UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM MICROELETRÔNICA

PRISCILA CAVALCANTE HOLANDA

DHyANA: a Digital Hierarchical Neuromorphic Architecture for Liquid Computing

Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Microeletronics

Advisor: Prof. Dr. Ricardo Reis
Coadvisor: Prof. Dr. Guilherme Bontorin

Porto Alegre
December 2016

CIP — CATALOGING-IN-PUBLICATION

Holanda, Priscila Cavalcante

DHyANA: a Digital Hierarchical Neuromorphic Architecture for Liquid Computing / Priscila Cavalcante Holanda. – Porto Alegre: PGMICRO da UFRGS, 2016.

97 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Microeletrônica, Porto Alegre, BR–RS, 2016. Advisor: Ricardo Reis; Coadvisor: Guilherme Bontorin.

1. DHyANA. 2. Spiking neural network. 3. Hierarchical network-on-chip. 4. Biomimetic. 5. Neuromorphic. I. Reis, Ricardo. II. Bontorin, Guilherme. III. Título.

“Whether we are based on carbon or silicon makes no fundamental difference.
We should each be treated with appropriate respect.”

— ARTHUR C. CLARKE, 2010: ODYSSEY TWO

ACKNOWLEDGMENTS

This work became a reality with the kind support and help from many individuals. It was also the product of a large measure of serendipity, fortuitous encounters with people who have changed the course of my academic career and also my life. I would like to extend my sincere thanks to every single person who found a way to encourage, support and guide me throughout such a remarkable journey.

First and foremost, I offer my sincerest gratitude to my Advisor, Dr. Ricardo Reis, who believed me even when I couldn't. Without your encouragement and guidance this work would never be possible. To Dr. Guilherme Bontorin, my dear Coadvisor, who has always supported me with his patience and knowledge whilst always allowing me the room to work in my own way. To the professors Dr. Cesar Zeferino, Dr. Marco Idiart and Dr. Fernanda Kastensmidt, who kindly shared several valuable insights to this work.

To the Federal University of Rio Grande do Sul (UFRGS), for providing everything I needed to produce and complete my works, and to the National Council for Scientific and Technological Development (CNPq) for funding my studies. To the colleagues from Lab. 67/217, thank you for the fruitful conversations and for the pretty amazing food. To Vitor, whose support was vital for this work, from the thoughtful help in software, version control, and other technical issues to keeping my mind sane with some company and pretty good wine. To Cezar, who was always so available and kind to help me and so patient to explain even the most elementary questions. To Raysa, for helping the conception of the most beautiful images in this work.

To the researchers who, before me, felt fascinated by the world of the brain. Your contribution to science, to the world, and also to my humble work, is very much appreciated. To all the professors, from school, from the Federal University of Ceará, from Stevens Institute of Technology, and from the Federal University of Rio Grande do Sul, who inspired and helped me become who I am today. To Prof. Dr. Luiz Henrique Barreto, who first accepted and supervised some of my research ideas and encouraged me to follow them.

To the most caring and thoughtful friends in the world. To Neuza, Raysa, Victor, Raffaela, Adan, Caio, Laís and Halina, no matter how far, you have always been my strength and my safe harbor. To Simone, Geovana and Joana, longstanding friends who may not always be at sight, but are always in my heart. To Nayara, for sharing the hardest and craziest times of a lifetime. To Cristhine, my soulmate friend, who held my hand and

lifted me off the ground in all the ways one possibly can.

To my lucky charm, Lorrana, who showed me how to appreciate the beauty of unexpected encounters. Thank you for taking time to understand and revise this work. But mostly, thank you for having given me the courage to not only visit the basements of my own self, but mainly try to inhabit them.

To my fortress, my beautiful family, for the unconditional support. To my impossibly annoying and loved brother, for believing me. To the happiest and loudest cousins in the world, thanks for having my back. To my aunts and uncles, for providing the best care and the greatest example. To my grandparents, for being my heroes.

And lastly, to my parents, Aldemir e Ana, who deposit an unwavering faith in me. You have always provided me with the best opportunities and support a daughter could have ever hope for. For that and so much more, I am eternally grateful.

DHyANA: a Digital Hierarchical Neuromorphic Architecture for Liquid Computing

ABSTRACT

Neural Networks has been a subject of research for at least sixty years. From the effectiveness in processing information to the amazing ability of tolerating faults, there are countless processing mechanisms in the brain that fascinates us. Thereupon, it comes with no surprise that as enabling technologies have become available, scientists and engineers have raised the efforts to understand, simulate and mimic parts of it.
In a similar approach to that of the Human Genome Project, the quest for innovative technologies within the field has given birth to billion dollar projects and global efforts, what some call a global blossom of neuroscience research.
Advances in hardware have made the simulation of millions or even billions of neurons possible. However, existing approaches cannot yet provide the even more dense interconnect for the massive number of neurons and synapses required.
In this regard, this work proposes DHyANA (Digital HierArchical Neuromorphic Architecture), a new hardware architecture for a spiking neural network using hierarchical network-on-chip communication. The architecture is optimized for Liquid State Machine (LSM) implementations.
DHyANA was exhaustively tested in simulation platforms, as well as implemented in an Altera Stratix IV FPGA. Furthermore, a logic synthesis analysis using 65-nm CMOS technology was performed in order to evaluate and better compare the resulting system with similar designs, achieving an area of $0.23mm^2$ and a power dissipation of 147mW for a 256 neurons implementation.

Keywords: DHyANA, spiking neural network, hierarchical network-on-chip, biomimetic, neuromorphic.

DHyANA: uma Arquitetura Digital Neuromórfica Hierárquica para Máquinas de Estado Líquido

RESUMO

Redes Neurais têm sido um tema de pesquisas por pelo menos sessenta anos. Desde a eficácia no processamento de informações à incrível capacidade de tolerar falhas, são incontáveis os mecanismos no cérebro que nos fascinam. Assim, não é nenhuma surpresa que, na medida que tecnologias facilitadoras tornam-se disponíveis, cientistas e engenheiros têm aumentado os esforços para o compreender e simular.
Em uma abordagem semelhante à do Projeto Genoma Humano, a busca por tecnologias inovadoras na área deu origem a projetos internacionais que custam bilhões de dólares, o que alguns denominam o despertar global de pesquisa da neurociência.
Avanços em hardware fizeram a simulação de milhões ou até bilhões de neurônios possível. No entanto, as abordagens existentes ainda não são capazes de fornecer a densidade de conexões necessária ao enorme número de neurônios e sinapses.
Neste sentido, este trabalho propõe DHyANA (Arquitetura Digital Neuromórfica Hierárquica), uma nova arquitetura em hardware para redes neurais pulsadas, a qual utiliza comunicação em rede-em-chip hierárquica. A arquitetura é otimizada para implementações de Máquinas de Estado Líquido.
A arquitetura DHyANA foi exaustivamente testada em plataformas de simulação, bem como implementada em uma FPGA Stratix IV da Altera. Além disso, foi realizada a síntese lógica em tecnologia 65nm, a fim de melhor avaliar e comparar o sistema resultante com projetos similares, alcançando uma área de 0,23mm^2 e potência de 147mW para uma implementação de 256 neurônios.

Palavras-chave: DHyANA, rede neural pulsada, rede em chip hierárquica, biomimético, neuromórfico.

LIST OF ABBREVIATIONS AND ACRONYMS

AER	Address Event Representation
BMI	Brain-Machine Interfaces
BPDC	Backpropagation Decorrelation
BRAIN	Brain Research through Advancing Innovative Neurotechnologies
CAM	Content-Addressable Memory
CMOS	Complementary Metal-Oxide Semiconductor
DARPA	Defense Advanced Research Projects Agency
DHyANA	Digital HierArchical Neuromorphic Architecture
DSP	Digital Signal Processing
EPFL	École Polytechnique Fédérale de Lausanne
EPSP	Excitatory Postsynaptic Potential
ESN	Echo State Network
FIFO	First In, First Out data buffer
flit	flow control unit
FPGA	Field-Programmable Gate Array
FSM	Finite-State Machine
GSOPS	Giga-Synaptic Operations Per Second
HBP	Human Brain Project
HICANN	High Input Count Analog Neural Network
HICIT	Hierarchical Crossbar-based Interconnection Topology
HMF	Hybrid Multiscale Facility
HNN	Hardware Neural Networks
IBM	International Business Machines Corporation
IF	Integrate-and-Fire neuron model

IP	Intellectual Property
IPSP	Inhibitory Postsynaptic Potential
ISM	Izhikevich Simple Model
LIF	Leaky Integrate-and-Fire
LSM	Liquid State Machine
NA	Neuron Address
NETT	Neural Engineering Transformative Technologies
NI	Network Interface
NoC	Network-on-chip
P2P	Point-to-point
PCB	Printed Circuit Board
phit	physical unit
PSP	Postsynaptic Potential
RAM	Random-Access Memory
RASoC	Router Architecture for Systems-on-Chip
RC	Resevoir Computing
RI	Routing Information
ROM	Read-Only Memory
SDRAM	Synchronous Dynamic Random-Access Memory
SiP	System-in-Package
SNN	Spiking Neural Networks
SoC	System-on-chip
SRAM	Static Random-Access Memory
STDP	Spike-Timing Dependent Plasticity
UFRGS	Universidade Federal do Rio Grande do Sul
UMC	United Microelectronics Corporation

LIST OF FIGURES

Figure 1.1 It Takes the World to Map the Brain. 14

Figure 2.1 Representation of a neuron 20
Figure 2.2 Main parts of a synapse 21
Figure 2.3 Neuro-computational properties of spiking neurons. 22
Figure 2.4 Plausibility versus Complexity. 24
Figure 2.5 Example of a Spike Raster Plot. 26
Figure 2.6 Structure of a RNN in the Framework of Reservoir Computing 28
Figure 2.7 Structure of a Liquid State Machine (LSM) 30
Figure 2.8 On-chip communication structures 32
Figure 2.9 Regular Forms of Topologies 32
Figure 2.10 Irregular Forms of Topologies 33
Figure 2.11 Generic Router Model. 34

Figure 4.1 DHyANA Global Level. 44
Figure 4.2 Global Link 45
Figure 4.3 Message Composition 46
Figure 4.4 Global Packet format and RI bits. 47
Figure 4.5 Packet Routing 48
Figure 4.6 Global Level: Router components. 50
Figure 4.7 Global Router Interface and Crosspoint Matrix. 51
Figure 4.8 DHyANA Cluster Level. 53
Figure 4.9 The Neuron Implementation 55
Figure 4.10 The Neuron Schematic 56

Figure 5.1 Neurons at the Cluster Level 58
Figure 5.2 Cluster Level Packet. 59
Figure 5.3 Conceptual view of a Content-Addressable Memory. 60
Figure 5.4 Memories at the Neuron Module. 60
Figure 5.5 Controller Detail at the Cluster Level 61
Figure 5.6 ROM Data for one Neuron. 62
Figure 5.7 Pseudo-code for Global Packet Header assembly 63
Figure 5.8 Cluster Controller FSM Logic 64
Figure 5.9 Pseudo-code for Decoding the Global Packet. 65

Figure 6.1 Two Neurons Spiking After Stimulation. 67
Figure 6.2 Afferent Datapath of the Cluster Level. 68
Figure 6.3 Routers Communication 69
Figure 6.4 Datapath of the Global Level. 69
Figure 6.5 Efferent Datapath of the Cluster Level. 70
Figure 6.6 Cluster Size versus FPGA Logic Utilization. 71
Figure 6.7 Cluster Size versus FPGA Synthesis Time Elapsed 72

Figure A.1 Nível Global da DHyANA. 92
Figure A.2 Nível Local da DHyANA. 93

LIST OF TABLES

Table 2.1 Comparison of the properties of neuron models 24
Table 2.2 Parameters for a NoC Communication Infrastructure 35

Table 3.1 Related Work synthesis data 42

Table 4.1 Description of RI bits 47
Table 4.2 Parameters for neurocomputational features. 56

Table 6.1 DHyANA compared to Related Works 73

Table A.1 Descrição dos bits RI 93
Table A.2 Parâmetros para características neurocomputacionais. 95
Table A.3 DHyANA compada a Arquiteturas Relacionadas 96

CONTENTS

1 INTRODUCTION **13**
1.1 Problem Formulation and Motivation **15**
1.2 Text Organization **17**
2 BACKGROUND **19**
2.1 Spiking Neural Networks **19**
2.1.1 Spiking Neurons: A Biological Perspective 19
2.1.2 Neuron Mathematical Models 21
2.1.3 Networks and Information Coding 25
2.2 Reservoir Computing **27**
2.2.1 Liquid State Machine 29
2.3 Networks-on-chip **31**
3 STATE-OF-THE-ART **37**
3.1 Non-scalable or Special Purpose Works in FPGA **37**
3.2 Reservoir Computing in Hardware **38**
3.3 Scalable and Large Scale General Purpose SNNs in Hardware **39**
4 PROPOSED NEUROMORPHIC ARCHITECTURE **43**
4.1 Global Level: The Mesh NoC Interface **45**
4.1.1 The Global Router 49
4.2 Cluster Level: The Bus Interface **52**
4.3 Processing Element: The Neuron **53**
5 CLUSTER LEVEL **57**
5.1 Neuron module **57**
5.2 The Cluster Controller **61**
6 RESULTS **66**
6.1 Testbench Setup **66**
6.2 FPGA Implementation **70**
6.3 Related Work Comparison and Final Remarks **72**
7 CONCLUSIONS AND FUTURE WORK **74**
APPENDIX A — RESUMO EM PORTUGUÊS **87**
A.1 Resumo **87**
A.2 Introdução **87**
A.2.1 Redes Neurais Pulsadas 87
A.2.2 Máquinas de Estado Líquido 88
A.2.3 Redes em Chip 89
A.2.4 Estado da Arte 90
A.3 Arquitetura Proposta **91**
A.3.1 Nível Global 91
A.3.2 Nível Local 93
A.3.3 Núcleos de Processamento: Neurônios 95
A.4 Resultados **95**
A.5 Conclusão e Trabalhos Futuros **97**

1 INTRODUCTION

The brain is an amazing and highly efficient three-pound machine, and definitely one of the most complex and magnificent structures in the human body. It is formed by a network of more than 100 billion single nerve cells interconnected in systems that construct our perceptions of the world, fix our attention, and control the machinery of our actions (KANDEL et al., 2013). Understanding how complex processes in the brain give rise to complex behavior is one of the key scientific challenges in the twenty-first century, and a grown effort is rising in a global scale.

Furthermore, for the last fifty years, Moore's Law have powered tremendous advances in computing, allowing supercomputers to reach petaflop information-processing rates. Such a growth rate is by far the largest of any kind within the roughly 10,000 years of human civilization, and have revolutionized science, technology and medicine to the extent that it is now becoming feasible to simulate networks of neurons, brain regions and, eventually, the whole brain.

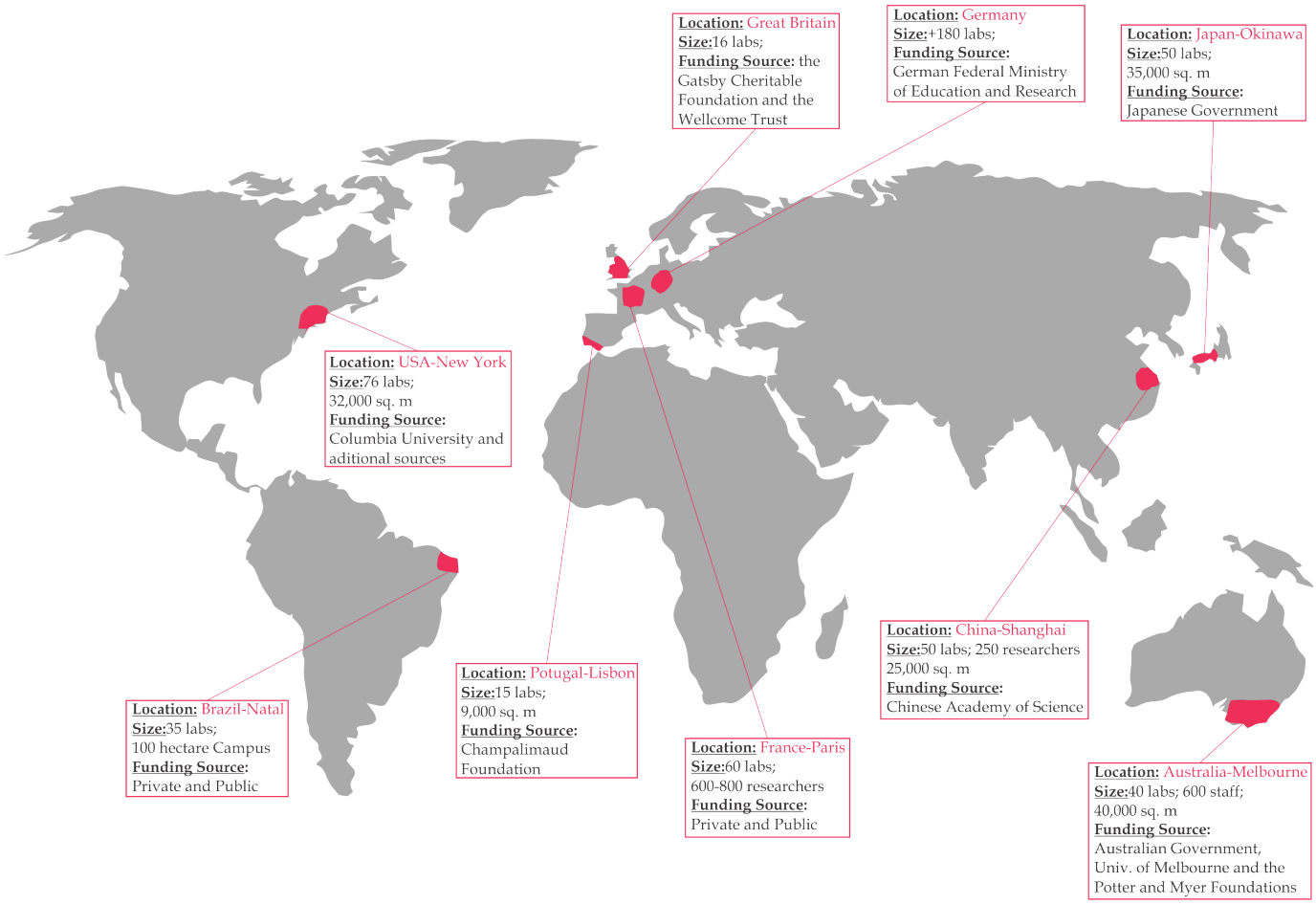
One of the first brain-scale models ever developed was the thalamocortical model by Izhikevich and Edelman, in which a million spiking neurons were simulated (IZHIKEVICH; EDELMAN, 2008). Interestingly, the model exhibited behavioral phenomena similar to normal brain activity that were not previously built in, such as spontaneous activity, sensitivity to changes in individual neurons and rhythms of spiking activity.

Around the same time, Henry Markram launched the highly publicized Blue Brain Project, a partnership with IBM and the École Polytechnique Fédérale de Lausanne (EPFL). Using the massive computing power of IBM's Blue Gene supercomputer, the project aims to simulate the brains of mammals with a high level of biological accuracy (MARKRAM, 2006).

From that on, a number of large-scale projects have risen. The multi-million dollar brain research initiatives such as European Union's HBP and NETT, Unites States MindScope and BRAIN Initiative, Japan's Brain/MINDS as well as the increasing number of research facilities worldwide (See Figure 1.1) prove that we are at a historical moment for the brain.

These researches can bring several benefits to the human kind. From the ability to better understand and seek treatment for brain disorders to the expertise to model and develop intelligent systems, the motivations for implementing brain models are countless. The ability to develop Brain-Machine Interfaces (BMIs) and neuroprostheses, for instance,

Figure 1.1: It Takes the World to Map the Brain.



Source: based on (SEGEV, 2016)

would help improve the quality of life of several people affected by neurological disorders, such as Parkinsons' and Alzheimer's disease. As of 2016, cochlear implants are among the most successful ones, and in about 30 years, more than 220.000 patients worldwide now enjoy restored hearing because of such devices. Other research cases developed brain-activated upper and lower limb prostheses for amputees and paralytics, as well as retinal implants and even hippocampus emulation.

Still, to become possible, brain-scale models require yet more computational power. And as the Moore's law becomes less feasible, further computational optimizations have to be researched in order to promote exascaling computing, for it is believed to be the order of processing power of the human brain at neural level.

1.1 Problem Formulation and Motivation

It is estimated that the brain is composed of approximately 10^{11} processing elements, the neurons, which are extremely specialized cells. They generate electrical signals in response to chemical agents or other inputs and disseminate them through its axons to other cells (DAYAN; ABBOTT, 2005). The neuronal ionic mechanisms that generate the action potentials are known today mostly due to the pioneering work developed by Hodgkin and Huxley in 1952 (HODGKIN; HUXLEY, 1952). The neuron behavior is modeled by their approach by using four differential equations to represent the membrane dynamics and the non-linear conductances of three types of ion channels.

Since then, a lot of mathematical models were developed, such as the Integrate-and-Fire and the Izhikevich models (IZHIKEVICH, 2004), whose networks were proven to be computationally more powerful and yet consume lower power than the McCulloch–Pitts and threshold gates based ones (MAASS, 1996; MAASS, 1997; ZHANG et al., 2015). Consequently, there has been a lot of effort toward developing more biologically inspired learning algorithms, network structures, and applications of Spiking Neural Networks (SNNs).

Individual elements of information, however, are encoded in the brain by populations or clusters of interconnected neurons, rather than by single cells (POUGET; DAYAN; ZEMEL, 2000). It is also accepted that it is the type and timing of these spike trains that encodes communications between neurons within the brain. Consequently, there has been a lot of effort toward developing more biologically inspired Spiking Neural Networks and learning algorithms, thus aiming at the potential sources of cognition itself.

SNNs are essentially a population of spiking neurons, exchanging spikes to each other via weighted connections, reflecting the way real neurons project to others and interact through synapses (CAO; PIPA, 2010). Implementations of SNNs are largely used, and can be applied in various areas, such as character recognition (KULKARNI; BAGHINI, 2013; LIU; YUE, 2014), medical diagnosis and analysis (SUN et al., 2011; ROY; SCHAFFER; LARAMEE, 2013), financial predictions (REID; HUSSAIN; TAWFIK, 2013) and robotics (HULEA; CARUNTU, 2014; KERR et al., 2012; WANG et al., 2010; ALNAJJAR; MURASE, 2005), to name a few.

The development of SNNs in hardware has been a wide research area, since it benefits from the intrinsic parallel nature of hardware implementations, allowing very large speedups compared to software implementations in sequential machines. Several overview

and survey publications on digital hardware implementations of neural networks have been published (MAGUIRE et al., 2007; MISRA; SAHA, 2010; SCHÄFER et al., 2002).

Nevertheless, SNNs are difficult to train in a supervised fashion, mostly since all simple spiking neuron models have hard thresholding, which makes the calculation of gradients very prone to errors, deteriorating the learning rule's performance (SCHRAUWEN et al., 2008). One way to circumvent this drawback is by using fixed parameters, which is what is embodied by the Liquid State Machine (LSM) concept, developed by Maass, Natschläger and Markram (2002). Inspired by the fact that the neocortex processes a wide spectrum of information by stereotypical neural microcircuitry, the LSM consists of a reservoir receiving input spike trains and a group of readout neurons receiving signals from the reservoir. The concept, in which a recurrent network of spiking neurons is constructed in a way that the network parameters are fixed and randomly chosen, is similar to that of Echo State Network (ESN) (JAEGER, 2001a), being both part of a new paradigm named Reservoir Computing (RC) (VERSTRAETEN, 2009).

LSMs could benefit even more of hardware implementations, since for most of its applications, quite large networks of spiking neurons need to be simulated with hard real-time constraints (SCHRAUWEN et al., 2008). Furthermore, multiple outputs can be generated from the same reservoir, which in turn allows the implementation of a generic hardware reservoir component operating on different applications and with multiple outputs.

Nevertheless, the ever increasing size of the networks poses a huge challenge when it comes to its high inter-neuron connectivity requirements. If one would have the need to simulate the interconnection topology of, say, a mammalian neocortex, the number of synapses per neuron should be in average between 2×10^3 and 2×10^4 (JOHANSSON; LANSNER, 2007), which is a significant limiting factor in the suitability of its implementation in hardware. In the context of a similar connectivity problem on System-on-Chip (SoC) design, where interconnect scalability is paramount, the works from Benini and Micheli (2002), Dally and Towles (2001), Hemani et al. (2000) and Jantsch and Tenhunen (2003) introduced the concept of Network-on-Chip (NoC). Within such concept, elements from traditional computer networking are employed in order to realize the communication of the hardware structure.

There are already some efforts, both in academy and industry, in which the concept of Networks-on-Chip were utilized in order to implement highly scalable large-scale hardware neuromorphic systems. The magnitude of such projects has shown a large growth

in the number of neurons and synapses over the past years, relatively to the study in (MISRA; SAHA, 2010). Such a growth, although not yet near of reaching the human brain scale, is very encouraging since the ever-increasing ability to emulate parts of the brain makes it possible to understand its functioning. It also enables the improvement of computing tasks that involve learning and autonomous decision making capabilities such as autonomous robotics and assistive technologies.

