x,3); j2=T(x,4);
  if i1==0 && i2==0        % assb K0 is special. It starts the system
    I1=i1*nNa+1:i1*nNa+nNa/2;
    I2=i2*nNa+nNa/2+1:(i2+1)*nNa;
    J1=j1*nNa+1:j1*nNa+nNa/2;
    J2=j2*nNa+nNa/2+1:(j2+1)*nNa;
  else                     % else, for ANY other assembly indexed x (Kx)
    if i1<0                % first half of the pre-synaptic assembly
      i1=-i1;              % if (-)indx consider lower half of interlaced assb
      I1=i1*nNa+1:i1*nNa+nNa/2;
    else                   % otherwise, consider higher half of interlaced assb
      I1=i1*nNa+nNa/2+1:(i1+1)*nNa;
    end
    if i2<0                % second half of the pre-synaptic assembly
      i2=-i2;
      I2=i2*nNa+1:i2*nNa+nNa/2;
    else
      I2=i2*nNa+nNa/2+1:(i2+1)*nNa;
    end
    if j1<0                % first half of post-synaptic assembly
      j1=-j1;
      J1=j1*nNa+1:j1*nNa+nNa/2;
    else
```

```
      J1=j1*nNa+nNa/2+1:(j1+1)*nNa;
    end
    if j2<0                % second half of post-synaptic assembly
      j2=-j2;
      J2=j2*nNa+1:j2*nNa+nNa/2;
    else
      J2=j2*nNa+nNa/2+1:(j2+1)*nNa;
    end
  end
% checking Connection Type(ct): HOW pre- are connected to post-synaptic neuroids
  if T(x,7)==0               % ct=0 NORMAL (all-to-all connections)
    % creating fully connections with the synaptic weight = w = T(x,5)
    W(I1,J1)=T(x,5);
    W(I1,J2)=T(x,5);
    W(I2,J1)=T(x,5);
    W(I2,J2)=T(x,5);
  elseif T(x,7)==1           % ct=1 ANALOG GATEs
    % creating a paired connection with the synaptic weight = teta/2 for
    % gating Kx, connect K(sync) and Kx with synaptic weights 0.5*w =
    % 0.5*T(x,5) each pre- is connected to a single post-synaptic neuroid.
    for y=1:size(I1,2)
      W(I1(y),J1(y))=T(x,5);
    end
    for y=1:size(I2,2)
      W(I2(y),J2(y))=T(x,5);
    end
  end
% fill in the propagation DELAY matrix ( D[] )
  if simMode==0           % SYNFIRE: timing among all assemblies is equal
    PD(I1,J1)=int32(T(x,6));
    PD(I1,J2)=int32(T(x,6));
    PD(I2,J1)=int32(T(x,6));
    PD(I2,J2)=int32(T(x,6));
  elseif simMode==1       % POLYCHRONIZED: each nrd has a singular temporal relation
    y3=0;                 % 1st half of pre- to 1st half of post-synaptic
    for y1=I1(1):I1(size(I1,2))
      y3=y3+1;
      y4=0;
      for y2=J1(1):J1(size(J1,2))
        y4=y4+1;
        PD(y1,y2)=T(x,6)-TmpIEv(i1+1,y3)+TmpIEv(j1+1,y4);
      end
    end
     y3=0;                 % 1st half of pre- to 2nd half of post-synaptic
    for y1=I1(1):I1(size(I1,2))
      y3=y3+1;
      y4=0;
      for y2=J2(1):J2(size(J2,2))
        y4=y4+1;
        PD(y1,y2)=T(x,6)-TmpIEv(i1+1,y3)+TmpIEv(j2+1,y4);
      end
    end
    y3=0;                 % 2nd half of pre- to 1st half of post-synaptic
    for y1=I2(1):I2(size(I2,2))
      y3=y3+1;
      y4=0;
      for y2=J1(1):J1(size(J1,2))
        y4=y4+1;
        PD(y1,y2)=T(x,6)-TmpIEv(i2+1,y3)+TmpIEv(j1+1,y4);
      end
    end
    y3=0;                 % 2nd half of pre- to 2nd half of post-synaptic
```

```
    for y1=I2(1):I2(size(I2,2))
      y3=y3+1;
      y4=0;
      for y2=J2(1):J2(size(J2,2))
        y4=y4+1;
        PD(y1,y2)=T(x,6)-TmpIEv(i2+1,y3)+TmpIEv(j2+1,y4);
      end
    end
  end
end
% -- end of net structure creation -- %

%% -- -- 3. SIMULATION -- %

while cINev<=size(InStr1,2)           % MAIN LOOP: for all the size of the InStr1...
  tci=tci+1;                          % time (ms) for the current iteraction