However, there is still a huge ground to be covered, and the remaining challenges are also fundamental issues. There is still no consensus within the literature as of appropriate neuron models, interconnect and architectural issues, etc. There are also various design choices that need to be explored, such as new combinations of hierarchical topologies and the use of more complex neuron models within the systems. It is thus vitally important that such trend of research keep its growth in order to address the remaining challenges, so that the ambition to better understand the brain can be overcome.

The aim of this work is, consequently, to present DHyANA, a digital spiking neural network architecture which is optimized for liquid computing and employs the NoC concept for its inter-neuron communication. The system is intended to be as scalable, complex and biologically-realistic as possible, so it can ultimately assist in the exploration and understanding of neuroscience.

The main contributions of this work can be divided in two main parts: the development of DHyANA, an architecture which employs hierarchical topology and network-on-chip communication, intended to be used in later neuroscience and machine learning research; and the design of the hierarchical infrastructure of the DHyANA, by implementing a bus-based communication, used in the Cluster Level of the architecture, as well as the control, assemblage and decryption of the Global Network-on-Chip Packets.

1.2 Text Organization

Chapter 2 presents the background theory and the challenges faced when implementing each technology. Chapter 3 presents a review of the main current designs on neuromorphic hardware, divided into three categories. The sessions thus introduce the state-of-the art for non-scalable or special purpose works in FPGA, hardware implementations of LSMs and scalable and large scale general purpose SNNs in hardware (mostly using NoCs). Chapter 4 describes the proposed architecture, exploring its building blocks and design choices, as well as how they have been put together in order to become the

DHyANA architecture. Chapter 5 explores the Cluster Level, describing in deep detail its architecture and how it fits within the whole system. Chapter 6 indicates the results and comparison with the literature and industry state-of-the-art. Finally, Chapter 7 concludes the work and points some directions for future research.

2 BACKGROUND

In order to understand the constraints involved in neural networks modelling, the beginning of this chapter will explore some neurobiological premises. With a focus on the neurons, the first section offers a sufficient introduction to the phenomenological level of the models to be presented, even though they are a simplification of the true complexity of neurobiology. Next, the mathematical models that are used to simulate neuronal cells and their respective waveforms are covered, as well as the main trade-offs on their implementation. Then, a brief on information coding and networks of neurons will be presented, introducing the key aspects of timing and learning.

Hereafter, the key features of Reservoir Computing will be covered. The Liquid State Machines will be emphasized, for the aim of this work is to optimize the proposed architecture for this specific kind of network.

Lastly, the concepts and technologies of Networks-on-Chip will also be reviewed. Since it is a very important part of this project, NoCs will be carefully explained in the third section, as well as some of the most important design premises necessary for its implementation.

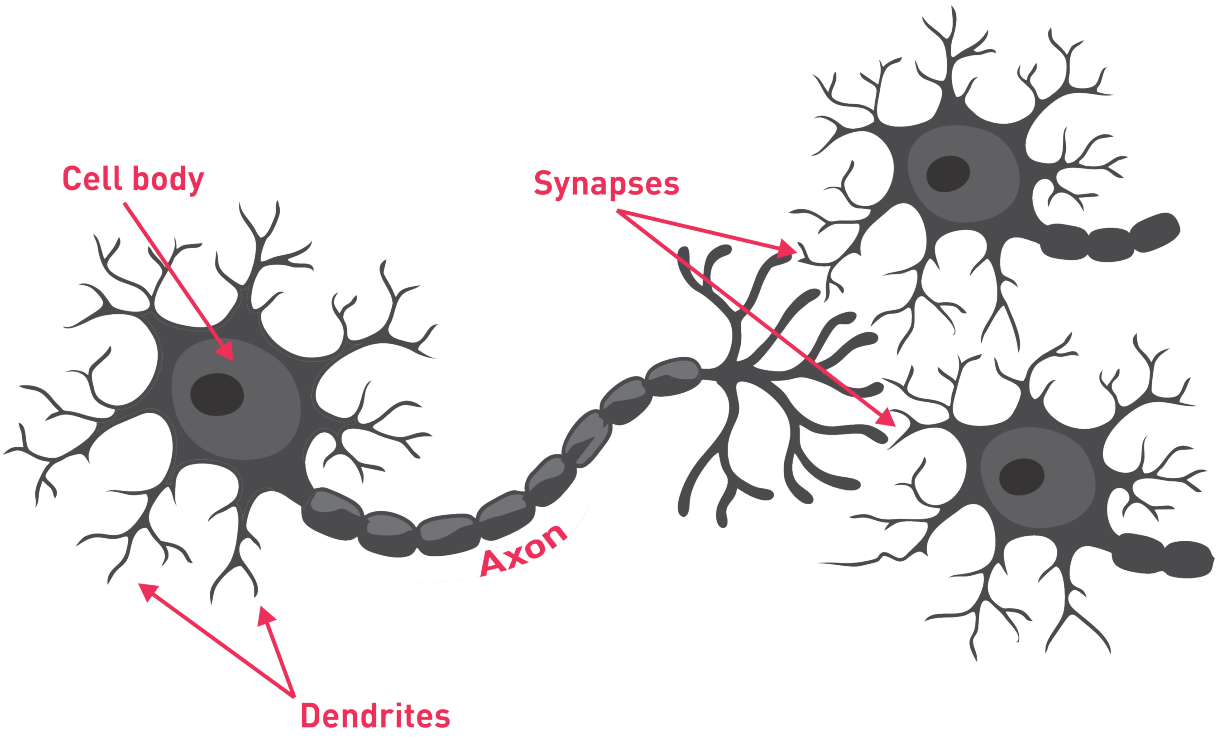
2.1 Spiking Neural Networks

2.1.1 Spiking Neurons: A Biological Perspective

A neuron is an extremely specialized cell, which generates electrical signals in response to chemical agents or other inputs, and disseminate them through its axons to other cells (DAYAN; ABBOTT, 2005). As observable in Figure 2.1, the neuron is composed mainly by the cell body, or soma, the dendrites, and the axon, which may cross large portions of the brain or even of the entire body.

The axon may divide itself in order to connect with the dendrites of many other cells. In fact, the gray matter of the human brain contains a large amount of such axons, in the order of 4 km in every cubic millimeter (MAASS, 2002). These connections, which can be seen as some kind of chemical resistor, are called the synapses. The electrical potential between the intra- and extracellular medium of a neuron, which under resting (polarized) conditions is about -70mV, is the relevant signal to the nervous system (DAYAN; ABBOTT, 2005).

Figure 2.1: Representation of a neuron



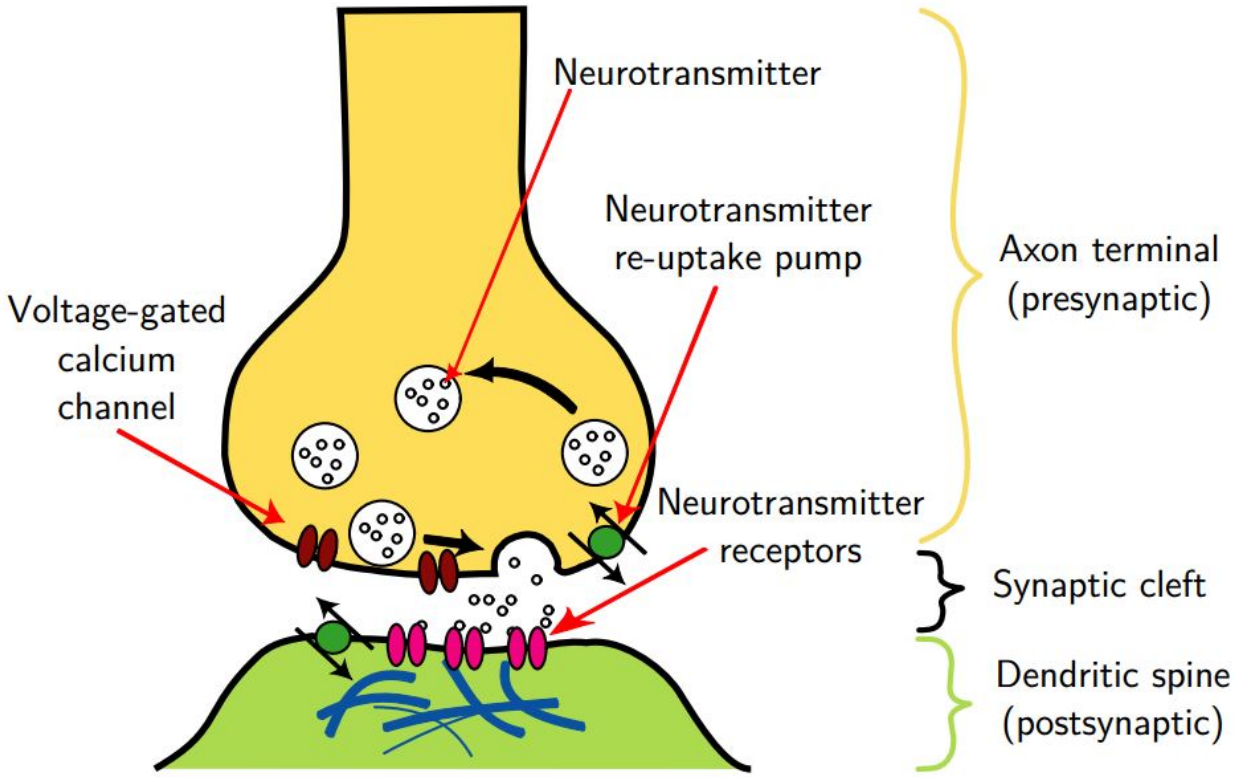
Source: The author

A typical neuron receives more than 10,000 inputs from other neurons through their synapses (MAGUIRE et al., 2007), which produce electrical transmembrane currents that change the membrane potential of the neuron. These changes are called postsynaptic potentials (PSPs). Large currents may produce significant PSPs, which can be amplified by the voltage-sensitive channels embedded in the neuronal membrane, and that can generate an action potential or spike.

During a spike, the synapses release a neurotransmitter that quickly diffuses to the post-synaptic neuron. In the post-synaptic neuron, these neurotransmitters affect the neuron's membrane potential. Excitatory Postsynaptic Potentials (EPSPs) increase the membrane potential (depolarize), while Inhibitory Postsynaptic Potentials (IPSPs) decrease the membrane potential (hyperpolarization) (GRUNING; BOHTE, 2014).

In contrast to the spikes, which are all very similar, the size and shape of these PSPs depends on the particular synapse that causes it. Actually, it will also depend on the current "mood" and the recent "experiences" of the particular synapse, since the postsynaptic potentials have different sizes, depending on the pattern of spikes that have reached the synapse in the past, on the interaction of these spikes with the firing activity of

Figure 2.2: Main parts of a synapse



Source: Bekolay (2011)

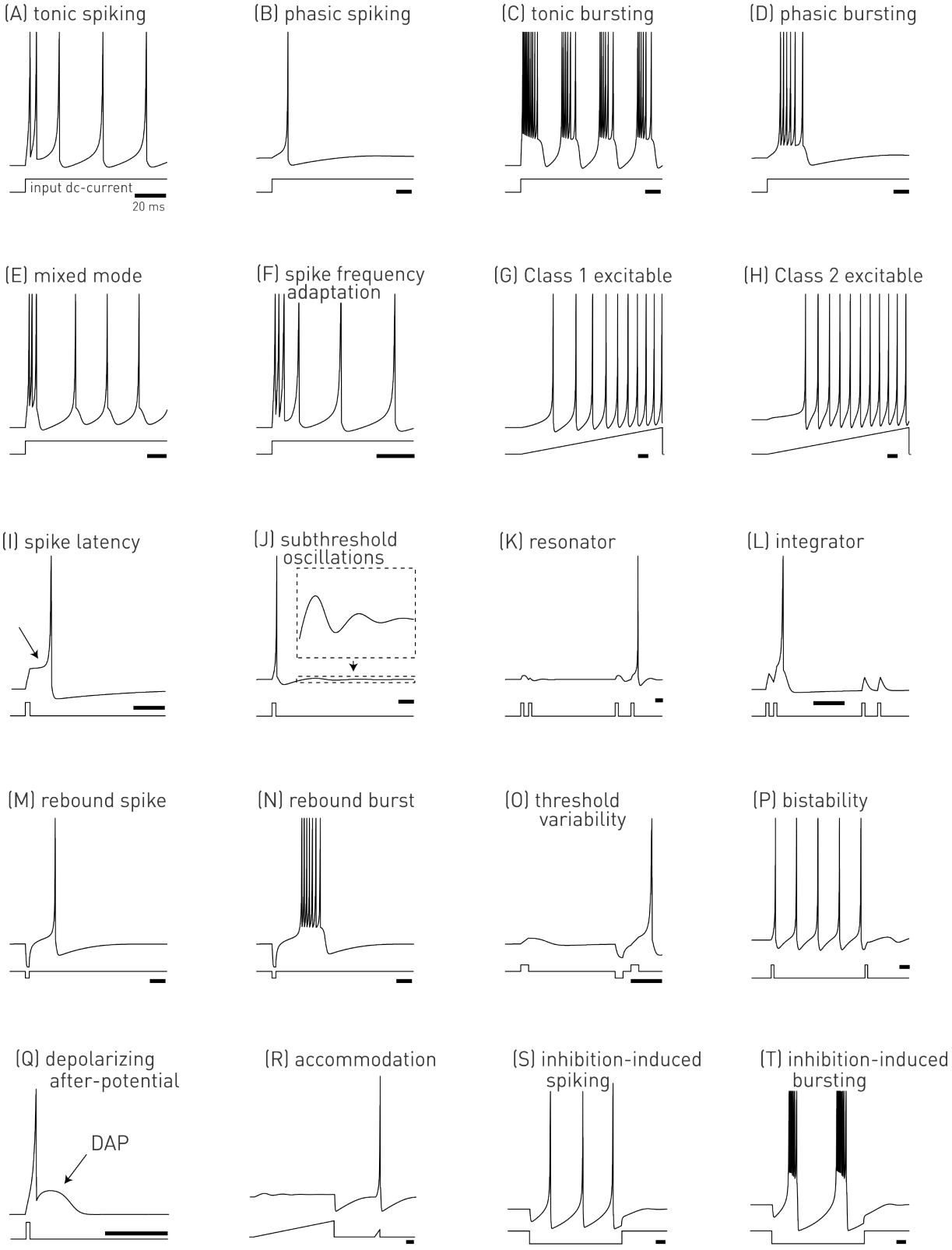
the postsynaptic neuron, and also on other signals that reach the synapse in the form of various molecules (e.g. neurohormones) through the extracellular fluid (MAASS, 2002).

Furthermore, depending on the type or state of a neuron, or even the timing and type of inputs it just received, the characteristics of the spike train will also change. Hence, there are some different spiking modes, including but not limited to: phasic spiking, where a single isolated spike is fired; tonic spiking, where single spikes are fired at regular intervals; and tonic bursting, where small bursts of spikes are fired at regular intervals (THOMAS; LUK, 2009). They can be seen in Figure 2.3.

2.1.2 Neuron Mathematical Models

There are many spiking neuron models, each varying in levels of complexity, computational intensity and biological accuracy, and a good overview of them can be seen in (IZHIKEVICH, 2004). The most biologically accurate and one of the most commonly used mathematical models of the neuron is the one developed by Hodgkin and Huxley

Figure 2.3: Summary of the neuro-computational properties of biological spiking neurons.



Source: Izhikevich (2004)

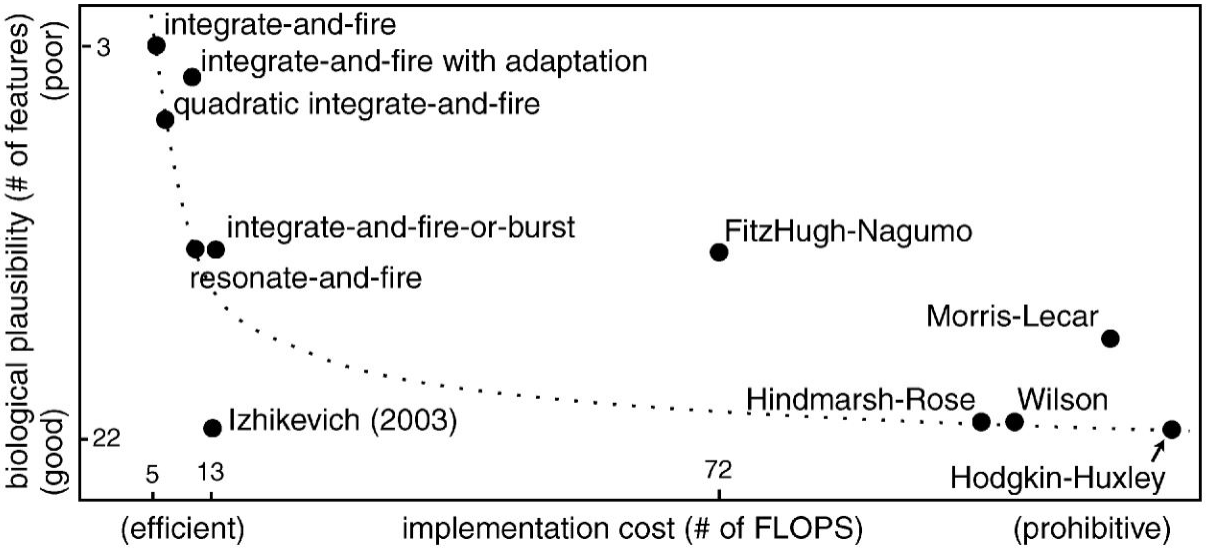
(1952). By researching the behavior of giant squid neurons (whose size is 100 to 1000 times larger than those in the human brain), they developed a mathematical model which can reproduce all kinds of neurons with a good precision in terms of shape of spikes and complex firing activities (AMBROISE et al., 2013). Nevertheless, this model also requires the estimation of a large number of parameters. Also, perhaps its major drawback is that the mathematical model is very computer intensive, as it uses up to as much as ten differential equations per neuron (FOX, 2013), which makes it even more problematic, especially in FPGA implementations, if one considers its hardware limitations.

Since the work from Hodgkin and Huxley was introduced, numerous models have been made in order to reduce its complexity. Nevertheless, none of the models is able to reach such biophysical accuracy, as well as the number of neural behaviors that the Hodgkin-Huxley model can reproduce. One of the simplest models is the Leaky Integrate-and-Fire (LIF), which uses only one differential equation to model the behavior of a neuron (FOX, 2013). It basically idealizes a neuron as having Ohmic leakage current and a number of voltage-gated currents de-activated at rest (IZHIKEVICH, 2007). The differential equation describes the voltage given by a capacitor charge, viewing the neuron as an integrator that, at a certain threshold voltage, is said to fire. Unfortunately, such simple mathematical model has flaws, in the sense that it is not as nearly biologically plausible, and may even be a waste of time for a computational neuroscientist who would want to simulate large-scale networks (IZHIKEVICH, 2007).

Another commonly used neuron mathematical model was proposed by Izhikevich (2003), and stands in the middle ground between complexity and biological plausibility scale, as observable in Figure 2.4. In fact, although not as simple as the integrate-and-fire, the Izhikevich model can mimic various nonlinear responses of biological neurons, making it almost as versatile as the Hodgkin-Huxley model at a fraction of its computational cost.

Note that, according to the metrics used by Izhikevich, the model proposed by him stands in an attractive position of the graphics when it comes to the trade-off between the two axes. Therefore, the model proves itself an interesting choice for digital implementations of neural networks, fact also confirmed by Table 2.1.

Figure 2.4: Biological Plausibility versus Computational Complexity of Spiking Neuron Models



Source: Izhikevich (2004)

Table 2.1: Comparison of the neuro-computational properties of spiking and bursting models. The "BIO" field means whether the model is biophysically meaningful; "FLP" is short for number of flops, an approximate number of floating point operations (addition, multiplication, etc.) needed to simulate the model during a 1 ms time span.

Models	BIO	(A)	(B)	(C)	(D)	(E)	(F)	(G)	(H)	(I)	(J)	(K)	(L)	(M)	(N)	(O)	(P)	(Q)	(R)	(S)	(T)	FLP
IF		x						x					x									5
IF with adapt.		x					x	x					x					x				10
IF-or-burst		x	x	x	x		x	x					x	x	x		x	x				13
resonate-and-fire		x	x					x	x		x	x	x	x			x	x	x			10
quadratic IF		x						x		x			x			x	x					7
Izhikevich		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	13
FitzHugh-Nagumo		x	x		x			x		x	x	x		x		x	x		x	x		72
Hindmarsh-Rose		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	120
Morris-Lecar	x	x	x		x			x	x	x	x	x	x	x	x	x	x		x	x		600
Wilson		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	180
Hodgkin-Huxley	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1200

Source: based on Fig 2 from Izhikevich (2004)

The neuron model idealized by Izhikevich is described by the following two coupled differential equations.

$$\frac{dv}{dt} = 0.04v^2 + 5v + 140 - u + I_{Izh} \tag{2.1}$$

$$\frac{du}{dt} = a(bu - v) \tag{2.2}$$

The variable *v* denotes the membrane potential, and *u* represents membrane recovery parameter. I is the synaptic input, and a, b are parameters controlling the dynamical behavior of the neural model. There is also a reset condition, controlled by the parameters c, d, and defined by equation (2.3).

$$v \geq 30mV \Rightarrow \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases} \tag{2.3}$$

2.1.3 Networks and Information Coding

The aim of artificial spiking neural networks is to carry out neural computation. This requires that meaning is given to neural spiking: the quantities relevant to the computation have to be expressed in terms of the spikes that neurons communicate with (GRUNING; BOHTE, 2014).

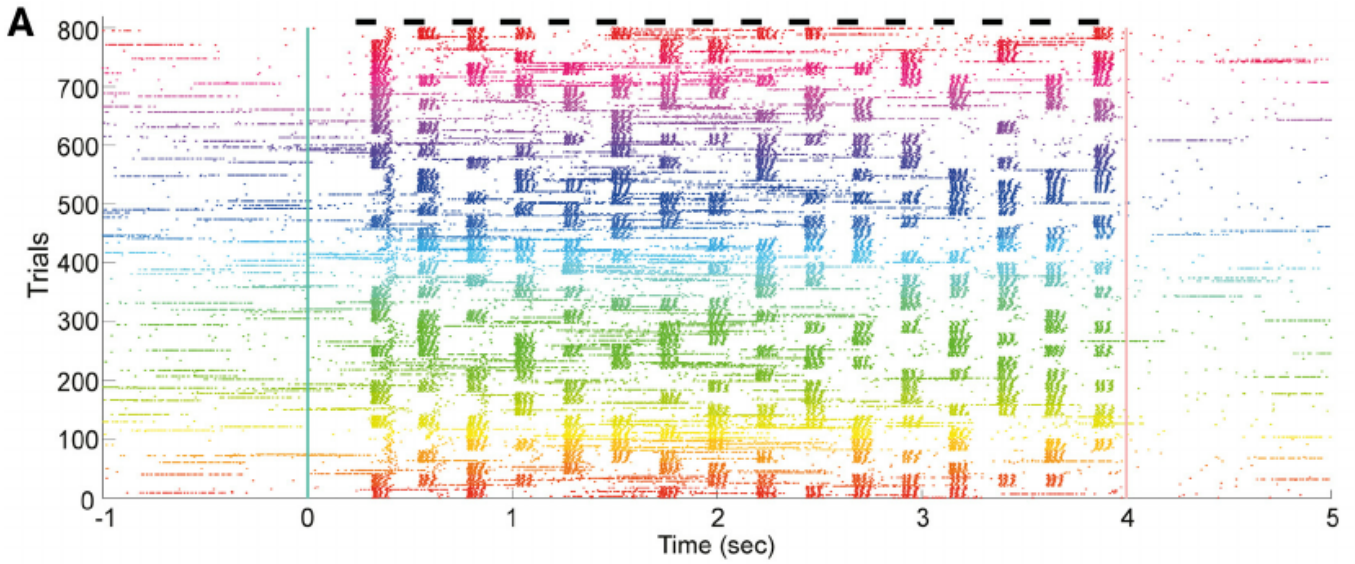
It is an accepted theory that individual elements of information are encoded in the brain by populations or clusters of interconnected neurons. Furthermore, the type and timing of these spike trains are essential for the encoding of communications between neurons within the brain.

Hence it would be appropriate to compare the output of a network of neurons with a piece of music played by an orchestra. To recognize such piece of music it is not enough to know how often each note is played by each musician. Instead one has to know how the notes of the musicians are embedded into the melody and into the pattern of notes played by other musicians (MAASS, 2002).

From this point of view, SNNs behave as complex systems, with "*emergent macroscopic-level properties resulting from the complex dynamic interactions between neurons, but hard to understand just looking at the microscopic-level of each neuron processing*" (PAUGAM-MOISY; BOHTE, 2012). A way to visualize the temporal computation processed by an SNN is by displaying a complete representation of the network activity on a spike raster plot as in Figure 2.5, in which a small bar or dot is plotted each time that a neuron fires a spike. Variations and frequencies of neuronal activity can be observed in such diagrams, in the same way as natural neurons activities can be observed in spike

raster plots drawn from multi-electrode recordings.

Figure 2.5: Example of a Spike Raster Plot. A small dot is plotted each time (in abscissa) a neuron (numbered in ordinates) spikes.



Source: Ju et al. (2015)

Since the basic principle underlying SNNs is so radically different from McCulloch–Pitts' first generation Neural Networks, it comes with no surprise that much of the work on such networks, such as learning rules and theoretical results, had to be adapted, or even fundamentally reconsidered.

Traditionally, neural networks have been applied to pattern recognition, in various guises. For instance, multi-layer networks can perform highly accurate handwritten character recognition, financial predictions, robotics, as well as function approximation, or regression, by using classic learning rules, such as error-backpropagation, Hebbian learning or distance based variants like Kohonen self-organizing maps.

Efforts have been made in the direction of adapting traditional learning methods to spiking neural networks both by augmenting weights with delay lines and using temporal coding. Among such methods are: SpikeProp, a method similar to traditional error back-propagation proposed by Bohte, Poutre and Kok (2002); RProp a learning rate adjustment technique, first adapted to SNNs by McKennoch, Liu and Bushnell (2006); and QuickProp, where Newton's method is used for minimizing the error-gradient, first adapted to SNNs by Florian (2012); For more on the subject, the reader is referred to (BEKOLAY, 2011).

However, there has been another direction within the field, in which networks and computational algorithms are exclusively developed for spiking neural networks (PAUGAM-

MOISY; BOHTE, 2012). They use the temporal domain as well as the increased complexity of SNNs to arrive at novel methods for temporal pattern detection with Spiking Neuron Networks. Among the examples are: ReSuMe, a biologically inspired method in which the result is obtained from a spike-timing dependent plasticity rule, proposed by Potjans, Morrison and Diesmann (2009); the Spike-Timing Dependent Plasticity (STDP) (MARKRAM et al., 1997) rules; and the so-called Reservoir Computing, a new paradigm which was based on the observation that as long as a randomly generated network possess certain properties, it is not necessary to train it, and training only a recurrence-free linear readout is sufficient for many tasks (LUKOSEVICIUS; JAEGER., 2007). The Reservoir Computing paradigm, which recently triggered many practical applications of Recurrent Neural Networks and a whole new stream of research, will be overviewed in the next section.

2.2 Reservoir Computing

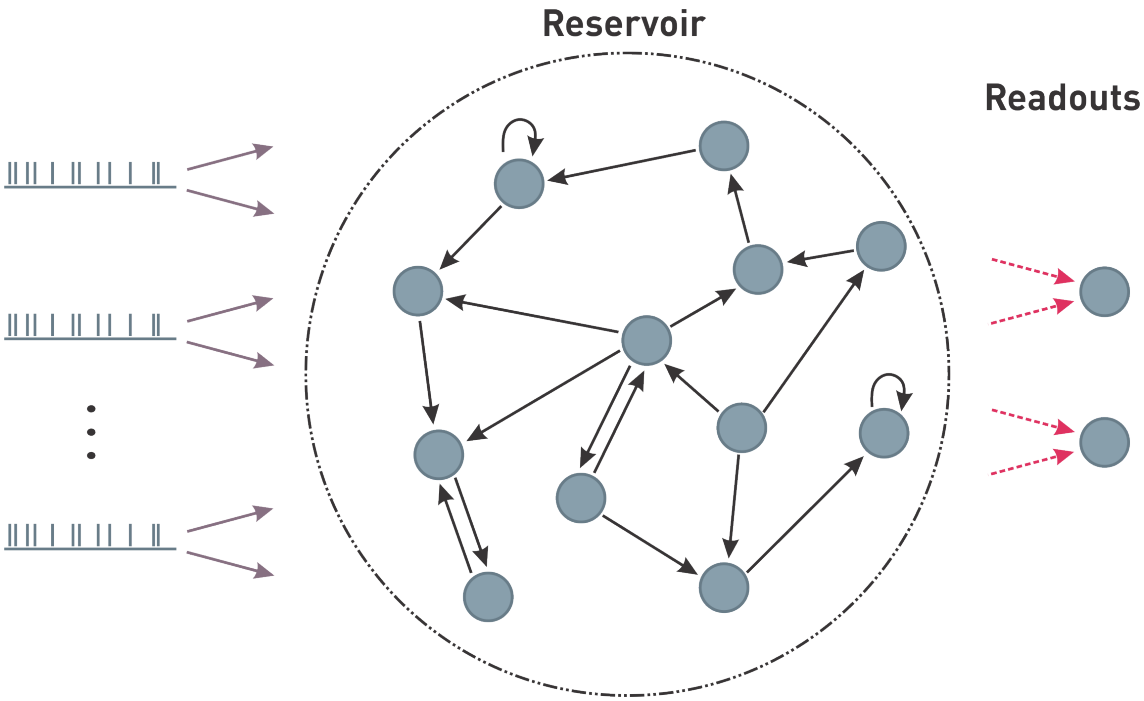
The concept of Reservoir Computing was introduced by two independent publications, both of which, although using different approaches, presented the idea of a dynamic recurrent reservoir of processing units left in a transient state, while only the output layer is subject to a supervised learning. The first publication, by Jaeger (2001a), in which the Echo State Networks were introduced, explored applications of randomly connected recurrent networks of sigmoidal neurons to complex time series prediction tasks. The second one, by Maass, Natschläger and Markram (2002), which is a more biologically oriented approach, considered reservoirs of spiking neurons structured and functioning as inspired by the properties of neocortical columns in the central nervous system of mammals.

Later on, the Backpropagation Decorrelation (BPDC) learning rule was also introduced, along with a few other methods better described in (LUKOSEVICIUS; JAEGER., 2007; LUKOSEVICIUS; JAEGER, 2009; SCHRAUWEN; VERSTRAETEN; CAMPENHOUT, 2007). The fact that similar ideas were independently discovered several times underlines the importance of such concept, which is why the group of Benjamin Schrauwen proposed in (VERSTRAETEN et al., 2007) the unification of such ideas into a common research stream, which they propose to call Reservoir Computing.

The intuitive motivation behind this concept can be explained by imagining the use of a liquid (such as a glass of water) to perform computation, as explained by Cao and Pipa (2010). From a dynamical system's perspective, this does not make much sense since the only stable state to which the liquid can converge after a perturbation (e.g a drop in the

liquid) is a "dead" state where it is perfectly still. However, the idea of reservoir computing is that the transient states of the liquid at given time still holds relevant information about a perturbation that occurred at previous time. Providing that the liquid has the ability to produce significantly different transient states under two different inputs, it is then theoretically possible to extract information directly from measures on the state of the liquid since it is supposed to hold information about the past (e.g. successive pictures of the perturbed glass of water). (See Figure 2.6).

Figure 2.6: Structure of a RNN in the Framework of Reservoir Computing; Only Dotted Synaptic Connectivities are Trained.



Source: the author

In fact, the study in (FERNANDO; SOJAKKA, 2003) actually took the “reservoir” and “liquid” concepts quite literally and successfully trained a readout multilayer perceptron on several classification tasks by feeding input via mechanical actuators into a reservoir full of water and recording the state of its surface optically.

Several successful applications of reservoir computing to both synthetic data and real world engineering applications have been reported in the literature, including autonomous sine generation (JAEGER, 2001a), dynamic pattern classification (JAEGER, 2001b) and wind speed forecasting (FERREIRA et al., 2008). RC systems were used in robotics to control a simulated robot arm (JOSHI; MAASS, 2004), to model an exist-

ing robot controller (BURGSTEINER, 2005), for action prediction (BARAGLIA; NAGAI; ASADA, 2013), and for localization and event detection (HERTZBERG; JAEGER; SCHöNHERR, 2002; ANTONELO; SCHRAUWEN; STROOBANDT, 2008). Applications in the field of Digital Signal Processing (DSP) have also been reportedly successful. Among those are speech recognition (VERSTRAETEN et al., 2005; GHANI et al., 2008), facial expression recognition (GRZYB et al., 2009), noise modeling (JAEGER; HAAS, 2004) or even classification of music styles (JU; XU; VANDONGEN, 2010).

2.2.1 Liquid State Machine

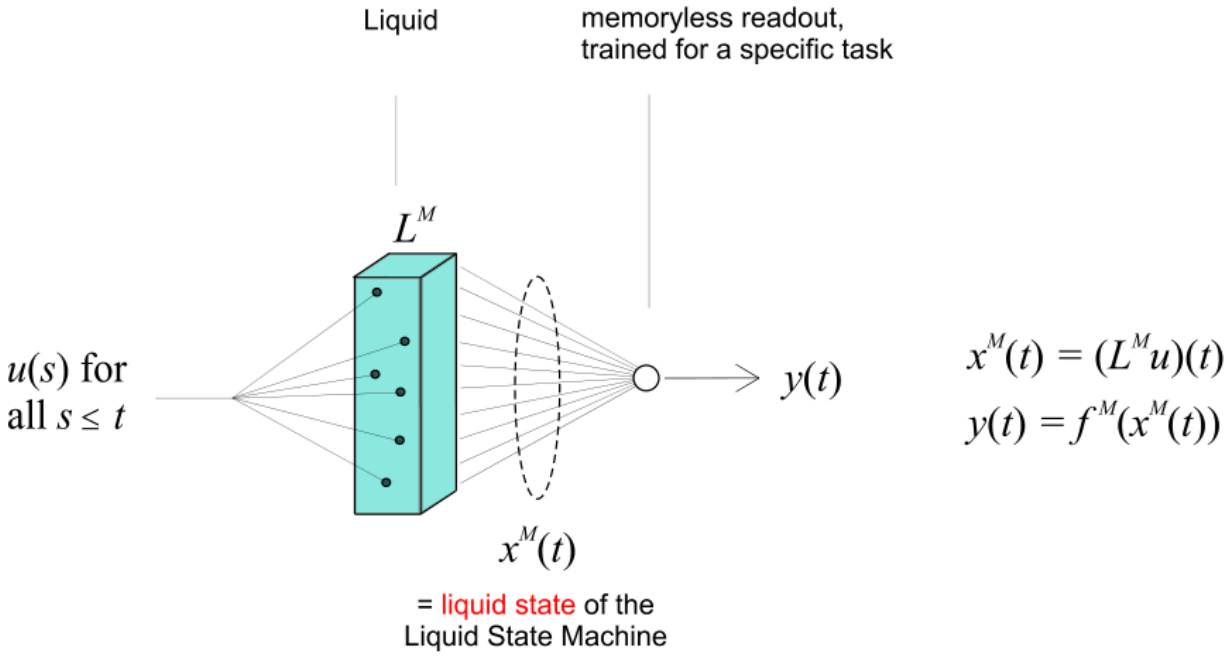
LSMs were introduced by a research lab active in robotics and neuroscience in the search for computational models that could help understanding the computations carried out in a local circuit of neurons in the neocortex (e.g. in a "cortical column"). Since it turned out to be quite successful, making it possible to carry out quite demanding computations with circuits consisting of spiking neurons and dynamical synapses, this approach will be the focus of this work.

The LSM was motivated by the hypothesis that the learning capability of an information processing device is its most delicate aspect, and that the availability of sufficiently many training examples is a primary bottleneck for goal-directed (i.e., supervised or reward-based) learning (MAASS, 2011). Thus, its architecture is composed of two layers (YAMAZAKI; TANAKA, 2007).

One layer is a reservoir of interacting spiking neurons showing recurrent topology that maps inputs into a dynamical state by generating a spatiotemporal activity pattern of neurons called a "liquid state". The network has to generate different liquid states for different input signals, because the performance of a liquid state machine mostly relies on the quality of the activity patterns. The other layer consists of neurons called "readouts", which receive the liquid state and instruction signals, compare its output to a target output in the training procedure and adapts its synaptic weights using a stateless learning rule.

Since all that needs to be trained is the readout module, instead of the recurrent neural network itself, learning is made fast and robust. The bulk of the LSM (the "Liquid") thus serves as pre-processor for such readout neuron, which amplifies the range of possible functions of the input streams $u(t)$ that it can learn (MAASS, 2011). This also means that the same Liquid can serve a large number of different readout neurons, and each one can learn to extract a different "summary" of information from the same Liquid.

Figure 2.7: Structure of a Liquid State Machine (LSM)



Source: Maass (2011)

Formally (CAO; PIPA, 2010), the liquid of neurons is a mapping from time-dependent inputs $\mathbf{u}(\cdot) = u\{u_i(\cdot)\}_i \in (1..k)$ lying in a subset U^k of $(\mathbb{R}^{\mathbb{R}})^k$ onto a n-dimensional dynamical liquid state $x(t)$ in $(\mathbb{R}^{\mathbb{R}})^n$:

$$L^M : U^k \rightarrow (\mathbb{R}^{\mathbb{R}})^n \tag{2.4}$$

$$u \mapsto x(t) \tag{2.5}$$

The second operation one needs to define is the readout function, mapping the liquid state into an output at every time t:

$$f^M : (\mathbb{R}^{\mathbb{R}})^n \rightarrow \mathbb{R}^{\mathbb{R}} \tag{2.6}$$

$$x(t) \mapsto y(t) \tag{2.7}$$

Thus, the liquid state machine is an operator, mapping time-varying functions onto one or many functions of time. Readout maps are generally chosen memory-less because

the liquid state $x(t)$ should contain all the information about past inputs $u(s)$, with $s \geq t$, that is required to construct the output $y(t)$. Therefore, the readout function f^M at time t does not have to map previous liquid state $x(s)$.

2.3 Networks-on-chip

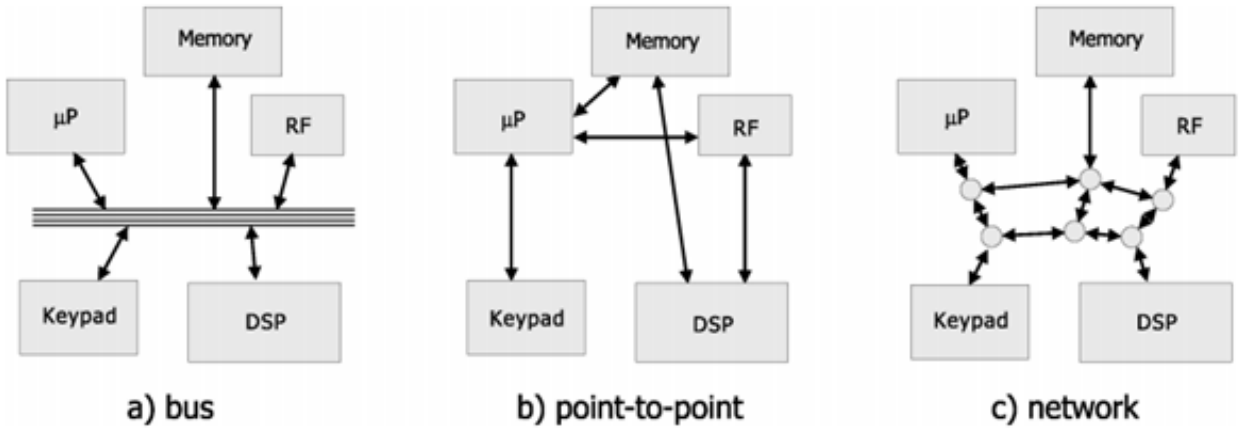
The ever increasing complexity and transistor count of nanoscale devices, where hundreds of IP cores are required to run multiple concurrent processes in a single chip, has put tremendous pressure on the communication architecture of SoCs, changing its entire design methodology to communication-based ones.

Two types of on-chip communication schemes have been traditionally used, namely, point-to-point (P2P) and bus-based communication architectures (LEE et al., 2008). P2P communication architectures can provide the utmost in communication performance at the expense of dedicated channels among all the communicating IP pairs, while they suffer from lack of scalability in terms of high complexity, cost, and design effort. On the other hand, bus-based architectures can connect a few tens of IP cores in a cost-efficient manner by reducing the design complexity and eliminating the dedicated wires required by P2P communication architectures, although they still fail to provide scalability enough when it comes to energy and performance.

The NoC-based approach represents a promising solution to the on-chip communication problems, scaling very well in terms of area, performance, power/energy consumption, and overall design effort. For this reason, scalability is said to be the utmost benefit from using NoCs. Figure 4 shows the three mentioned communication structures in a mobile phone.

Within its architecture, routing nodes are interconnected by P2P links, thus describing a network topology (NEDJAH; MOURELLE, 2014). The routing nodes are also connected to the processing elements that constitute the system, via a network interface (NI). Therefore, the information generated by a processing element, which may be divided into smaller parts, or packets, is sent over the network through the router attached to it via the NI (OGRAS; MARCULESCU, 2013). Then, the packet is stored at the input channels and the router starts servicing it. After that, the packet moves to the next router on its path and the process repeats until the packet arrives at its final destination. As a result, the communication among various cores is achieved by generating, processing and forwarding packets through the network infrastructure, rather than by routing global wires.

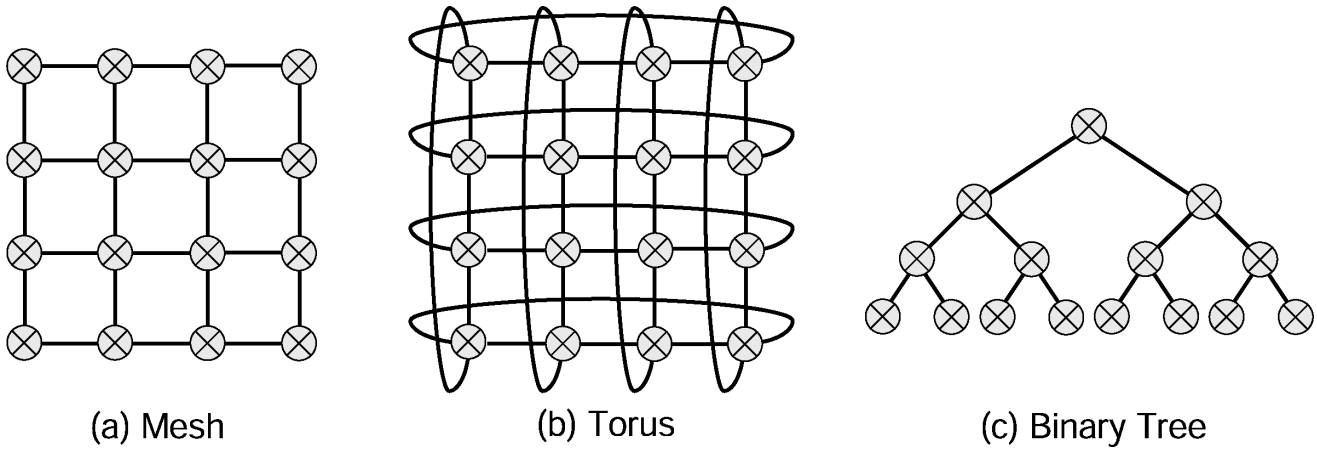
Figure 2.8: On-chip communication structures



Source: Bjerregaard and Mahadevan (2006)

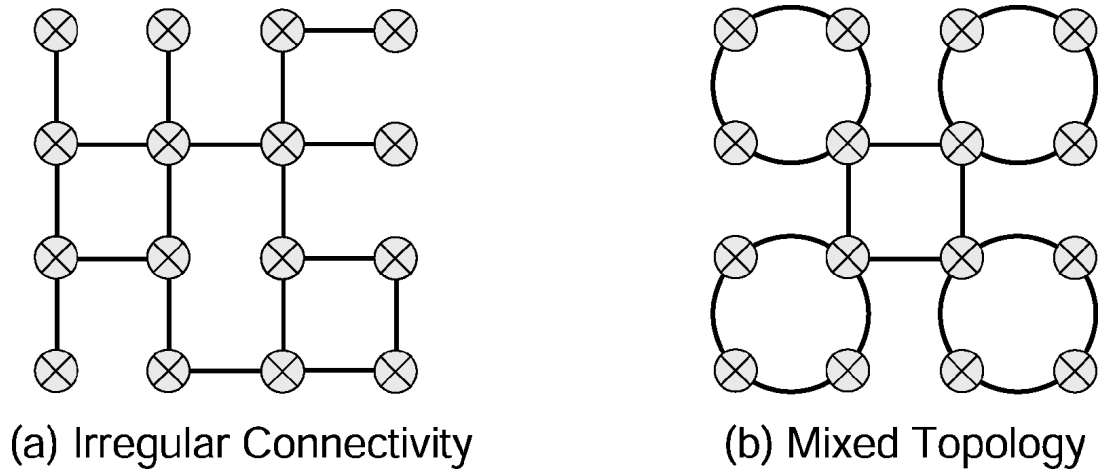
The ability of the network to efficiently disseminate information depends largely on its topology, which have a paramount effect on the network bandwidth, latency, throughput, overall area, fault-tolerance and power consumption, as well as an important role in designing the routing strategy and mapping the IP cores to the network (OGRAS; MARCULESCU, 2013). There have been various topologies for NoC architecture, including, but not limited to, mesh, torus, ring, butterfly, octagon and irregular interconnection networks (AGARWAL; SHANKAR, 2009). Figures 2.9 and 2.10 show examples of, respectively, regular and irregular forms of topologies.

Figure 2.9: Regular Forms of Topologies Scale Predictably with Regard to Area and Power. Examples are (a) 4-ary 2-cube mesh, (b) 4-ary 2-cube torus and (c) binary (2-ary) tree.



Source: Bjerregaard and Mahadevan (2006)

Figure 2.10: Irregular Forms of Topologies are Derived by Altering the Connectivity of a Regular Structure such as shown in (a) Where certain links from a mesh have been removed or by mixing different topologies such as in (b) where a ring coexists with a mesh.



Source: Bjerregaard and Mahadevan (2006)

Due to its simplicity and regularity structure, the mesh topologies are very attractive when it comes to timing closure improvement, reduction of dependency on interconnect scalability and facility to use high performance circuits. Thus, regular topologies, both one-dimensional (e.g. ring) and two-dimensional (e.g. mesh and torus) are the default choices for NoC designers (OGRAS; MARCULESCU, 2013).

For some implementations, customization might be desirable. For instance, when the size or shape of the cores varies widely, in which case regular topologies may waste area. The communication requirements of the components can also vary widely for real applications. Thus, designing the network to meet the requirements of highly communicating cores results in underutilization of other components, while designing it for the average case results in performance bottlenecks. Furthermore, a priori understanding of the communication workload can be exploited to fully customize the network topology, in case of application-specific implementations.

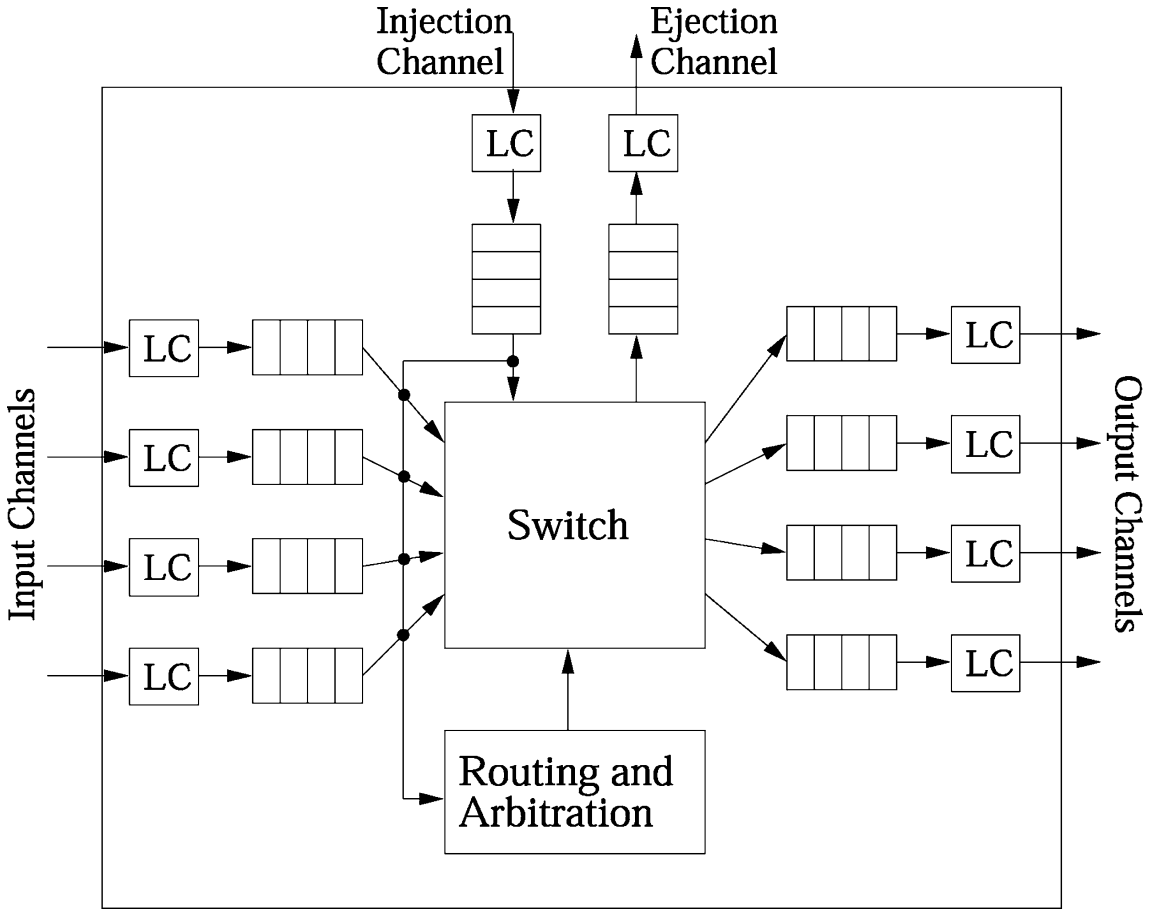
Generally speaking, the problem of optimal topology synthesis for a given application does not have a known theoretical solution. Although the synthesis of customized architectures is desirable, distorting the regular grid structure leads to various implementation issues such as complex floorplanning, uneven wire lengths, etc.