%% %%  -- Four times the program interfere and tiggers spikes directly:
% -- 1st intervention: for firing the STARTER at the time TIEv (ms)
  TmpEvFire=int32(find(TIEv==tci));   % check if nrds in STARTER event (K0) must fire
  if ~isempty(TmpEvFire)              % if so, forces a spike on all K0 neuroids
    V(TmpEvFire(:,1),1)=vf;           % by putting V(K0i) = vf mv.
  end
% -- 2nd intervention: if A1 fired but InStr1(cINev)=0, it resets A1's neuroids
  Fired=find(V>=vf);                  % find all nrds fired in this iteration
  FiredCK=find((Fired(:,1)>nNa)&(Fired(:,1)<=2*nNa), 1); % have A1 nrds fired?
  if InStr1(1,cINev)==0&&~isempty(FiredCK)               % but the InStr1 bit ='0'?
    for x=nNa+1:2*nNa
      V(x,1)=ve;                      % reset A1 nrds for default recovering V
    end
  end
% -- 3rd intervention: if A2 fired but InStr2(cINev)=0, it resets A2's neuroids
  FiredCK=find((Fired(:,1)>2*nNa)&(Fired(:,1)<=3*nNa), 1); % have NA(2) nrds fired?
  if InStr2(1,cINev)==0&&~isempty(FiredCK)                 % but the InStr2 bit ='0'?
    for x=2*nNa+1:3*nNa
      V(x,1)=ve;                                           % reset A2 nrds for default recovering V
    end
  end
  if InStr2(1,cINev)==1&&~isempty(FiredCK)
    if cburst==0
      cburst=lburst;
      ctick=dtburst;
    end
  end
  if cburst>0
    if ctick==dtburst
      V((3*nNa+1:4*nNa),1)=32;                     % reset A2 nrds for default recovering V
      ctick=ctick-1;
    elseif ctick>0
      ctick=ctick-1;
    else
      cburst=cburst-1;
      ctick=dtburst;
    end
  end

%% %% -- adjusting the counter index for the next input bit of InStr1
  Fired=find(V>=vf);                  % redo the search for fired nrds after interferences
  % increments the string pointer counter after the pacemaker 'PM-1' and 'PM-2' have fired
  if ~isempty(find((Fired(:,1)>8*nNa)&(Fired(:,1)<=9*nNa),1))&&fINev==0   % if 'PM-1' has fired
    fINev=1;                          % set the flag to 1
  end
```

```
  if ~isempty(find((Fired(:,1)>9*nNa)&(Fired(:,1)<=10*nNa),1))&&fINev==1   % if 'PM-2' has fired
    cINev=cINev+1;                    % inc counter/adjust indx for InStr1 next event
    fINev=0;                          % turn the flag back to 0
  end
%% %% resume the simulation
  Spks=[Spks; tci+0*Fired,Fired];     % save fired cells to Spks
  Vout=[Vout; tci, V(nTarget,1)];     % memorizing the analog voltage of a membrane
  V(Fired,1)=vr;                      % force reset on fired nrds membrane V()=vr
  I=noiseFactor.*randn(nN,1);         % force noise in all synaptic current
  ti=tci-maxPD; if ti<1 ti=1; end     % backward ti(ms) in time (ti=tci-maxPD)
  te=tci-1;                           % up to current time - 1(ms)
  tx=ti;
  for t=ti:te                         % for the last maxPD milliseconds
    Fired=Spks(Spks(:,1)==t,2);       % check Fired at 't'(ms) (ti<=t<=(tt-1))
    if isempty(Fired)                 % if none have fired
      tx=tx+1;
      if tx>te
        break
      end
      continue                        % just increase 't'
    end
    for i=1:size(Fired,1)           % for each nrd(i) fired in 't'
      Rgi=uint32(t+1+PD(Fired(i,1),:));
      for j=1:nN                      % for all nrds(j)
        if tci==Rgi(1,j)              % if the nrd(i) spike reach the nrd(j)
          I(j,1)=I(j,1)+W(Fired(i,1),j); % add I+W() from all fired at 't'+'D'
        end
      end
    end
  end
% tau*dv/dt = -(v-ve)+R.I(t);          % Equation for LIF (original)
%  V = V +(ve-V)/tau+I/tau;            % equation for LIF (original num)
  V = V +0.5*((ve-V)+r*I)/tau;         % R normalized = 1 (1/2 for num stability)
  V = V +0.5*((ve-V)+r*I)/tau;         % R normalized = 1 (1/2 for num stability)
end
% -- end of SIMULATION --

%% -- Plotting the results
% plot the NORMAL raster (nrds and assemblies are in sequence)
plot(Spks(:,1),Spks(:,2),'.');        % the raster plot for this simulation
%print('raster_plot', '-dpng', '-r600'); %<-Save as PNG with 600 DPI
['ok - press enter for shuffling raster']

   pause
%
%   % -- plot the raster after permuting the neuron number (line positions) --
   p = randperm(nN);                    % randomly permute
   SpksPerm=Spks;                       % image of the spike history
   SpksPerm(:,2)=0;                     % zeros for all the neuroid indexes
  for i=1:size(Spks,1)                 % for all 't' in which fires occurred
      j=find(p==Spks(i,2));             % find in 'p' the indx permutated
      SpksPerm(i,2)=j;                  % change it inton SpksPerm
  end
  plot(SpksPerm(:,1),SpksPerm(:,2),'.');% the raster after shuffling
  plot(Vout(:,1), Vout(:,2),'-');       % the raster plot for this simulation
%  print('LIF_burst_A', '-dpng', '-r600'); %<-Save as PNG with 600 DPI
['ok - this is the shuffled raster plot']
```