The design of the router is also a complex task in the implementation of a NoC, since it has significant impact in terms of performance, power consumption and area (OGRAS; MARCULESCU, 2013). The design involves determining the flow control

techniques, number of virtual channels, buffer organization, switch design, pipelining strategy while adhering to target clock frequency and power budgets.

The main focus in designing a router is to minimize the latency through it, while meeting bandwidth requirements. The router protocol concerns the strategy of moving data though the NoC (BJERREGAARD; MAHADEVAN, 2006). Figure 2.11 presents the major components of any routing node that is, buffers, switch, routing and arbitration unit, and link controller.

Figure 2.11: Generic Router Model. LC = link controller



Source: Bjerregaard and Mahadevan (2006)

The switch connects the input buffers to the output buffers, while the routing and arbitration unit implements the algorithm that dictates these connections. Thus, switching can be thought of as the mere transport of data, while routing can be defined as the intelligence behind it, in which the path of the data transport is determined (BJERREGAARD; MAHADEVAN, 2006).

The Routing algorithm determine the path between source and target switches for a

given packet (MELLO et al., 2004). It must prevent three situations: deadlock, livelock and starvation. The first can be defined as a cyclic dependency among nodes requiring access to a set of resources so that no forward progress can be made (MORAES et al., 2004). A livelock refers to packets circulating the network without ever making any progress towards their destination. And starvation is a condition in which a packet in a buffer requesting an output channel is blocked because the output channel is always allocated to another packet.

There is a vast parameter space when it comes to the communication infrastructure of the NoC. There may be concepts such as line and packet switching. Routing algorithms can be divided into static and adaptive algorithms as well as minimal-path and non-minimal-path ones (BLUME et al., 2008). Each of such choices might impact on all network metrics, thus being a key task of modern NoC design to efficiently explore the design space regarding all such aspects. Table 2.2 summarizes the most important parameters for a NoC communication infrastructure.

Table 2.2: Parameters for a NoC Communication Infrastructure

Parameter	**Definition/Function**
Topology	Defines the arrangement of routers and links in the form of a graph.
Routing	Determines how a message chooses a path in the graph.
Switching	Defines how and when a router input channel is connected to an output channel selected by the routing algorithm.
Flux Control	Deals with the allocation of channels and buffers for a message that traverses the graph.
Arbitration	Determines which input channel router can use a given output channel.
Memorization	Defines how and where messages blocked on a router will be stored.

Source: adapted from Zeferino (2003)

For detailed explanation of all the parameters, refer to the surveys in (BJERREGAARD; MAHADEVAN, 2006) and (OGRAS; MARCULESCU, 2013). This work will also present an exploration for all of its design choices in later sections.

The work in (NEDJAH; MOURELLE, 2014) suggests that, in general, NoCs are developed to perform a specific application. There is a variety of examples of such systems in the literature, and surveys such as (BJERREGAARD; MAHADEVAN, 2006), (MORAES et al., 2004) and (SALMINEN; KULMALA; HAMALAINEN, 2007) offer

comparison between their different design choices. Furthermore, some real CMOS application implementations reported to date are worth mentioning (HOWARD et al., 2010; VANGAL et al., 2007; CHEN et al., 2014).

3 STATE-OF-THE-ART

There is a wide range of SNNs found in the literature. Several of them were developed using hardware, as the surveys from Maguire et al. (2007), Misra and Saha (2010) and Cassidy, Georgiou and Andreou (2013) demonstrate. The next sections will present a review of some specific types of SNNs, including their main aspects, results and applications.

3.1 Non-scalable or Special Purpose Works in FPGA

The work made by Ambroise et al. (2013), for instance, developed a neural network of 117 biorealistic neurons with one single computation core, which calculates each cell in turns. Their work, which was one main point of the Brainbow European project, was different in that it took into account biological real time. They implemented a Izhikevich neuron model using very few resources of a Xilinx Virtex 4 SX55.

Another spiking neural network implementation was made by Imperial College London (THOMAS; LUK, 2009). In their work, a 1024 neuron fully-connected network was implemented using the Virtex-5 XC5VLX330T device. Their architecture was based on re-usable interconnection for storing synapse weights and calculating thalamic input, making use of the large number of available FPGA's block-RAMs and fine-grain parallelism. Having Izhikevich as the model of choice as well, their work was able to run at a clock frequency of 133MHz, 16 times faster than a 3GHz Core2 CPU, and 1.1 times faster than a single-precision 1.2GHz 30-core GPU.

The work realized by Cassidy and Andreou (2008), from the Johns Hopkins University, developed an array of dynamical digital silicon neurons implementing the Izhikevich neuron model, as an extension of their previous works, which implemented a leaky integrate-and-fire neural array. The neural array, which was implemented in a Xilinx Spartan XC3S1500 FPGA, was composed of 32 physical neurons, each with 8 virtual neurons in a single FPGA, for a total of 256 independent neurons in the system operating at 80MHz. Recently, Cassidy, Georgiou and Andreou (2013) have very much incremented their work. While the work was derived from the aforementioned implementation, the synaptic weight RAM and the re-mapper RAM were both implemented in external SRAM in order to have higher RAM capacity.

While all of the works previously analyzed implemented the Izhikevich neuron

model, there are some worth mentioning neural networks that used the Hodgkin-Huxley model. The work from Smaragdos et al. (2014), for example, chose to implement an extended HH accelerator model, since their intention was to design a biophysically-meaningful model of the Inferior Olive, a part of the olivocerebellar subsystem in the brain responsible for motor coordination and learning. Using a Virtex 7 XC7VX485T FPGA, their design was able to simulate a network of 96 neurons in real-time, a performance with sufficient speedup for use in neuroscience experiments, which was more than x700 faster than their original Matlab model and x12.5 faster than their C implementation.

Another one is the work from Bonabi et al. (2014), where the authors developed a set of techniques for efficient implementation of the HH model in FPGA. In order to accomplish such optimizations, they used computational techniques such as CORDIC (Coordinate Rotation Digital Computer) algorithm and step-by-step integration in the implementation of arithmetic circuits, whose equations were described using VHDL as the hardware description language. They also used shared resources in order to preserve details of the model, to increase the network size and to keep the network execution speed close to real time while having high precision.

Perhaps the most relevant related work for this section is the one by Zamarreno-Ramos et al. (2013). The publication, from the Instituto de Microelectrónica de Sevilla, presented a modular, scalable method for assembling hierarchically structured neuromorphic AER systems. The approach, which can be accomplished by arranging modules in a 2D mesh, was tested on single and multiple Virtex-6 FPGAs. Among the applications tested are AER-based vision processing and character recognition based on convolutional type neural networks.

3.2 Reservoir Computing in Hardware

In spite of all the advantages that a Liquid State Machine can bring to hardware implementations, to the best of the author's knowledge, there are few SNN architectures with a focus on the subject to date. Perhaps this is due to the fact that this is a relatively new field of research.

There are, however, some very interesting and worth mentioning projects, such as the work in (SCHRAUWEN et al., 2008). The authors, from the Ghent University, are very engaged researchers in Reservoir Computer, with a great number of publications in the field. In the publication, they proposed an application oriented real-time, isolated digit speech

recognition in FPGA using a Liquid State Machine. The neurons were designed using the Leaky Integrate and Fire neuron model, but compensated their undeniable simplicity by combining then with both Dirac and exponential synapse models. The results are very promising, showing that real-time speech recognition is possible on very limited or even faulty FPGAs using an LSM.

The work in (WANG; JIN; LI, 2015), from Texas A&M University, developed a general-purpose Liquid State Machine based neuromorphic processor. It was shown that the resulting implementation, which is composed by a generic pre-processor and one or multiple task processors, efficiently reuses an optimized reservoir for different tasks. Such an encouraging result was made possible by an efficient reservoir re-use by power gating, in which a certain number of neurons are powered off to operate the reservoir with the desired size.

3.3 Scalable and Large Scale General Purpose SNNs in Hardware

Among the wide range of SNNs found in the literature, the most relevant proposed architectures for this work will be presented in this section. The focus of the strategic studies was on recent highly scalable hardware neuromorphic efforts. Most of them have implemented custom large-scale computing platforms and utilize network-on-chip strategies as the main communication structure.

The Neurogrid (see the website in (NEUGRID, 2006)) is a mixed signal neuromorphic multi-chip system developed at Stanford University. It uses analog computation to emulate neural dynamics, and a digital communication scheme to support synaptic connections. The main building block is the Neurocore, a 162mm^2 chip in a 180nm process (BENJAMIN et al., 2014), which is composed of an array of 65,536 analog quadratic integrate-and-fire neurons (CHOUDHARY et al., 2012). Altogether, a Neurogrid neuron has 61 graded and 18 binary programmable parameters—common to all the neurons in that Neurocore (for example, modeling a specific cell type in a layer of cortex). The system is formed by a total of 16 chips, organized in a tree routing network, in which the long-range connections are implemented using an FPGA daughterboard and a bank of SRAMs. The neurons on any chip send messages asynchronously to synapses on other chips via multicast communication (MEROLLA et al., 2014). In total, Neurogrid simulates one million neurons in real-time for approximately 3.1W.

The BrainScaleS project (see the website in (BRAINSCALES, 2011)) at the Uni-

versity of Heidelberg (which is the follow-on of the FACETS project (FACETS, 2005)) proposes a wafer-scale system called the Hybrid Multiscale Facility (HMF), composed of analog neurons and a digital multi-layer bus communication scheme. Their main processing building block is the High Input Count Analog Neural Network (HICANN) chip, containing up to 512 adaptive exponential integrate-and-fire neuron models and 112.000 synapses in total. 352 HICANN chips are located on a single wafer, organized in reticles of eight chips each, which makes a total of $4*10^7$ synapses and up to 180k neurons in a 20cm wafer. (SCHEMMEL et al., 2010). Furthermore, the wafer is not cut into separate chips, but left uncut, and a communication layer connecting the chips is added via post-processing (SCHMIDT, 2014). Larger systems can be built by interconnecting several wafer modules, being the communication based on a 2D-torus topology, implemented on a FPGA. The resulting system is said to operate at approximately up to 105 times faster than biological neurons real-time, however, this comes at the expense of an estimated power consumption of 1 kW per wafer in a 180nm technology.

The SpiNNaker is a project at University of Manchester (see the website in (SPINNAKER, 2005)) whose goal is to provide a platform for high-performance massively parallel processing appropriate for the simulation of large-scale neural networks in real-time. For that, a microprocessor-based system was developed, each node being a System-in-Package (SiP) fabricated by UMC on a 130nm CMOS process, containing 18 ARM968 processor cores plus a 128Mbyte off-die SDRAM stacked on top of it (FURBER et al., 2013). The nodes are packaged and mounted in a 48-node hexagonal array on a PCB (Printed Circuit Board), the full system requiring 1,200 such boards. The interconnection between each node is handled by an NoC using six links wrapped into a triangular lattice; this lattice is then folded onto the surface of a toroid. In operation, the engine consumes at most 90kW of electrical power for emulating 109 neurons and 1012 synapses, being 1W per node.

The EMBRACE (HARKIN et al., 2009) is another ongoing project that aims to enable future investigation of embedded neural network applications. The project from the University of Ulster and the National University of Ireland is a mixed-signal approach to NoC-based-embedded neural information hardware. The proposed architecture proposes a scalable SNN based on a hierarchical array of NoC routers, based on a hybrid star-mesh topology (CARRILLO et al., 2013). By doing so, the project exploits data locality by creating clusters of neurons located at the bottom of the hierarchy. Unlike the SpiNNaker, the EMBRACE architecture does not consider the use of multiprocessors to emulate

the neurons but instead uses an analogue, Integrate-and-Fire neuron cell on hardware to model spiking neurons. The synthesis results for the EMBRACE, based on 65-nm CMOS technology, demonstrates a low-cost area utilization of 0.587mm^2 and power consumption of 13.16mW for a single cluster facility of 400 neurons (CARRILLO et al., 2012). The project also offers a hardware acceleration relatively to the biological real-time, being able to accommodate firing frequencies of up to 174kHz.

The TrueNorth, which is the achievement of the Defense Advanced Research Projects Agency (DARPA) SyNAPSE project, is a chip from IBM and its university partners composed of 4096 neurosynaptic cores tiled in a 2-D array, containing an aggregate of 1 million digital neurons and 256 million synapses (AKOPYAN et al., 2015). The TrueNorth architecture uses hierarchical communication, with a high-fanout crossbar for local communication and a network-on-chip for long-distance communication, and global system synchronization to ensure real-time operation. The chip, fabricated using Samsung's 28-nanometer process technology, operates at real-time with an impressive low typical power consumption of 56mW. It also attains a peak computational performance of 58 giga-synaptic operations per second (GSOPS) and computational energy efficiency of 400 GSOPS per Watt (GSOPS/W).

The Reconfigurable On-line Learning Spiking Neuromorphic Processor (ROLLS neuromorphic processor) is a full-custom mixed-signal VLSI device with learning circuits that emulate the biophysics of real spiking neurons and dynamic synapses (QIAO et al., 2015). Idealized by the Professor Giacomo Indiveri's group at the Institute of Neuroinformatics, University of Zurich and ETH Zurich, the chip was developed for exploring the properties of computational neuroscience models and for building brain-inspired computing systems. Having approximately 12.2 million transistors, the chip was fabricated using standard 180 nm Complementary Metal-Oxide-Semiconductor (CMOS) 1P6M process occupies an area of 51.4 mm^2.

By comparing all the above projects, the reader is able to identify some similarities and major differences among them. The BrainScaleS, Neurogrid and EMBRACE projects all implemented different adaptations of integrate-and-fire analog neuron models, while both the TrueNorth and SpiNNaker preferred to work with digital neurons. All the designs implemented some kinds of on and off-chip networks, although the topologies greatly differ. Although those are examples of design choices that significantly affect the final results, the similarities among some designs were not enough to produce similar design specs, with even the quantity of neurons and synapses per chip being very different for

expectedly similar cases. Consequently, it is not difficult to infer that several design choices have yet to be explored and combined to others in order to better understand their impacts on the systems as well as to determine the optimal cases for each project.

Table 3.1 shows a summary of some relevant data from the aforementioned neuromorphic architectures. Note that although related, most measurements are not from the same benchmark nor use the same technology, so they cannot be compared directly.

Table 3.1: Related Work synthesis data

Project Name	Neuron Model	A/D	Neurons	Power	Power/ Neuron	Size mm^2	Area/ Neuron	CMOS Process	Synapses per Neuron	Topology
SpiNNaker	IF or IZHI	DS	$16x10^3$	1W	6,25E-05	102	6,38E-03	130nm	$1x10^3$	Triangular Lattice*
Neurogrid	Quad. IF	A	983,040	5W	5,08E-06	168	1,71E-04	180nm	$6x10^9$	Star
BrainScaleS	Exp. IF	A	512	1kW	1,95	430	8,40E-01	180nm	$112x10^3$	Hierarchical Buses**
EMBRACE	IF	A	400	13.16mW	3,29E-05	0.587	1,47E-03	65nm	400	Hybrid (star-mesh)
TrueNorth	-	-	$1x10^6$	65mW	5,60E-08	430	4,30E-04	28nm	256	Mesh
ROLLS	Exp. IF	A	256	4mW	1,56E-05	51.4	2,01E-01	180nm	256	Programmable Switch-matrix

A = Analog, D = Digital, DS = Digital (Software), IF = Integrate-and-Fire, IZHI = Izhikevich. *folded into a toroid surface. **2D Torus (wafer). SpiNNaker, BrainScaleS, TrueNorth data per chip; EMBRACE data per cluster.

Source: The author

4 PROPOSED NEUROMORPHIC ARCHITECTURE

In common with most engineering projects, the development of the proposed neuromorphic architecture required a careful consideration of many trade-offs and compromises. This chapter presents an overview of the Digital HierArchical Neuromorphic Architecture (DHyANA), including the description of its main building blocks, a careful explanation of the system's behavior and a discussion about the main trade-offs and design choices that made possible the realization of this project.

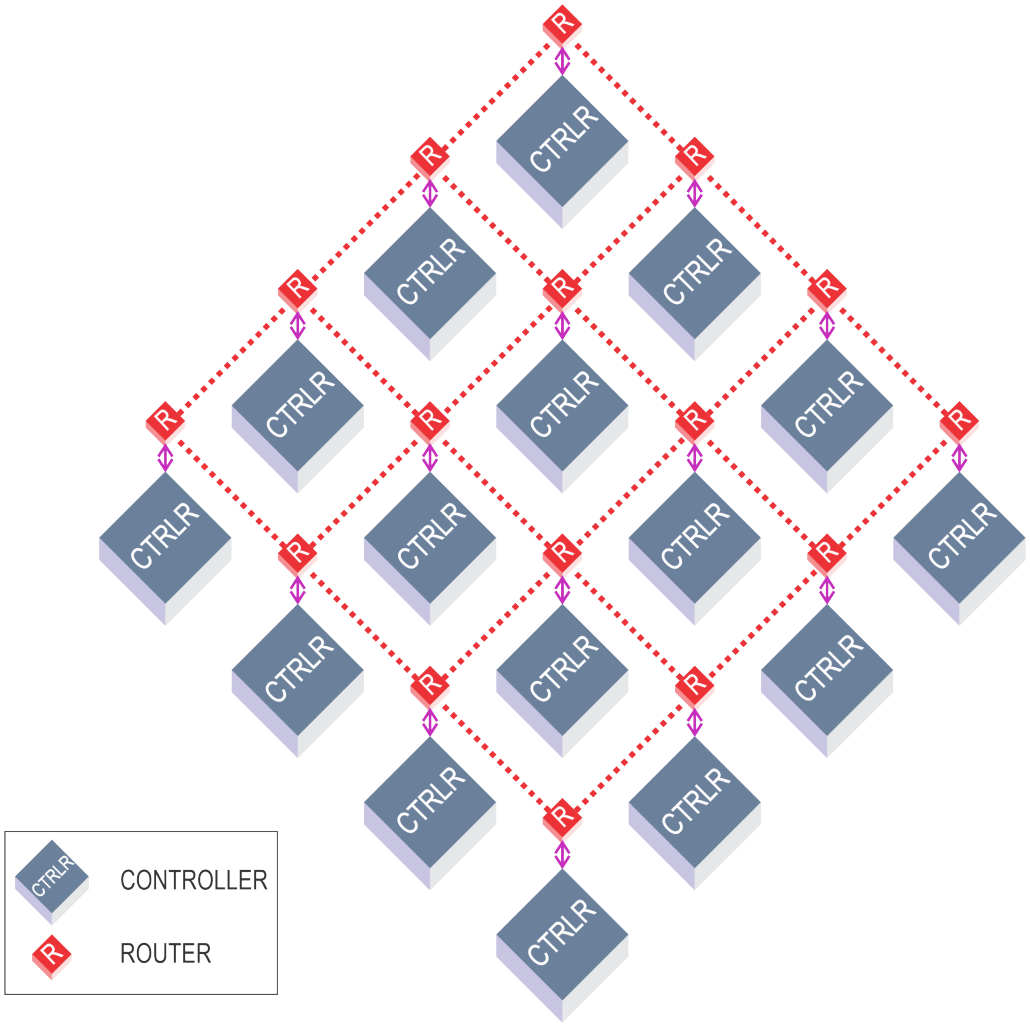
DHyANA is a Neural Network composed of a two-level hierarchical network-on-chip. The processing elements, digital Izhikevich neuron models, as well as a Controller are locally connected through a shared bus at the Cluster Level. The interface between the two levels is achieved through a connection between the Controller and its corresponding Router. Finally, at the Global level, a 2D Mesh NoC performs the connection between clusters of neurons. A system's view at the Global Level is available at Figure 4.1. It shows the system prototype size for the global communication, a 4x4 mesh topology.

The motivation for choosing bus and mesh, two well-known topologies, as communication levels of the hierarchy was driven by practicality, performance and energy optimizations. Bus-based designs have been well architected for small multichip processors, within the order of very few tens of IP cores, being quite simple and power efficient. Nevertheless, as mentioned in section 2.3, such topology does not scale for larger configurations. Thus, a bus-based topology was considered very well suited to support local communication, while the choice for a planar, low complexity mesh topology was considered appropriate for the global network.

The hybrid topology choice was also conducted in order to optimize the system for locality in communication. In terms of chip multiprocessors, the concept of locality was defined in (DAS et al., 2009) as "the percentage of packets injected by a node that are satisfied by its immediate neighbors in the network". Thus, applications with high locality tend to have nearest neighbor communication pattern with high local traffic. In neuronal biologic systems, the concept is somehow similar. On the scale of roughly a cubic millimeter of cortex, containing about 100,000 neurons, the work by Mehring et al. (2003) suggests that the connection probability decreases in a Gaussian fashion with the distance between neurons. The idea of locality in such systems was also suggested in other studies (KUMAR; ROTTER; AERTSEN, 2010; STROGATZ, 2001).

Based on the aforementioned locality premises, one can infer that the inter-cluster

Figure 4.1: DHyANA Global Level. For the sake of visual clarity, the Cluster Level is omitted, as if under the Controllers.



Source: The author

communication is supposed to have lower activity during execution compared with intra-cluster communication, being the shared bus a convenient choice for such configuration. However, these fewer messages demand high throughput to handle the burst of spikes, and a mesh-based NoC can provide sufficient throughput, avoiding bottlenecks in an unexpected inter-cluster communication, independently of message rate. In summary, the choice for a hybrid topology allows exploration of faster local connections, while still maintaining great scalability and reliability for the global level.

The following sections will describe the two levels of the hierarchy, as well as the system's processing elements.

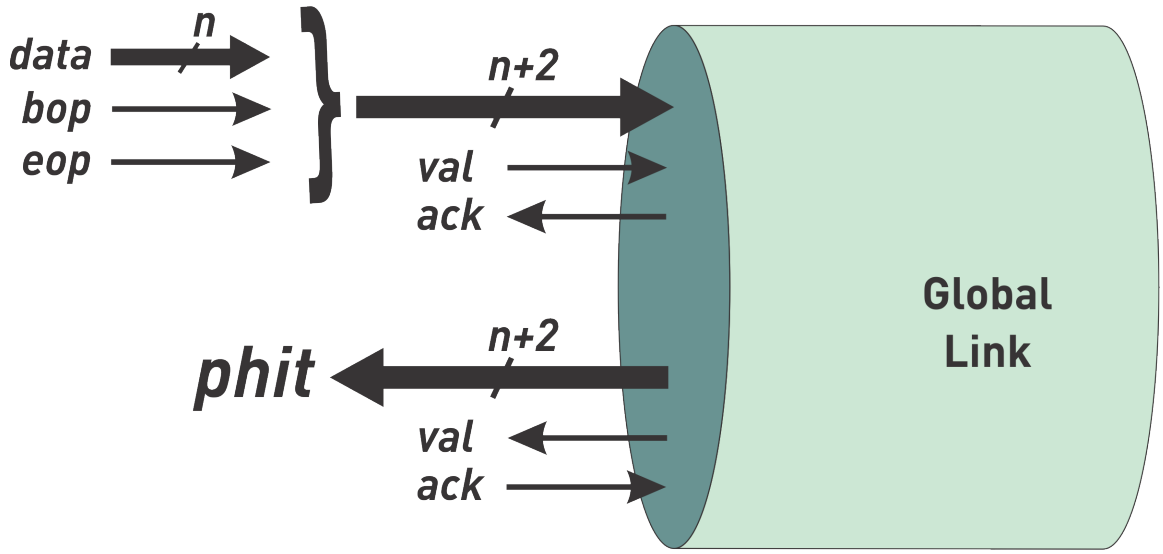
4.1 Global Level: The Mesh NoC Interface

The mesh NoC used for DHyANA's Global Level was the System-on-Chip Interconnection Network (SoCIN) (REINBRECHT, 2012; ZEFERINO, 2003), a scalable network based on a parametric router architecture for customized low cost NoCs. The network, which is based on the XY routing algorithm, employs wormhole packet switching and handshake flow control to assure the deadlock freedom at a low cost.

It is depicted at Figure 4.1, where the red boxes represent the Global Routers, while the blue boxes represent the Cluster Controllers (which are interconnected to the bus at the Cluster Level). The dotted traces in red between routers represent communication links for its XY routing. The purple arcs represent the terminal link for connecting to the corresponding Cluster Controllers. From the Router's perspective, the purple terminal links are denominated Local Ports, which constitute the interface between communication levels.

The DHyANA Global link (seen in Figure 4.2) is composed of two simplex unidirectional opposite channels, each one with its data, framing and flow control signals (ZEFERINO; SUSIN, 2003).

Figure 4.2: Global Link



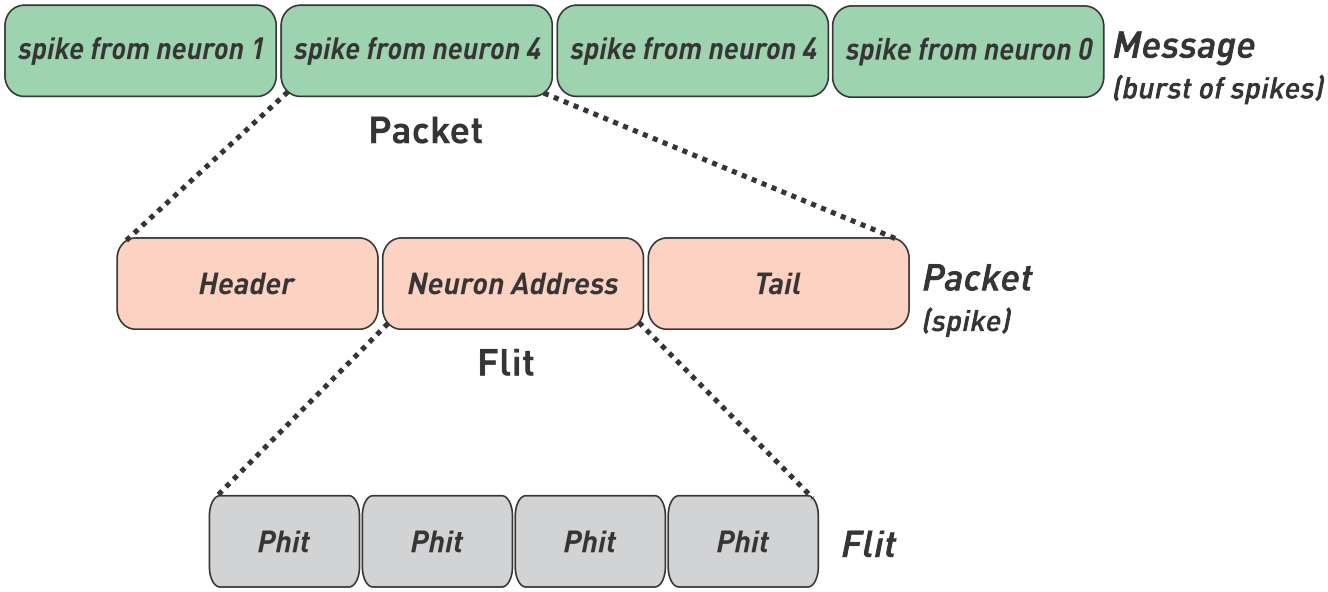
Source: Adapted from Zeferino and Susin (2003)

Each channel consists of n bits for data and two bits for packet framing, so the value of the physical width of the channel (phit) is $n + 2$. In this work $n = 8$, thus the phit is 10-bit wide. The two-bit sideband constitutes the bop (begin-of-packet), which is set

only at the packet header; and the eop (end-of-packet), only set at the last payload word, the packet tail. In addition, each channel includes a pair of signals required to control flow, which are used to validate data at the channel (val) and to acknowledge the received data (ack).

The network uses wormhole switching (DALLY; SEITZ, 1986), a model which allows the building of small and fast routers, being the preferred switching approach in interconnection networks for parallel computers and in NoCs. Each spike is sent by means of a packet, which is composed by flits (flow control units). A flit, as explained more didactically through Figure 4.3, is the smaller unit over which the flow control is performed. In this work, a flit equals the physical channel word, which is the size of a phit (physical unit), n + 2 bits.

Figure 4.3: Message Composition



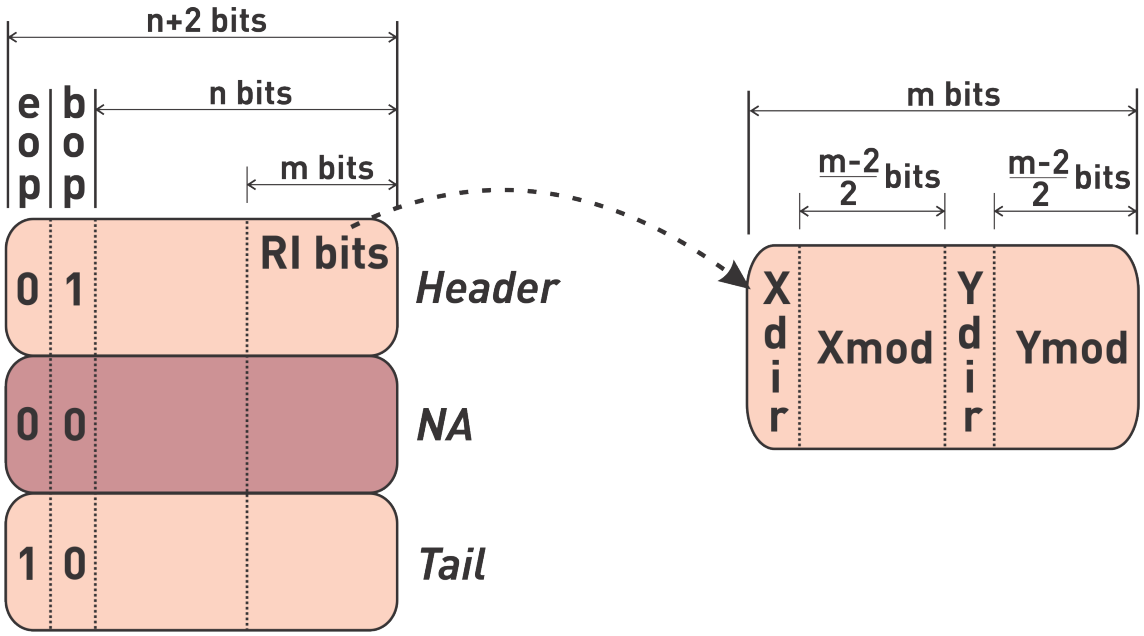
Source: Adapted from Reinbrecht (2012)

The packets in the Global Level are composed by three flits: the Header; the Neuron Address (NA); and the Tail (as seen in Figure 4.3). The Header is the first flit of the package and includes the information needed to establish the path of the packet on the network. The remaining flits include the information to be transferred by the package, and follow the Header flit by the network in a pipelined fashion. The NA contains the spiking neuron location within its Cluster. The Tail, the flit which sets the end of the package, is also used to recover the address of the Global Router who originated the spike. Thus, as will be further explained later in this work, by combining the Neuron Address and the Tail data,

the destination Cluster Controller is able to decode the information needed for establishing the synapses.

The packet format is described in more detail in Figure 4.4. As previously stated, each flit is $(n + 2)$-bit wide, which, for this project, results in $8 + 2 = 10$ bits. The (n)th bit represents the bop marker and the $(n + 1)$th bit denotes the eop marker. The m least significant bits of the Header flit are reserved to the Routing Information (RI) bits, which includes the information needed for packet routing. It is composed of four fields: Xdir, Xmod, Ydir and Ymod. The description of each field is included in Table 4.1.

Figure 4.4: Global Packet format and RI bits. In this work, $n = m = 8$ bits.



Source: Adapted from Zeferino and Susin (2003)

Table 4.1: Description of RI bits

Field	Meaning
Xdir	The packet must be routed in the East/West direction
Xmod	Number of links remaining to be traversed on X (East/West) direction
Ydir	The packet must be routed in the North/South direction
Ymod	Number of links remaining to be traversed on Y (North/South) direction

Source: Adapted from Zeferino (2003)

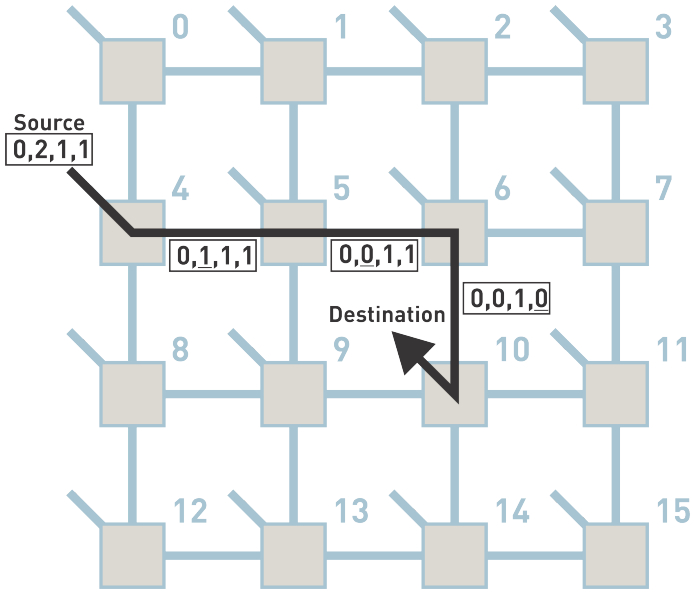
The Global routing scheme is the XY-routing, a dimension-ordered approach in

which a packet going from a source to a destination must first travel in the X direction and, when it reaches the column of the destination, follow to the target terminal in the Y direction (ZEFERINO; SUSIN, 2003). Consequently, at a router, after the schedule of an output channel, the routing circuits must update the RI bits, by decrementing by one the mod field related to the direction in which the packet is being routed (X or Y). When a router receives a packet with X mod null, it means that the packet must be routed in the Y direction (North or South, depending on Ydir value). However, if Y mod is also null, the packet is delivered to the Cluster attached to the Local Port (purple arcs at Figure 4.1).

The routing is also deterministic and source-based, hence each Cluster Controller must determine the path to be used by a packet and include the corresponding RI in the packet header. Given the sender-receiver pair, the determinism implies that always the same path will be used to the packets following from the sender to the receiver.

An example of the routing process is seen in Figure 4.5. A package, whose destination is one or more neurons at Cluster 10, is injected into the network by the Cluster Controller 4, connected to the Global Router 4. Note that the Clusters were suppressed at the picture for visual simplicity. The connection to the Clusters is represented by the diagonal lines. The routing information for the network in the packet at Router 4 is equal to "0,2,1,1", which indicate that the packet should move two links East and then one link South.

Figure 4.5: Packet Routing



Source	Destination	Xdir, Xmod, Ydir, Ymod
4	10	0, 2, 1, 1

Source: Adapted from Zeferino (2003)

The example of Figure 4.5 shows that, upon receiving a packet, the router evaluates the Xmod field. Then, after identifying that Xmod is not null, it evaluates the Xdir field to determine if the packet is forwarded to the East or the West. In the example, since Xdir is 0 at the source, Router 4, the packet is forwarded East and the Xmod field is decremented from 2 to 1. In fact, it is repeatedly decremented while passing each Router, until it reaches Router 6. Once Xmod reached zero, the packet will not be forwarded in the X direction anymore. Consequently, as soon as it reaches the Router 6, it checks the value of Ymod, which in the example is 1. Afterwards, the Router 6 evaluates Ydir and determines that since this is 1, the packet is forwarded South right after the value of Ymod is decremented. Finally, once Router 10 receives the packet, it verifies that both Xmod and Ymod are null. Thus, the router delivers the packet to the Cluster connected to it, through the Local Port.

The architecture of the Global Router will be further described in the next subsection, as well as an overview on its Arbitration, Flow Control and Buffering.

4.1.1 The Global Router

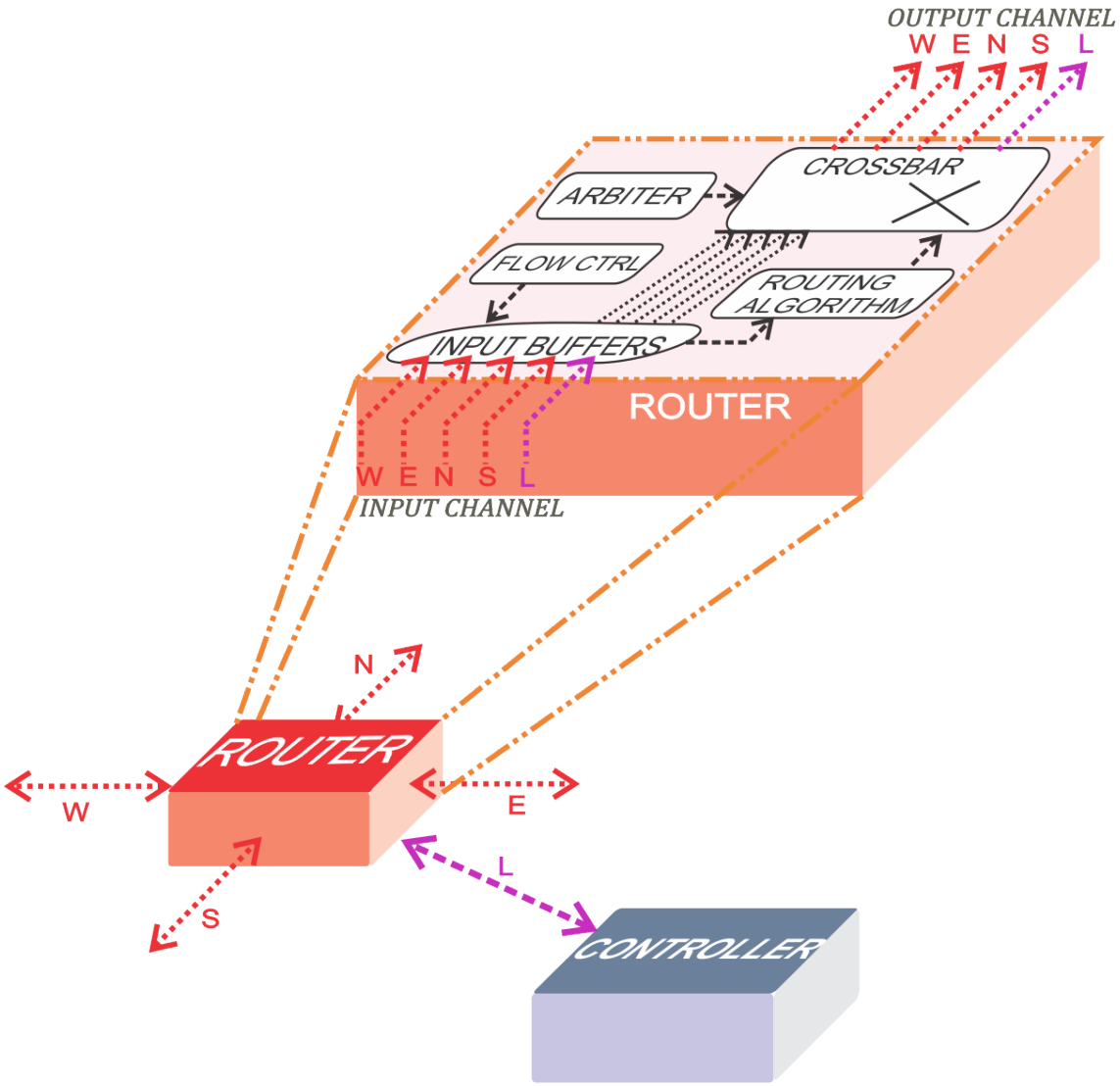
The basic building block of the Global Level is the Router Architecture for Systems-on-Chip (RASoC) proposed in (REINBRECHT, 2012; ZEFERINO, 2003), a soft-core configurable in three dimensions: the width of the communication channels, the buffers depth and the width of the routing information in the packet header. It contains up to five bi-directional ports, all compatible with the link shown in Figure 4.2: four to connect to the neighbor routers (North, East, South and West ports) and the Local port to connect to the Cluster Controller (ZEFERINO; KREUTZ; SUSIN, 2004). Figure 4.6 shows a detailed representation of its main components.

Each of the router's input ports have First-in First-out (FIFO) buffers to handle high throughput messages. The buffers implement partitions between the input channels to store packet flits, which are read in the same order in which they are written.

Moreover, the network uses a dynamic and distributed approach for arbitration. By using an exhaustive Round-Robin arbiter at each requested output channel of a router, it ensures fairness and that no packet is permanently blocked for having lower priority on the network (starvation).

The flow control is based on handshake, a simple and cheap strategy in which the sender informs its intention to send data to the receiver via a validation line; and then the receiver confirms the availability to receive data via an acknowledgment line. An

Figure 4.6: Global Level: Router components.



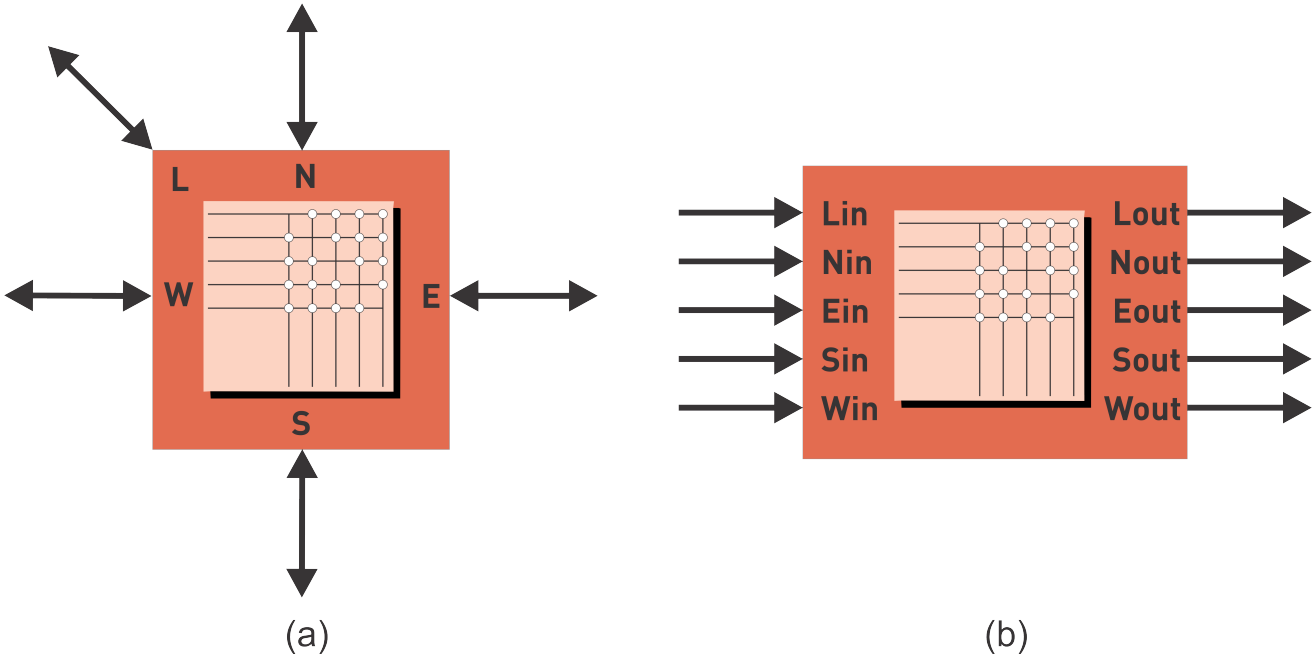
Source: The author

acknowledgment is given under the condition that there is a validation in the channel and the receiver input buffer is not full. Indeed, it has been shown in Figure 4.4 that the Global link has two signals for control flow: val and ack. The two signals are connected to the control circuits at each end of the channel so as to implement the handshake protocol, which regulates the data flow in the channel. This very conservative mechanism ensures that no flit is discarded, since the transmission is performed only after an agreement between the sender and receiver.

The crossbar module is composed by a data crossbar and two control crossbar, which perform the switching of data and internal flow control signals, respectively. Each

crossbar is composed by five 4x1 switches as seen in Figure 4.7.

Figure 4.7: Global Router Interface and Crosspoint Matrix. (a) bi-directional ports (b) alternative representation, representing the unidirectional channels



Source: Adapted from Zeferino (2003)

Note that the crosspoint matrix shown in Figure 4.7 does not implement the main diagonal because a packet incoming the input channel of a port cannot be forwarded to the output channel of the same port. For each column in the crosspoint matrix, there exists a data switch in an output channel instance. For example, the first column is related with an (n+2)-bit 4x1 data switch at Lout channel, which selects an input channel among Nin, Ein, Sin, and Win.

As previously mentioned, the Local Port terminal link allows the connection of a Global Router with its corresponding Cluster Controller, thus describing the communication between hierarchy levels. The Cluster Level will be introduced in the next section. The Cluster Controller will be further described in Chapter 5.

4.2 Cluster Level: The Bus Interface

According to the Cambridge Dictionary, a cluster is “a group of similar things that are close together, sometimes surrounding something” (CAMBRIDGE DICTIONARY, 2016). In a computational system, for (LENG et al., 2005), such groups must be organized by computation complexity, communication requirement and functional relationship of IP cores.

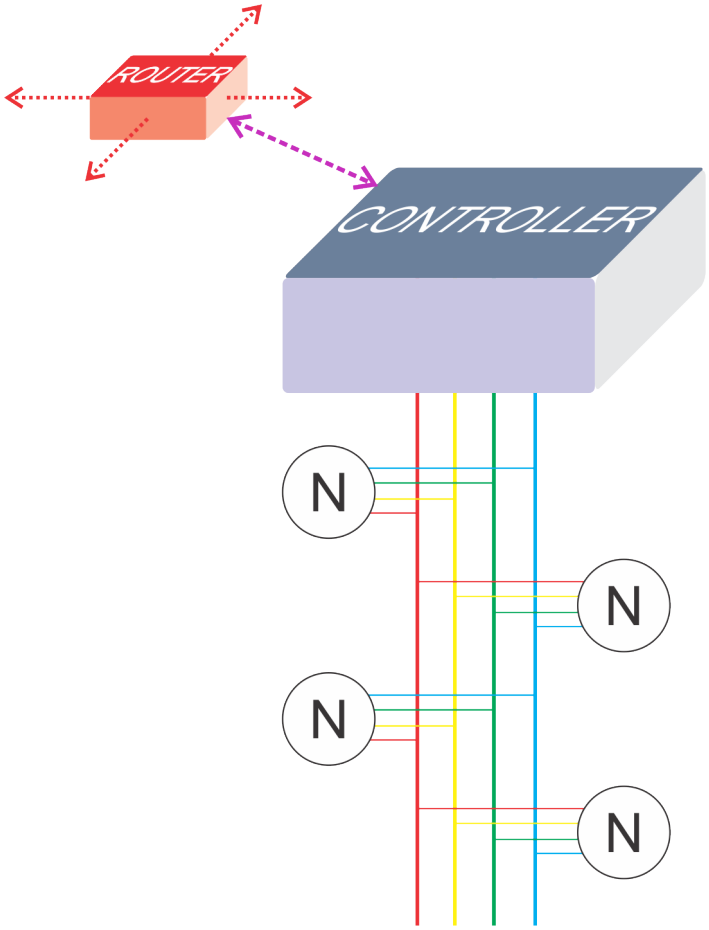
This work proposes a so-called Cluster Level as the lowest level of the DHyANA communication hierarchy, following the premises of locality mentioned in the beginning of this Chapter. The neurons are connected to each other by a shared bus, a low cost solution which permits multicast messages (one neuron sends to many) and low latency.

Although the crossbar from the Hierarchical Crossbar-based Interconnection Topology (HICIT) in (REINBRECHT, 2012) would also be a good choice, it would come with a huge power penalty. Thus the choice for a bus in lieu of a crossbar is justified by its high bandwidth, low latency and low power since the switching energy will be greatly reduced, as explained in (DAS et al., 2009). However, such choice comes with the cost of low Cluster scalability, thus if future applications require bigger Clusters, the Cluster Level may change to the HICIT crossbar-based solution, or even a hybrid in which a new Level of communication is incorporated.

As seen in Figure 4.8, the Cluster Level consists of three main parts: the Bus system, the Cluster controller and the Neurons.

The Cluster Level is responsible for two very important tasks within the DHyANA architecture: connecting the neurons within the same block, as well as building the packets for communication between levels of hierarchy. It also has two critical, performance-limiting steps which occurs during spike generation, when it performs two very memory intense operations: 1) the Cluster Controller has to access a ROM in order to check for inter-cluster connections; and 2) the synapses are verified by an access to a CAM memory at each Neuron Cell. For being a critical implementation as well as the main contribution of this work, a careful explanation of the Cluster Level will be realized in the next chapter. However, before that, the next section will address the processing unit of the neural network: the mathematical neuron model.

Figure 4.8: DHyANA Cluster Level. For the sake of simplificy, only four neurons are depicted.



Source: The author

4.3 Processing Element: The Neuron

DHyANA uses a digital, low latency Izhikevich Simple Model (ISM) on hardware, proposed in (BANDEIRA et al., 2015), for the spiking neurons.

The choice for the neuron model proposed by Izhikevich was mostly due to the fact that it stands in the middle ground between complexity and biological plausibility (as seen in section 2.1.2). This choice, however, comes without loss of generality. Thus, whenever another neuron model is found to be more suitable for a certain application, it can be incorporated to the system.

In order to make its implementation viable in terms of digital logic, some changes in the ISM equations 2.1 to 2.3 were made, following the approach described in (CASSIDY; ANDREOU, 2008). The new equations, which are ensured to have the same behavior as the previous equations, are as follows:

$$h\frac{dv}{dt} = \frac{1}{32}v^2 + 3.90625v + 109.375 - u^* + I^* \tag{4.1}$$

$$h\frac{du^*}{dt} = a^*(b^*u - v) \tag{4.2}$$

$$v \geq 30mV \Rightarrow \begin{cases} v \leftarrow c \\ u^* \leftarrow u^* + d \end{cases} \tag{4.3}$$

where $h = 0.78125$, $u^* = hu$, and $I^* = hI$; the parameters *a*, *b*, and *d* are replaced by *a**, *b**, and *d**, respectively, each one also multiplied by *h*. The transformation is suggested in (CASSIDY; ANDREOU, 2008) and (AMBROISE et al., 2013), although both disregard the factor *h*.

The Euler's Method was used to solve such equations, producing accurate results. The approach outcomes were the following:

$$v_{n+1} = v_n + \Delta t \left[\frac{1}{32}v_n^2 + 3.90625v_n + 109.375 - u_n^* + I_n^*\right] = v_n + \Delta t.kv \tag{4.4}$$

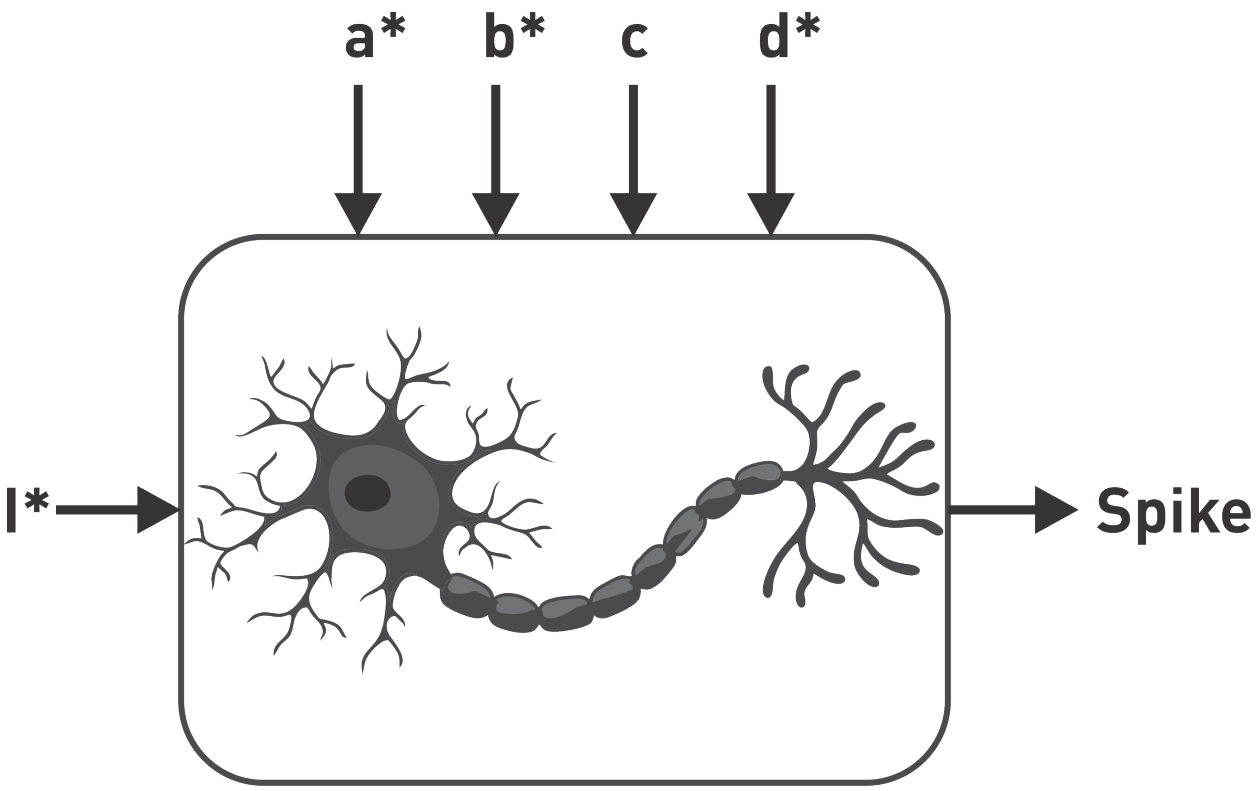
$$u_{n+1}^* = u_n^* + \Delta t \left[a^*(b^*v_n - u_n^*)\right] = u_n^* + \Delta t.ku \tag{4.5}$$

where Δt is the time increment of the Euler's Method.

Additionally, it was considered that $3.90625v \approx 4v$, as in (CASSIDY; ANDREOU, 2008). *kv* and *ku* are used further in the implementation and they represent the variation for each iteration.

A single neuron is represented in Figure 4.9. Its circuitry was intended to be as parallel as possible, optimized for latency rather than the area with no reuse of any adder or multiplier.

Figure 4.9: The Neuron Implementation



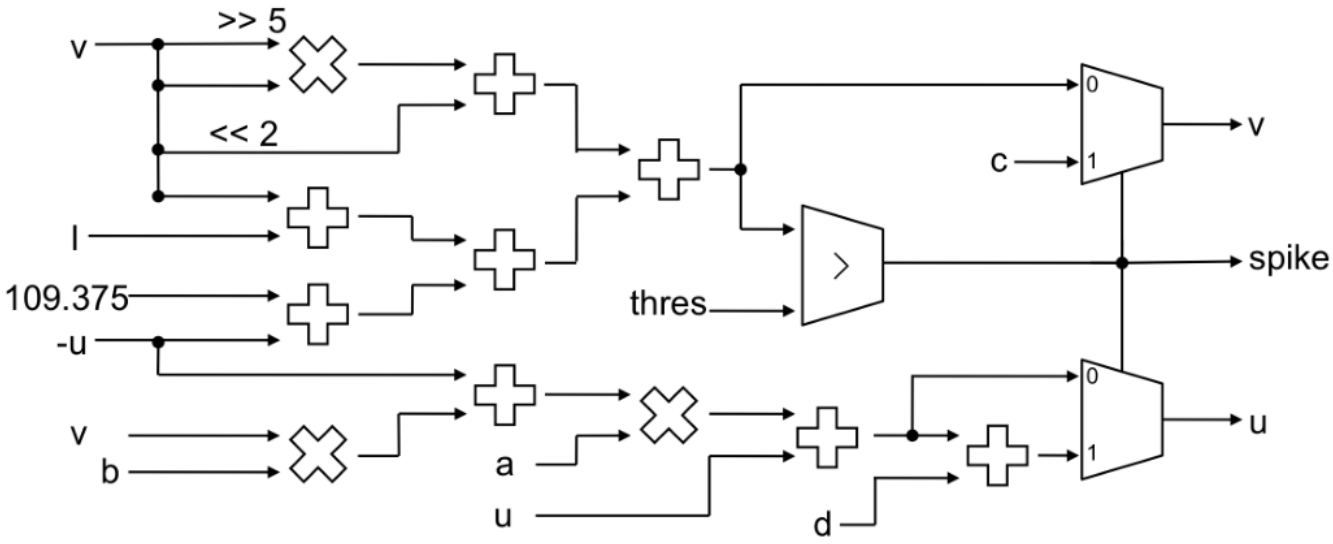
Source: The author

A 18-bit fixed point two's complement representation was chosen, as it is a better suited approach for digital implementations than floating point. Furthermore, as exposed in (AMBROISE et al., 2013), the quantity of 18 bits is an efficient choice as it allows usage of the available hardware without compromising the accuracy.

Figure 4.10 represents the computation for the new value of *v, u**, as well as the activity log. The next *v and u** is calculated in two parallel operations, one for the case with a spike and other without a spike. The selection between the two is performed afterwards.

Each neuron receives the parameters (*a*, b*, c, d**) from a top module and stores locally the initial values for *v and u**. The initial values of *v* and *u** are, respectively -70 mV and -15.63 mV. The time incremental is $\Delta t = h = 0.78125$ ms (milliseconds). The parameters and injected currents are displayed in Table 4.2.

Figure 4.10: The Neuron Schematic



Source: Bandeira et al. (2015)

Table 4.2: Parameters for each neurocomputational feature and injected current used in the implementation

		Neural Behaviour					
		Tonic spiking	Phasic spiking	Tonic bursting	Phasic bursting	Mixed mode	Spike-frequency adaptation
Input	I*	10.9375	5	11.71875	4.6875	9.375	23.4375
Parameters	a*	0.015625	0.015625	0.015625	0.015625	0.015625	0.0078125
	b*	0.15625	0.1953125	0.15625	0.1953125	0.1953125	0.1953125
	c	-65	-65	-50	-55	-55	-65
	d*	4.6875	4.6875	1.56125	0.0390625	3.125	6.25

Source: Bandeira et al. (2015)

The parameters in Table 2.2 are adapted from the original publication (IZHIKEVICH, 2003) considering the factor *h*, and it depends on the type of the simulated neuron.

5 CLUSTER LEVEL

This section presents the main contribution of this work, DHyANA's Cluster Level. The main building blocks are delineated, incorporating details regarding its implementation, the circuit schematics and the explanation of its behavior.

The Cluster Level is mainly composed, as seen in Figures 4.8 and 5.1, of the Neuron modules and the Cluster Controller. The interface with the Global Level is possible due to the packet communication made between the Cluster Controller and the Global Router.

5.1 Neuron module

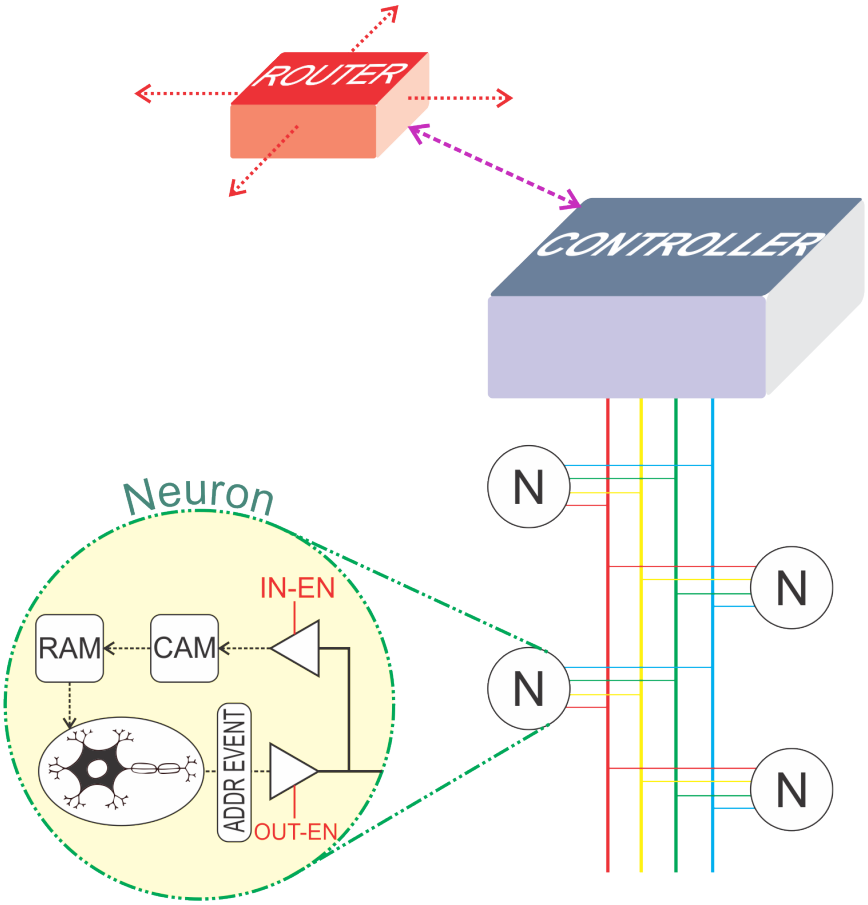
Besides the Izhikevich Simple Model (ISM) module, which was designed to run at a lower clock speed then the rest of the system, each Neuron module is mainly composed by an Address-Event (AE) Encoder module and two memories as the synapses, as observed in Figure 5.1.

The ISM inputs indicated in Figure 4.10 are implemented as 18-bit constants, with their values set offline. This is true for all except for the current I signal, which is connected to the output of the RAM memory and will be explained later in this section. The only relevant output for this implementation is the 1-bit *spike* signal, which indicates the occurrence of a spike at the neuron.

After a spike happens, the AE module is responsible for sending the Neuron Address to the shared bus. It uses the so-called Address Event Representation (AER) approach (MAHOWALD, 1992; SIVILOTTI, 1991), in which an address is assigned to each neuron cell in a chip, so that when a cell activity occurs it is broadcast to all computational nodes within a defined region.

Such approach, which has been used for some time by the neuromorphic community, solves real-time neuron communication requirements by making the event duration short enough (approximately 1 microsecond) compared with the width of neural spikes (approximately 1 microsecond), making it less likely for two events to overlap (DEISS et al., 1998). Furthermore, even if events from several nodes did occur simultaneously, the limited maximum firing rate and a manageable arrangement of close succession of signals could bring the loss of information in rate coding to a minimum. Much cortical action potential processing has a temporal resolution in the order of a millisecond or longer (SINGER, 2008; SHADLEN; NEWSOME, 1994; DECHARMS; ZADOR, 2000), whereas

Figure 5.1: Neurons at the Cluster Level



Source: The author

the maximum time-skew introduced by queuing of address events is much shorter — of the order of 0.1 milliseconds (DEISS et al., 1998).

In order to further avoid loss of information, a simple handshake protocol was implemented, in which the Bus Control module at the Cluster Controller handles the permissions with a priority encoder. Using the input- and output enable signals, the module controls the buffers at the Neuron modules, preventing simultaneous information to reach the shared bus.

At the Neuron module, a simple FSM is also implemented in order to follow and control the protocol at each module. Furthermore, as the ISM output spike signal comes from a slower clock domain, the FSM is also important to guarantee that the spike flag signal is correctly handled. The FSM at the Neuron module thus keeps track of each spike from the ISM, assigns the address to each specific neuron according to its position within the system, and also controls the status of the communication.

For this work's prototype network, the neuron address is defined by the following premises: each neuron module is assigned an 8-bit address, in which the 4 least significant bits represent the neuron position within each Cluster and the 4 most significant bits represent the Cluster position within the system (see Figure 5.2). The Cluster addresses at the system were assigned by VHDL generics mapped into Verilog parameters, as the system was designed in a mixed-language approach.

Figure 5.2: Cluster Level Packet, containing the Neuron and Cluster Addresses.

Source: The author

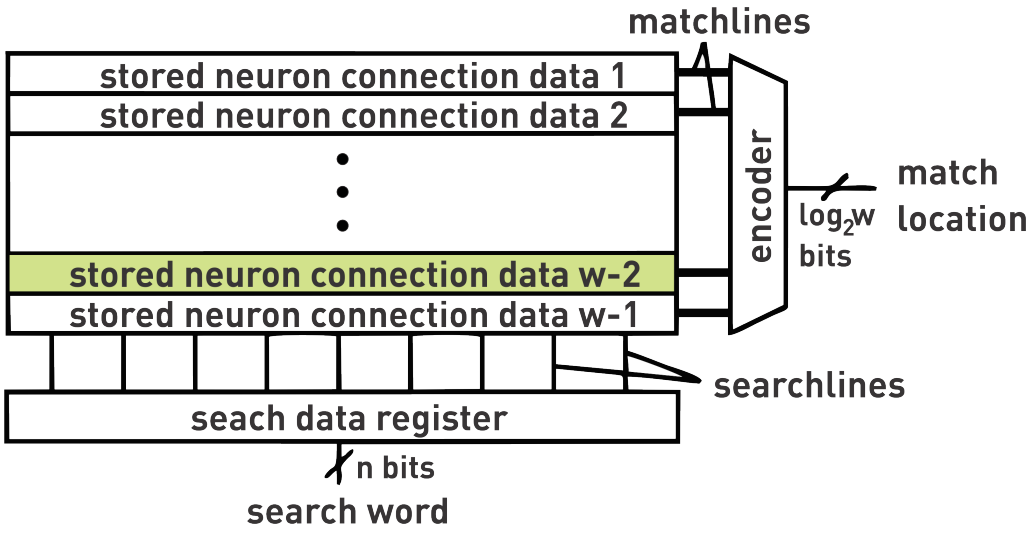
The two memory cells at the Neuron module are a Content-Addressable Memory (CAM) and a Random-Access Memory (RAM). Both memories were initially described in Verilog by adapting the Altera's Recommended HDL Coding Styles guide in order to ensure optimal synthesis results for our target Altera FPGA device, as well as to have a good testbench structure.

The two memories mainly differ in the type of information that is handled at the input/output. At the RAM, the inputs are address locations of its contents and the outputs are the contents of that addresses. The input of the CAM, oppositely, is a data to be searched at the memory contents, so that when there is a match, its respective address will be available at the output.

In order to provide such mechanism, the CAM memory has a dedicated comparison circuitry, in which a fast lookup-table function is implemented. A simplified block diagram of the CAM is shown in Figure 5.3.

The input of the system is the search word, which in this project is the address of a neuron that spiked, the 8-bit word set according to Figure 5.2. Each stored word has a matchline that indicates whether the search word has an identical stored word, configuring a match case. The matchlines are then fed to an encoder, which generates a binary match location corresponding to the matchline that is in the match state. Thus, in the occurrence of a match case, two signals are output: the memory address that contains the matching *neuron connection data* word, and a hit signal (not shown in the figure) is set, which is

Figure 5.3: Conceptual view of a Content-Addressable Memory, containing w words. In the example, the search word matches location (w-2) as indicated by the colored box. The matchlines provide the row match results. The encoder outputs an encoded version of the match location using $\log_2 w$ bits.

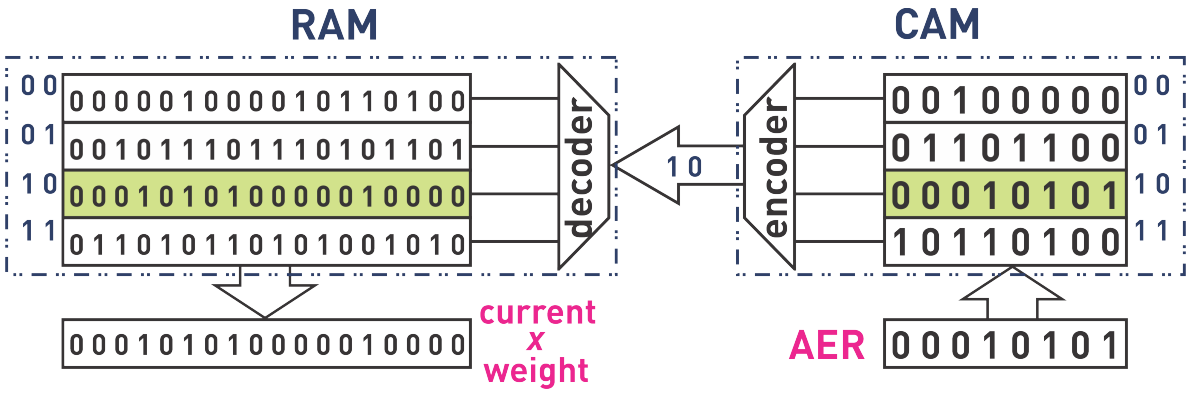


Source: Adapted from Pagiamtzis and Sheikholeslami (2006)

then used for preparing the Neuron module to receive the 18-bit data from the RAM.

The two memories, therefore, represent the synapses in the following manner: the CAM memory tests connectivity of an AE being broadcast at the Bus; in positive cases, it sends to the RAM the address at which is stored the synapse *current x weight* for that particular connection, as visualized in Figure 5.4.

Figure 5.4: Memories at the Neuron Module. In the example, the AER (search word) matches location *10* as indicated by the colored box at the CAM. The CAM then outputs the address of the matching word, which points to the value of the synapse at the RAM.



Source: The author

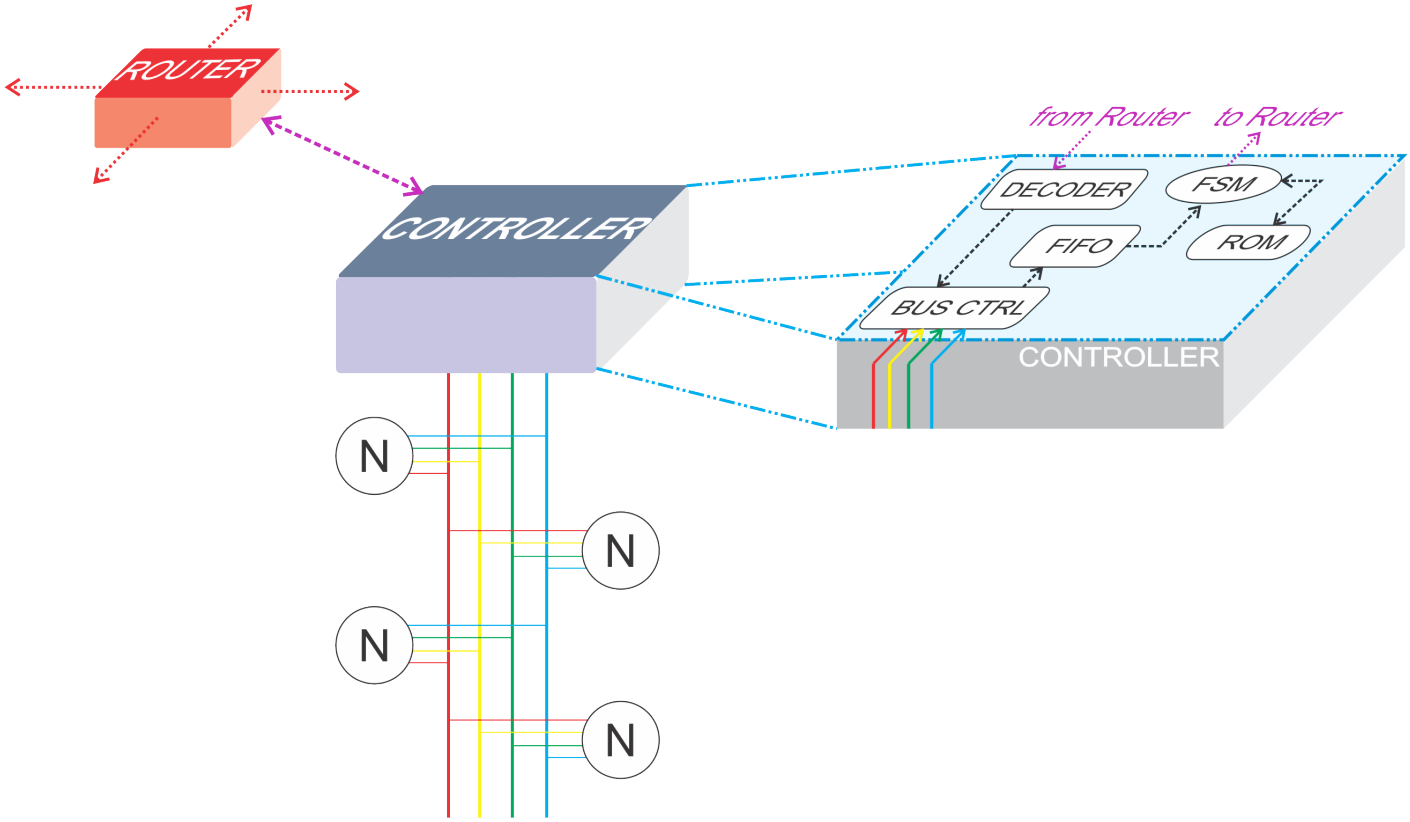
The initialization values and *.mif* files for the memories are randomly generated offline by a program in Python, following the constraints imposed by the desired network to be implemented.

The RAM output is connected to the 18-bit input signal *I* of the ISM. However, the two entities were designed to have different clock domains (with the ISM being much slower than the rest of the system). Thus in order to prevent data loss and metastability, a two-stage shift register was implemented at the crossing of domains.

5.2 The Cluster Controller

As already mentioned, the Cluster Controller is a key block within the architecture. It is composed of a ROM cell, a FIFO buffer, a Finite State Machine, a Decoder and a Bus Controller, as shown in Figure 5.5.

Figure 5.5: Controller Detail at the Cluster Level



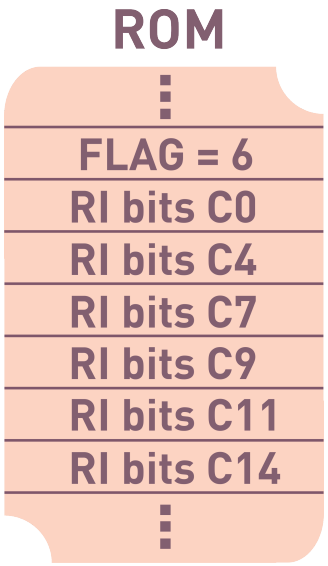
Source: The author

The Bus Controller is a necessary asset, as already mentioned, since it annuls the possibility of two neurons simultaneously sending their AE to the bus. For such, a priority encoder was implemented, and the granting signals are given in the following order: a new input from another Cluster has the highest priority, and the neurons from the same Cluster have an ascending order priority.

The FIFO is yet another module implemented in order to handle high throughput and prevent data loss. Thus, while the Cluster Controller is still in the process of sending the previous packet to the Global Router, the new AE from a neuron is stored at the FIFO and will be taken care of as soon as the FSM is ready. For 16 neurons within a single Cluster, a depth of 4 was considered enough.

The ROM, as shown in Figure 5.6, contains the following data: a FLAG for each neuron within a Cluster, containing the number of Clusters it connects with; and the RI bits, data necessary to assemble the header of the Global packet.

Figure 5.6: ROM Data for one Neuron. In the example, the Neuron connects with 6 Clusters, namely the 0, 4, 7, 9, 11 and 14. Note that no information on Neuron connections within said Clusters is necessary at the ROM.

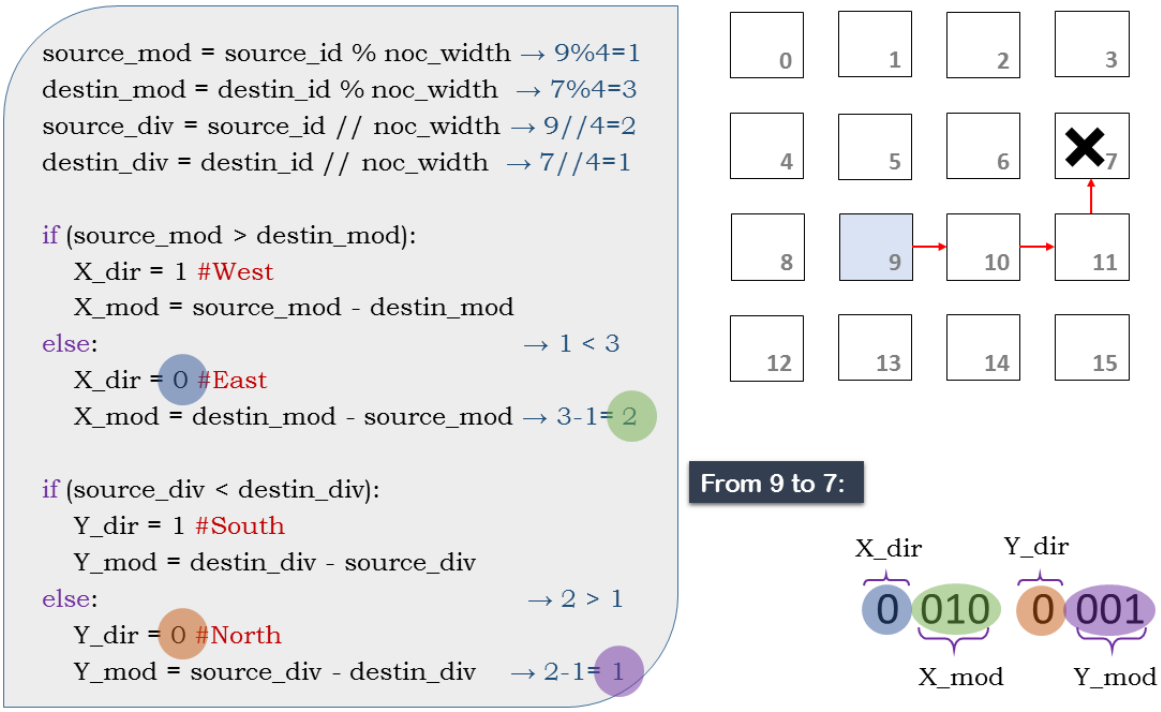


Source: The author

The aforementioned values for the ROM are generated off-line by a Python program. The program defines the connections and then assembles the RI for the Global packet according to the premises described in section 4.1.

The pseudocode for this task is shown in Figure 5.7. It also presents an example RI formulation for a given packet, to be sent from Cluster 9 to Cluster 7.

Figure 5.7: Pseudo-code for Global Packet Header assembly



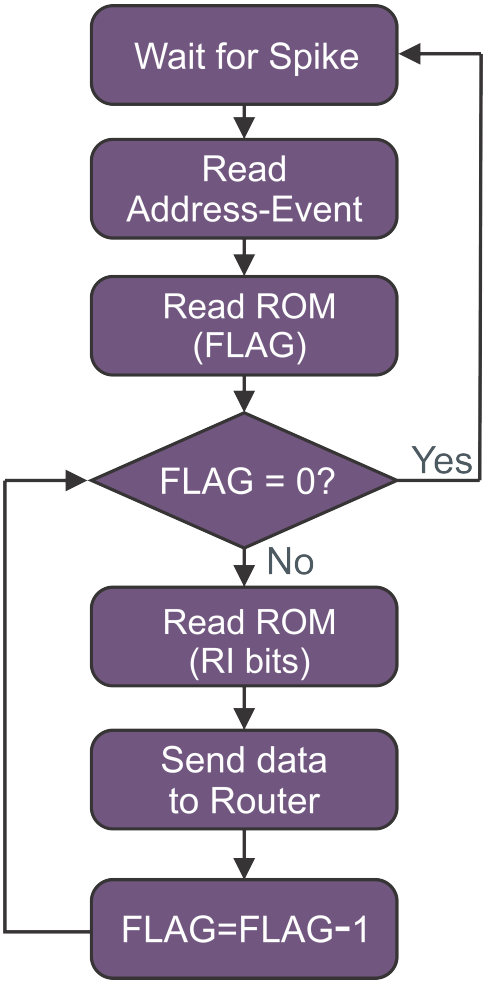
Source: The author

It is worth remembering that, as explained in Section 4.1, each Cluster Controller determines the path to be used by a packet, by including the corresponding RI in the packet header. The RI clearly expresses the pathway of each flit of the package in the following way: The most significant bit, Xdir, represents the East/West direction to be routed; the next three bits, Xmod, specify the number of links to be traversed on X (East/West) direction; the next bit, Ydir, represents the North/South direction to be routed; and finally the three least significant bits, Ymod, designate, as well, the number of links to be traversed, but now in the Y (North/South) direction.

The Finite State Machine is the module that controls the communication between the Cluster Controller and the Global Router. It assembles the Global packet and maintains the timing controls for successful transmission of each flit.

The FSM first reads the FIFO for the next Address-Event, as shown in Figure 5.8. Then, it sets a ROM address corresponding to the neuron that originated the AE, and reads its value. As previously stated, that specific first ROM address corresponds to a FLAG value, which indicates the number of Clusters that are connected to the specific neuron that spiked.

Figure 5.8: Cluster Controller FSM Logic



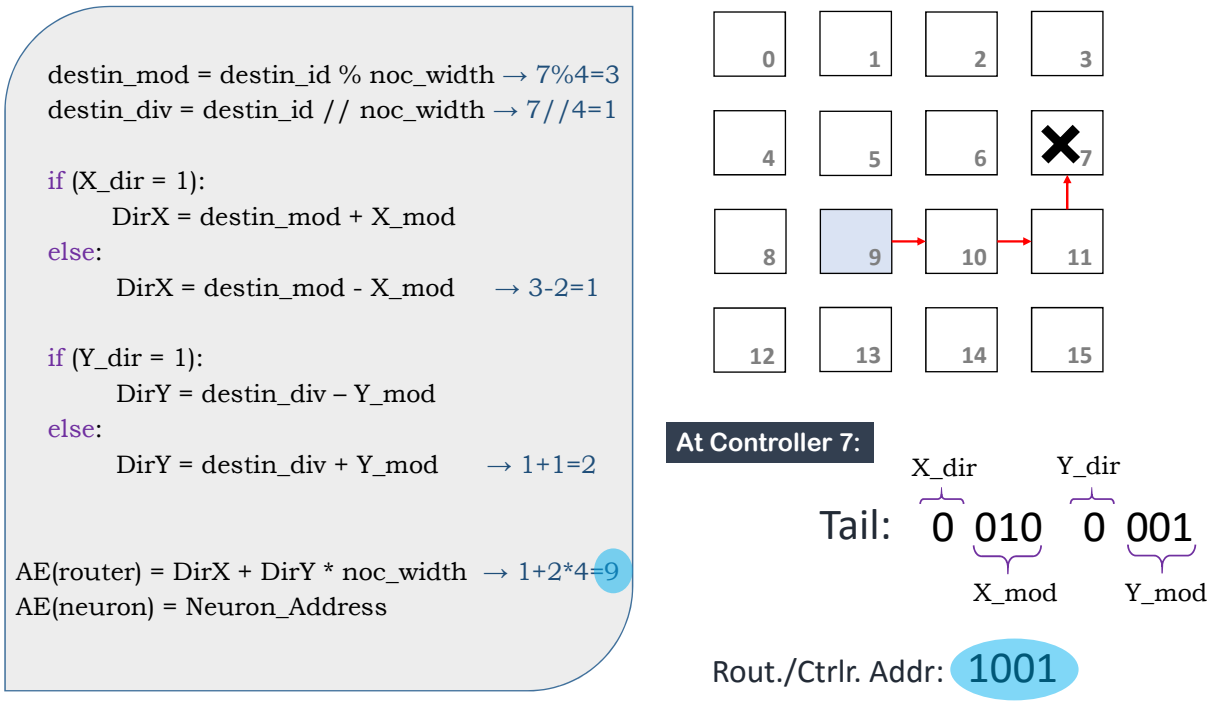
Source: The author

After reading the FLAG value at the ROM and storing it at a register, the FSM repeats the following operations, until the value at the register becomes zero: the immediately next ROM address is set, and its value is read; then, the Global packet is prepared, using the RI bits from the ROM, and sent to the Global Router; after sending the packet, the value at the register is decremented, and, if its value is still non-zero, the last operations are executed again.

The Decoder module, as the name indicates, decrypts the information received from another Cluster in order to make it readable for the neurons at a given Cluster. It takes advantage of the fact that the package Tail does not modify itself along the way (as opposed to the Header, in which the module bits are modified at every new Router in order to keep track of the path, as explained in section 4.1). Thus, when a new packet arrives from the Global Router, the Neuron Address and the Tail flits goes to the Decoder, where it recovers the original AE from the neuron that spiked.

Figure 5.9 presents the pseudo-code for this module. The Figure also shows, taking the same example as the one in Figure 5.7, the recovery of the address (Cluster 9) at the receiving side (Cluster 7).

Figure 5.9: Pseudo-code for Decoding the Global Packet. Note that the division and modulo operations use constant data, and can be inferred as constant values and not operations by the compiler.



Source: The author

6 RESULTS

Without loss of generality, a prototype 4 x 4 NoC with a varying number of neurons in each cluster has been described in mixed-language, VHDL and Verilog HDL. The validation of the proposed architecture was accomplished in two steps: functional hardware evaluation through simulation and FPGA implementation for real-time performance analysis. Moreover, a 65nm synthesis was performed for evaluation and comparison with similar state-of-art projetcs. This chapter describes the experimentation environment, the datapath traversed by an action potential after a neuron fires and the quantitative and qualitative analysis of the proposed architecture.

6.1 Testbench Setup

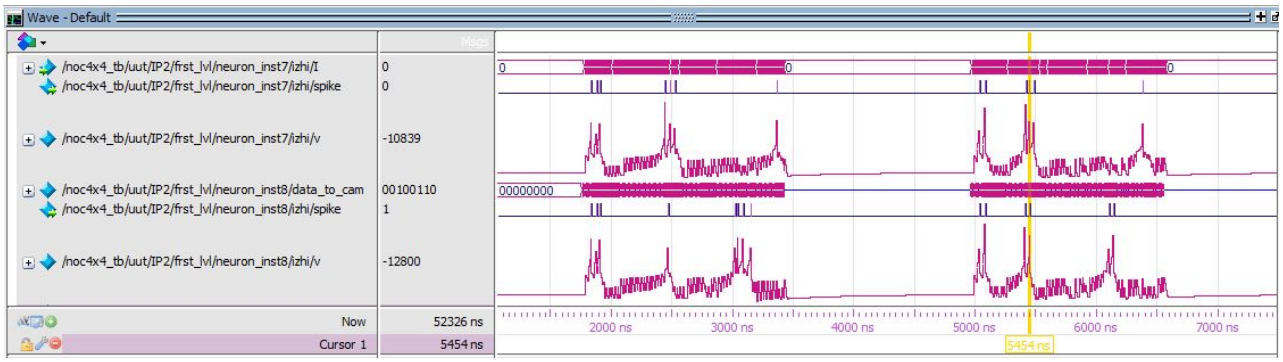
The functional hardware evaluation of DHyANA was performed using Mentor Graphics' Modelsim SE 10.1c and Altera's Quartus Prime 16.0. The validation was carried out on different stages: first the communication between neurons by direct connection was simulated; then, Clusters of different sizes were tested; afterwards the inter-level connection (Cluster Controller to Global Router) was established to verify the communication protocols; lastly, Clusters with different sizes were connected each to a Global Router and the whole architecture's functionality as verified.

This section describes the datapath that a spike takes after leaving the neuron cell until the moment it reaches other cells at different situations. Moreover, simulation images are displayed in order to better describe some of the system's functioning.

Figure 6.1 shows the voltage at two neurons after receiving input current stimulus. The peaks in voltage correspond to the action potentials that were generated at the neuron.

When such spike occurs at a neuron cell, it first asks the Bus Control permission to send data to the Bus. It then assembles the AE according to the neuron address within the system, and after the permission is set, the AE is broadcast at the Bus, which connects the Cluster Controller and all the other neuron modules.

Figure 6.1: Two Neurons Spiking After Stimulation. Note the peaks in the V signals (analog representation) and their corresponding digital signals "spike", which are the output of the neurons.



Source: The author

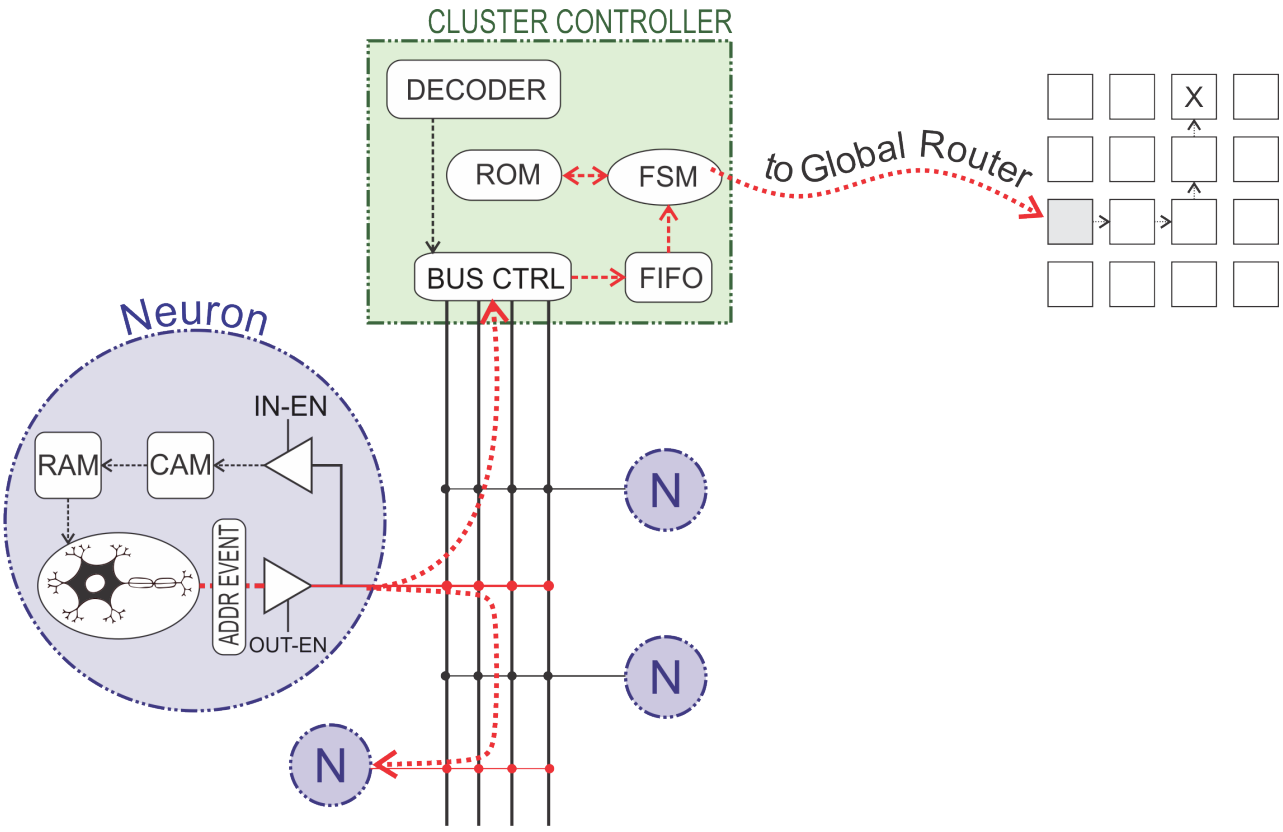
As soon as it enters the Cluster Controller, the AE is stored at the FIFO buffer. Afterward, when the FSM is ready, it takes the next AE in line and looks at its first ROM position in order to check how many connections said neuron has with other Clusters. The number of connections is determined by a FLAG signal. Then, for "FLAG times", the state machine will prepare the packet, and wait for the router confirmation to initiate the next connection. The whole FSM process was far explained in section 5.2, especially at Figure 5.8.

At Figure 6.2, the inter-Cluster pathway is indicated in red. For the situation shown at the Figure, the spike originated from a neuron (eg, n = 18) at the Cluster 8, and will be sent to one at Cluster 8 (eg, n = 38) as well as one or more neurons at the Cluster 2 (eg, n = 02 and n = 12).

Once a packet is intended to be transferred to the Global Router, a handshake between the Cluster Controller and the Global Router happens. Using the signals ACK and VAL, the packet is safely delivered to the Router (and from it to other Routers) and stored at one of the Input FIFO Buffers. The Flow Control at the Router and the FSM at the Controller coordinate such effort. Figure 6.3 shows the handshake signals as well as the packet flits being exchanged among Routers at the Global Level.

Once at the Global Router, the Header flit is analyzed for the RI bits. Then, the Routing Algorithm takes action in order to select an output channel A round-robin algorithm is then run at the Arbiter in order to select one of the requests emitted by the input channels. At the Crossbar, the signals are then transferred to the correct output according to the routing algorithms.

Figure 6.2: Afferent Datapath of the Cluster Level. For the sake of visual clarity, note that two red arrows were used to indicate the pathway from neuron 1 to the neuron 3 and to the Cluster Controller. However, this only indicates that, after the neuron 1 fired, only the neuron 3 had a CAM Hit, and the Controller will search for connections at other Clusters.



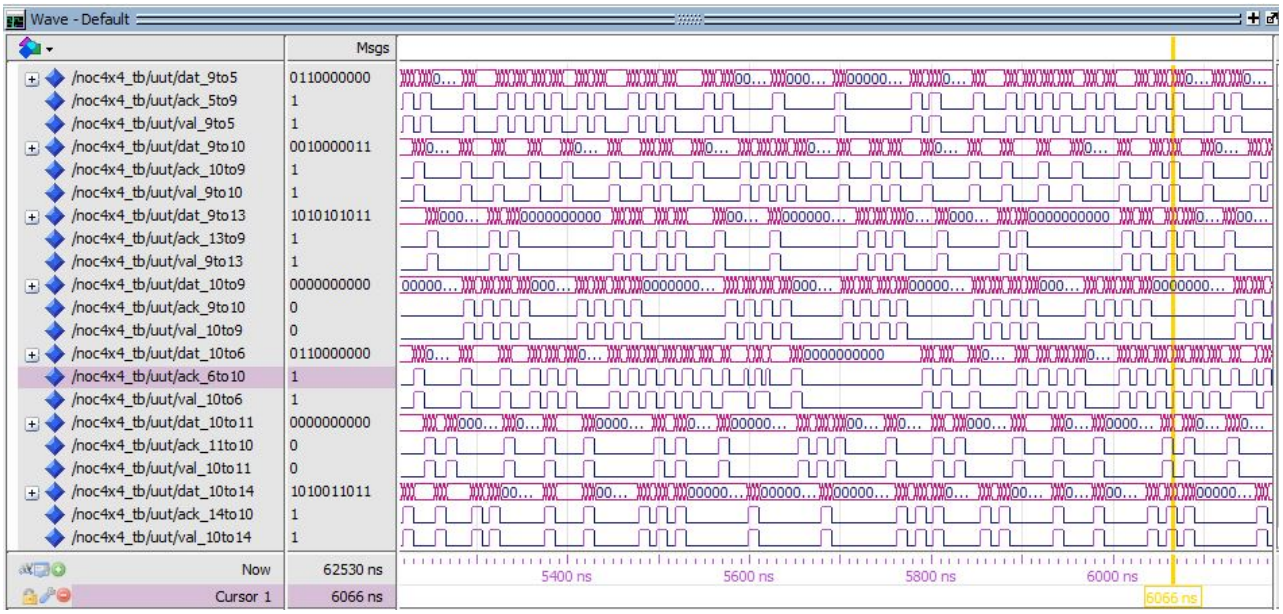
Source: The author

The mod field at the RI bits are decremented by one, related to the direction in which the packet is being routed. Such operations will happen until the two mod fields are null, meaning that the packet will finally be delivered to the Cluster. Figure 6.4 shows the pathway followed by the example previously given.

At Figure 6.4 the spike, which occurred at the Cluster 8, is delivered to one or more neurons at the Cluster 2. The image shows the update of the RI bits at each Global Router. It also shows the representation of the Router. Note that, as it represents the Router 6, the South Input and the North Output are marked red, since they are the ones used at the example.

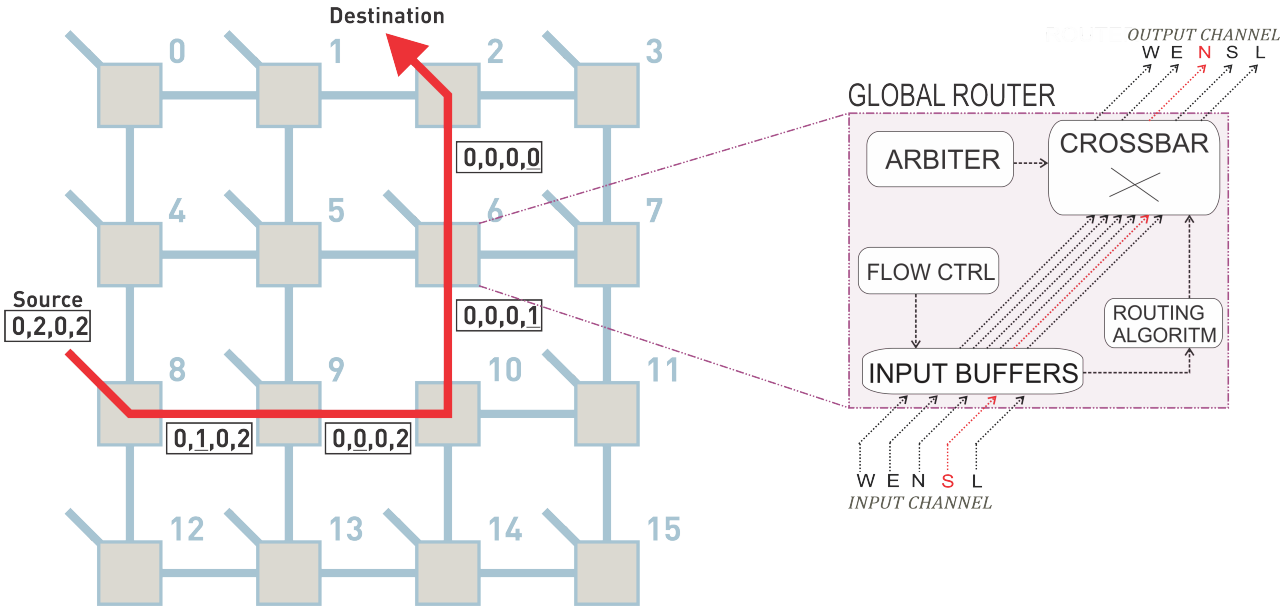
When a new data is received from the Router, it goes straight to the Decoder at the Cluster Controller. Then, after the process illustrated in Figure 5.9, the AE is ready and the Bus Control grants permission for the data to be broadcast to the Bus. Subsequently, all the neurons connected to the Bus are then able to test for connectivity, through the CAM

Figure 6.3: Routers Communication



Source: The author

Figure 6.4: Datapath of the Global Level.



Source: The author

memory.

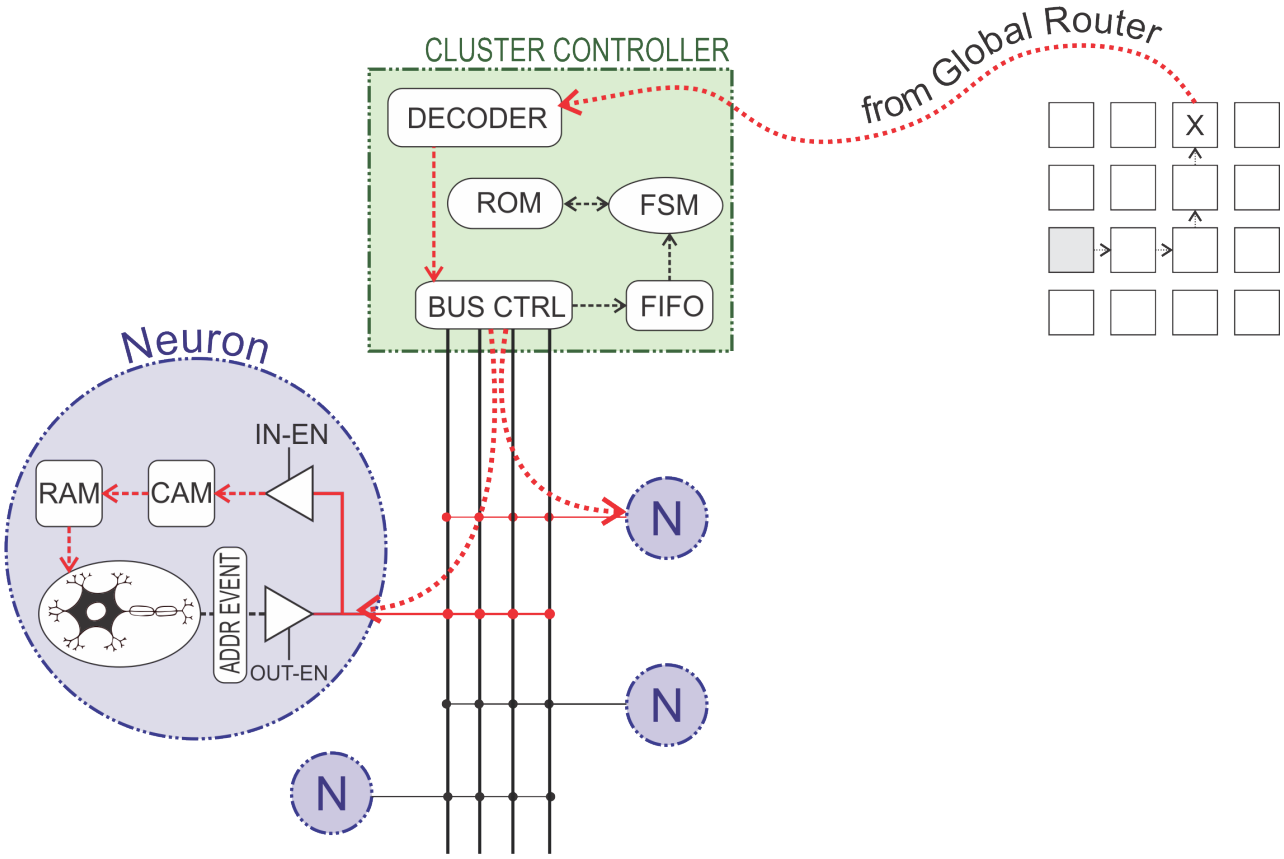
When a CAM Hit is set at any particular Neuron, it means that said neuron is connected to the Neuron who originated the spike. In that case, the address of such

connection is sent to the output of the CAM memory, which is connected as input to the RAM. At the given RAM address is stored the value of the synapse, ie, the multiplication of the current times the weight representing the connection strength.

Finally, the value of the synapse, at the output of the RAM, is connected to the input of the Izhikevich module, which will, through the equations shown in Section 4.3, compute the voltage of the neuron and maybe fire a spike and then re-initiate the cycle.

At Figure 6.5, the explained pathway is indicated in red.

Figure 6.5: Efferent Datapath of the Cluster Level. Besides the DAT signal vectors, which represent the flits, note the ACK and VAL signals, part of the handshake protocol in order to certify the connection success.



Source: The author

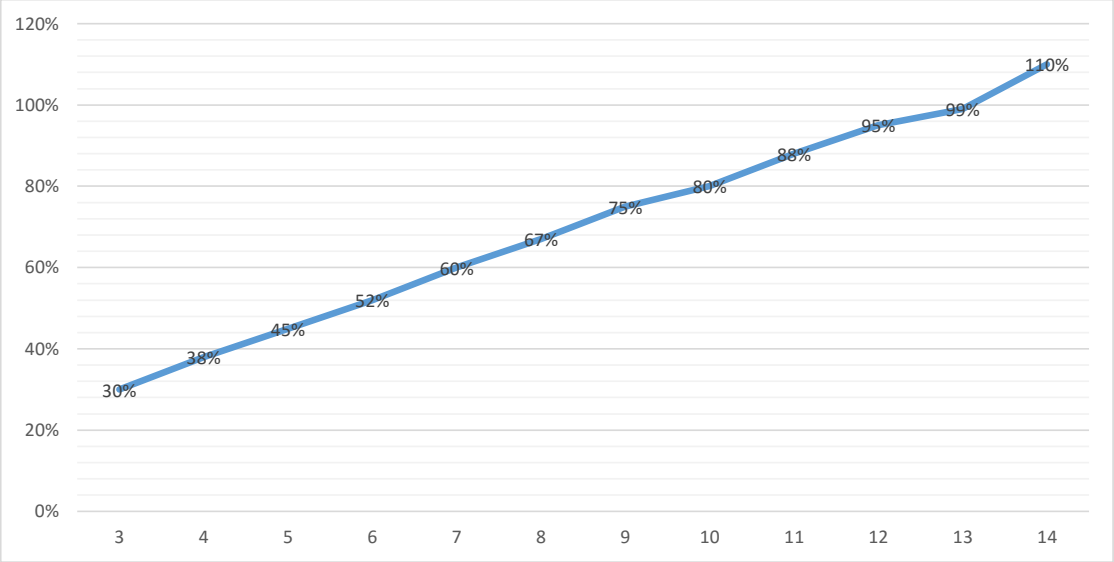
6.2 FPGA Implementation

After a careful functional examination and testing of the proposed architecture, it was implemented in an Altera Stratix IV EP4SGX230KF40C2 FPGA in order to verify the

system's real-time performance on a hardware platform.

The FPGA synthesis was realized using the Altera's Quartus Prime 16.0 software. Figure 6.6 shows the increase in synthesis time for different Cluster sizes. The graph presented, as expected, a linear curve.

Figure 6.6: Cluster Size versus FPGA Logic Utilization. The X-axis represents the quantity of neurons contained in each Cluster.



Source: The author

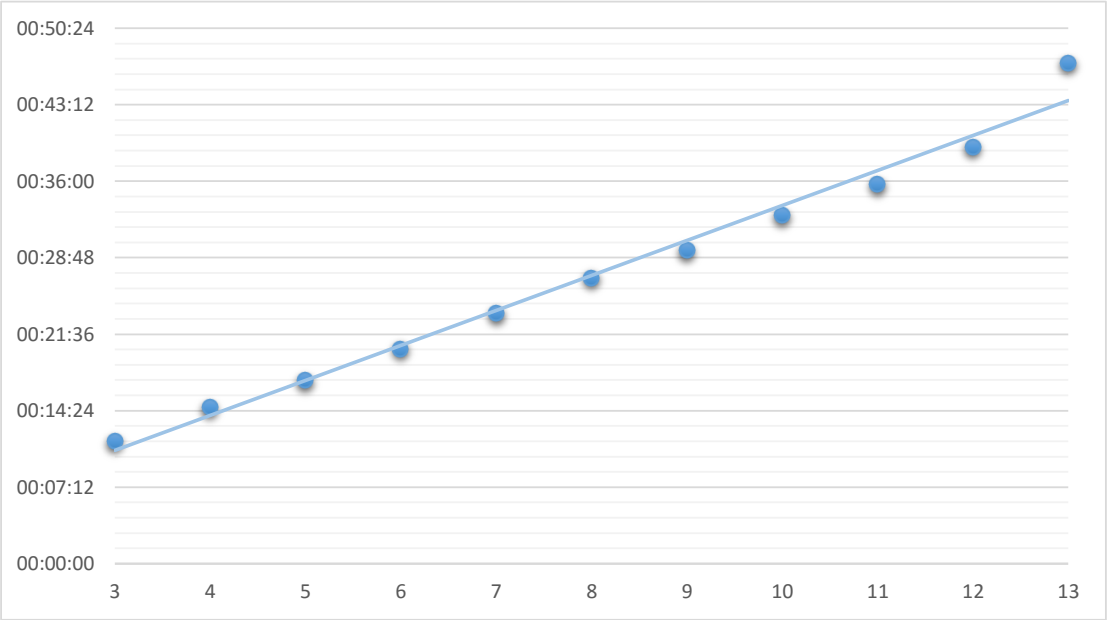
The FPGA was able to admit up to 13 neurons per Cluster (99% Logic Utilization), with a total of 208 neurons in the system. For comparison, the same FPGA chip can be filled with up to 364 ISMs (BANDEIRA et al., 2015).

Such a number is considered good enough for some classification tasks, such the spoken word recognition in (HOPFIELD; BRODY, 2001), and other benchmark Liquid State Machine implementations, as the original paper from Maass, Natschläger and Markram utilized 135 integrate-and-fire neurons for the liquid plus 51 read-out IF neurons in order to implement such applications (MAASS; NATSCHLäGER; MARKRAM, 2002).

A similarly linear growth is expected for higher numbers. However, numbers higher then 20 or 25 neurons per Cluster are expected to encounter timing issues, thus being worth considering the use of other network topologies for the lower level, such as the crossbar in (REINBRECHT, 2012), a star such as in (CARRILLO et al., 2013), a mesh as suggested in (VAINBRAND; GINOSAR, 2011) or even the creation of another level within the hierarchy.

A similar result was obtained for the increase in synthesis time, shown in Figure 6.7. The graph proves that the FPGA synthesis time is a direct consequence of the total FPGA logic utilized, which was also an expected result.

Figure 6.7: Cluster Size versus FPGA Synthesis Time Elapsed



Source: The author

For the full FPGA capacity (13 neurons per Cluster), the maximum frequency was determined to be 62.2 MHz for the communication clock. Additionally, even though the maximum frequency allowed for the neuron was 97.45 MHz, it must have a tenth of the communication frequency in order for the system to perform correctly.

6.3 Related Work Comparison and Final Remarks

A 65nm logic synthesis using Cadence Design Systems tools was performed for 256 neurons (4 x 4 mesh NoC with 16 neurons per Cluster), in order to evaluate and have some kind of comparison with similar designs. For the stated configuration, the synthesis established a power of 147mW for the system, and a chip area of 0.23mm^2.

Table 6.1 summarizes some of the DHyANA features, as well as other state-of-the-art large-scale neuromorphic platforms. It is worth mentioning, however, that although related, most measurements are not from the same benchmark nor use the same technology,

so they cannot be compared directly.

Table 6.1: DHyANA compared to Related Works

Project Name	Neuron Model	A/D	Neurons	Power	Power/ Neuron	Size mm^2	Area/ Neuron	CMOS Process	Synapses per Neuron	Topology
SpiNNaker	IF or IZHI	DS	$16x10^3$	1W	6,25E-05	102	6,38E-03	130nm	$1x10^3$	Triangular Lattice*
Neurogrid	Quad. IF	A	983,040	5W	5,08E-06	168	1,71E-04	180nm	$6x10^9$	Star
BrainScaleS	Exp. IF	A	512	1kW	1,95	430	8,40E-01	180nm	$112x10^3$	Hierarchical Buses**
EMBRACE	IF	A	400	13.16mW	3,29E-05	0.587	1,47E-03	65nm	400	Hybrid (star-mesh)
TrueNorth	-	-	$1x10^6$	65mW	5,60E-08	430	4,30E-04	28nm	256	Mesh
ROLLS	Exp. IF	A	256	4mW	1,56E-05	51.4	2,01E-01	180nm	256	Programmable Switch-matrix
DHyANA	IZHI	D	256	147mW	5,74E-04	0.23	8,98E-04	65nm	256	Hybrid (bus-mesh)

A = Analog, D = Digital, DS = Digital (Software), IF = Integrate-and-Fire, IZHI = Izhikevich. *folded into a toroid surface. **2D Torus (wafer). SpiNNaker, BrainScaleS, TrueNorth data per chip; EMBRACE data per cluster.

Source: The author

The results from Table 6.1 demonstrate, firstly, that DHyANA is comparatively similar with other works, and does not represent any significant gain in terms of area or power per neuron. This similarity in turn represents a major success of the work, taking into account that most of the related works are being developed by larger teams and/or for a larger period of time.

From a qualitative comparison with the designs reviewed in section 3.3, one can notice that although the topologies greatly differ, it is a tendency to utilize on and off-chip networks in order to not only improve communication but also to allow the necessary scalability of the systems. Moreover, some forms of intra and inter-chip (or even inter-wafer) hierarchy were found among the larger designs.

The neuron models utilized greatly vary, being the variations of the integrate-and-fire model the most utilized. This can be explained by the facility in its implementations. However, as shown in section 2.1.2, this comes at the expense of not being able to represent various types of neuron waveforms, thus restricting the biological accuracy of the whole simulation.

7 CONCLUSIONS AND FUTURE WORK

In this work, the Digital HierArchical Neuromorphic Architecture (DHyANA) was proposed. With a focus on Liquid State Machine Recurrent Neural Network implementations, the architecture utilizes hierarchical and network-on-chip approaches to improve communication and system scalability.

To the best of the author's knowledge, DHyANA is the first of its kind in that it implements a scalable general-purpose SNN with a digital hardware Izhikevich neuron model. The network implements a hybrid bus-mesh topology for communication. At the Cluster level, a shared bus connects the neurons as well as the Cluster Controller, used for controlling and for communication with the Global Level. At such level, the communication packets are composed by Address Event Representations. At the higher level, Routers communicate using XY routing, packet switching, and handshake flow control.

Different system configurations were implemented, and experimental results showed that FPGA logic utilization grew linearly with the number of neurons per Cluster. The FPGA was able to comport up to 13 neurons per Cluster, 208 neurons in total. A configuration of 256 neurons, 16 per Cluster, was also synthesized, achieving an area of $0.23mm^2$ and a power dissipation of 147mW.

Such number of neurons was considered enough for a prototype implementation, since applications from speech recognition to other benchmark configurations were realized with an even smaller number of neurons (MAASS; NATSCHLäGER; MARKRAM, 2002). Additionally, although it did not find major improvements in area and power, the design demonstrated great potential, to be explored in many different aspects in future research.

In order for the system to become more biorealistic, it is important to investigate further improvements. For instance, the implementation of even more complex neuron models, such as the Hodgkin-Huxley model, would bring further realism, although at the expense of area and power. The modeling of synaptic plasticity and other details in neuron mechanisms would also improve the system in this direction.

Additionally, some architectural changes would also bring some good advances for the system. A strategic multicasting communication, such as in (ZAMARRENO-RAMOS et al., 2013), at the Global Level would bring major improvements in timing, but could only be implemented after careful examination of the system's traffic. Also, as already mentioned, the implementation of another layer of hierarchy might be good or

even necessary for expansions of the system. Further expansions might as well greatly benefit from the implementation of a 3D mesh NoC, as in (MATOS et al., 2013; MATOS et al., 2015) and (MARCON et al., 2014).

This work has also presented a great potential for future research, and among the future directions are the development of applications in engineering and biology for the system, including, but not limited to, BMI, neuroprosthesis, large vocabulary speech recognition, self-driving cars, autonomous robot control, medical signal processing as well as experiments and data analysis in neurophysiology. Moreover, as the brain has an amazing hability to adapt to failure, future versions of DHyANA could include fault tolerance features in communication, as well as adaptability and reconfigurability proprieties. The development of CAD tools to assist the implementation of neurochips is also encouraged, as it would facilitate the design, testing and further study of the physical aspects of the system at deeper levels.

REFERENCES

AGARWAL, A.; SHANKAR, R. Survey of network on chip (noc) architectures & contributions. **Journal of Engineering, Computing and Architecture**, v. 38, p. 21–27, 2009. ISSN 1934-7197.

AKOPYAN, F. et al. Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. **Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on**, v. 34, n. 10, p. 1537–1557, Oct 2015. ISSN 0278-0070.

ALNAJJAR, F.; MURASE, K. Self-organization of spiking neural network generating autonomous behavior in a real mobile robot. In: INTERNATIONAL CONFERENCE ON COMPUTATIONAL INTELLIGENCE FOR MODELLING, CONTROL AND AUTOMATION AND INTERNATIONAL CONFERENCE ON INTELLIGENT AGENTS, WEB TECHNOLOGIES AND INTERNET COMMERCE (CIMCA-IAWTIC'06). **Proceedings...** Viena: IEEE, 2005. v. 1, p. 1134–1139.

AMBROISE, M. et al. Biorealistic spiking neural network on fpga. In: INFORMATION SCIENCES AND SYSTEMS (CISS), 2013 47TH ANNUAL CONFERENCE ON. **Proceedings...** Baltimore, MD: IEEE, 2013. p. 1–6.

ANTONELO, E. A.; SCHRAUWEN, B.; STROOBANDT, D. Event detection and localization for small mobile robots using reservoir computing. **Neural Networks**, v. 21, p. 862–871, 2008. Available from Internet: <http://reslab.elis.ugent.be/node/152>.

BANDEIRA, V. et al. Low latency fpga implementation of izhikevich-neuron model. In: INTERNATIONAL EMBEDDED SYSTEMS SYMPOSIUM (IESS 2015). **Proceedings...** Foz do Iguaçu, Brazil: [s.n.], 2015.

BARAGLIA, J.; NAGAI, Y.; ASADA, M. Action understanding using an adaptive liquid state machine based on environmental ambiguity. In: 2013 IEEE THIRD JOINT INTERNATIONAL CONFERENCE ON DEVELOPMENT AND LEARNING AND EPIGENETIC ROBOTICS (ICDL). **Proceedings...** Osaka, Japan: IEEE, 2013. p. 1–6.

BEKOLAY, T. **Learning in large-scale spiking neural networks**. Dissertation (Master) — University of Waterloo, Waterloo, Ontario, Canada, 2011. Available from Internet: <http://compneuro.uwaterloo.ca/files/publications/bekolay.2011a.pdf>.

BENINI, L.; MICHELI, G. D. Networks on chips: a new soc paradigm. **Computer**, v. 35, n. 1, p. 70–78, Jan 2002. ISSN 0018-9162.

BENJAMIN, B. et al. Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations. **Proceedings of the IEEE**, v. 102, n. 5, p. 699–716, May 2014. ISSN 0018-9219.

BJERREGAARD, T.; MAHADEVAN, S. A survey of research and practices of network-on-chip. **ACM Comput. Surv.**, ACM, New York, NY, USA, v. 38, n. 1, jun. 2006. ISSN 0360-0300. Available from Internet: <http://doi.acm.org/10.1145/1132952.1132953>.

BLUME, H. et al. Petri net based modelling of communication in systems on chip. InTech 2008, 2008.

BOHTE, S. M.; POUTRE, H. L.; KOK, J. N. Error-backpropagation in temporally encoded networks of spiking neurons. **Neurocomputing**, v. 48, p. 17–37, 2002.

BONABI, S. Y. et al. Fpga implementation of a biological neural network based on the hodgkin-huxley neuron model. **Frontiers in Neuroscience**, v. 8, n. 379, 2014. ISSN 1662-453X. Available from Internet: <http://www.frontiersin.org/neuromorphic_engineering/10.3389/fnins.2014.00379/abstract>.

BRAINSCALES. **BrainScaleS Project Website**. 2011. <http://brainscales.kip.uni-heidelberg.de/>. Accessed November, 2015.

BURGSTEINER, H. Training networks of biological realistic spiking neurons for real-time robot control. In: PROCEEDINGS OF THE 9TH INTERNATIONAL CONFERENCE ON ENGINEERING APPLICATIONS OF NEURAL NETWORKS (EAAN2005). **Proceedings...** Lille, France: [s.n.], 2005. p. 129–136.

CAMBRIDGE DICTIONARY. **Cluster.** [S.l.]: Cambridge University Press, 2016.

CAO, R.; PIPA, G. **Extended Liquid Computing in Networks of Spiking Neurons**. Deutschordenstraße 46 D-60528 Frankfurt/Main, 2010.

CARRILLO, S. et al. Hierarchical network-on-chip and traffic compression for spiking neural network implementations. In: NETWORKS ON CHIP (NOCS), 2012 SIXTH IEEE/ACM INTERNATIONAL SYMPOSIUM ON. **Proceedings...** Copenhagen: IEEE, 2012. p. 83–90.

CARRILLO, S. et al. Scalable hierarchical network-on-chip architecture for spiking neural network hardware implementations. **Parallel and Distributed Systems, IEEE Transactions on**, v. 24, n. 12, p. 2451–2461, Dec 2013. ISSN 1045-9219.

CASSIDY, A.; ANDREOU, A. G. Dynamical digital silicon neurons. In: 2008 IEEE BIOMEDICAL CIRCUITS AND SYSTEMS CONFERENCE. **Proceedings...** Baltimore, MD: IEEE, 2008. p. 289–292. ISSN 2163-4025.

CASSIDY, A. S.; GEORGIOU, J.; ANDREOU, A. G. Design of silicon brains in the nano-cmos era: Spiking neurons, learning synapses and neural architecture optimization. **Neural Networks**, v. 45, p. 4 – 26, 2013. ISSN 0893-6080. Neuromorphic Engineering: From Neural Systems to Brain-Like Engineered Systems. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0893608013001597>.

CHEN, G. et al. 16.1 a 340mv-to-0.9v 20.2tb/s source-synchronous hybrid packet/circuit-switched 16x16 network-on-chip in 22nm tri-gate cmos. In: 2014 IEEE INTERNATIONAL SOLID-STATE CIRCUITS CONFERENCE DIGEST OF TECHNICAL PAPERS (ISSCC). **Proceedings...** San Francisco, CA: IEEE, 2014. p. 276–277. ISSN 0193-6530.

CHOUDHARY, S. et al. Silicon neurons that compute. In: VILLA, A. E. P. et al. (Ed.). [S.l.]: Springer, 2012. (Lecture Notes in Computer Science, v. 7552), p. 121–128. ISBN 978-3-642-33268-5.

DALLY, W. J.; SEITZ, C. L. The torus routing chip. **Distributed Computing**, v. 1, n. 4, p. 187–196, 1986. ISSN 1432-0452. Available from Internet: <http://dx.doi.org/10.1007/BF01660031>.

DALLY, W. J.; TOWLES, B. Route packets, not wires: On-chip inteconnection networks. In: 38TH ANNUAL DESIGN AUTOMATION CONFERENCE. **Proceedings...** New York, NY, USA: ACM, 2001. (DAC '01), p. 684–689. ISBN 1-58113-297-2. Available from Internet: <http://doi.acm.org/10.1145/378239.379048>.

DAS, R. et al. Design and evaluation of a hierarchical on-chip interconnect for next-generation cmps. In: 2009 IEEE 15TH INTERNATIONAL SYMPOSIUM ON HIGH PERFORMANCE COMPUTER ARCHITECTURE. **Proceedings...** Raleigh, NC: IEEE, 2009. p. 175–186. ISSN 1530-0897.

DAYAN, P.; ABBOTT, L. F. **Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems**. [S.l.]: The MIT Press, 2005. ISBN 0262541858.

DECHARMS, R. C.; ZADOR, A. Neural representation and the cortical code. **Annual Review of Neuroscience**, v. 23, n. 1, p. 613–647, 2000. PMID: 10845077. Available from Internet: <http://dx.doi.org/10.1146/annurev.neuro.23.1.613>.

DEISS, S. R. et al. **A Pulse-Coded Communications Infrastructure for Neuromorphic Systems**. 1998.

FACETS. **FACETS Project Website**. 2005. <http://facets.kip.uni-heidelberg.de/>. Accessed November, 2015.

FERNANDO, C.; SOJAKKA, S. Pattern recognition in a bucket. In: Advances in Artificial Life: 7th European Conference, ECAL 2003. **Proceedings...** Berlin, Heidelberg: Springer 2003. p. 588–597. ISBN 978-3-540-39432-7. Available from Internet: <http://dx.doi.org/10.1007/978-3-540-39432-7_63>.

FERREIRA, A. A. et al. Investigating the use of reservoir computing for forecasting the hourly wind speed in short -term. In: PROCEEDINGS OF THE INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS, IJCNN 2008, PART OF THE IEEE WORLD CONGRESS ON COMPUTATIONAL INTELLIGENCE, WCCI 2008. **Proceedings...** Hong Kong, China: IEEE, 2008. p. 1649–1656.

FLORIAN, R. V. **Supervised Learning in Spiking Neural Networks**. Boston, MA: Springer US, 2012. 3245–3247 p. ISBN 978-1-4419-1428-6. Available from Internet: <http://dx.doi.org/10.1007/978-1-4419-1428-6_1714>.

FOX, P. J. **Massively parallel neural computation**. [S.l.], 2013. Available from Internet: <http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-830.pdf>.

FURBER, S. et al. Overview of the spinnaker system architecture. **Computers, IEEE Transactions on**, v. 62, n. 12, p. 2454–2467, Dec 2013. ISSN 0018-9340.

GHANI, A. et al. Neuro-inspired speech recognition with recurrent spiking neurons. In: ARTIFICIAL NEURAL NETWORKS - ICANN 2008: 18TH INTERNATIONAL CONFERENCE, Prague, Czech Republic, September 3-6, 2008. **Proceedings...** Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. p. 513–522. ISBN 978-3-540-87536-9. Available from Internet: <http://dx.doi.org/10.1007/978-3-540-87536-9_53>.

GRUNING, A.; BOHTE, S. M. Spiking neural networks: Principles and challenges. In: ESANN 2014 PROCEEDINGS, EUROPEAN SYMPOSIUM ON ARTIFICIAL NEURAL NETWORKS. **Proceedings...** Bruges, Belgium: [s.n.], 2014. p. 23–25. ISBN 978-287419095-7.

GRZYB, B. J. et al. Facial expression recognition based on liquid state machines built of alternative neuron models. In: 2009 INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS. **Proceedings...** Atlanta, GA: IEEE, 2009. p. 1011–1017. ISSN 2161-4393.

HARKIN, J. et al. A reconfigurable and biologically inspired paradigm for computation using network-on-chip and spiking neural networks. **Int. J. Reconfig. Comput.**, Hindawi Publishing Corp., New York, NY, United States, v. 2009, p. 2:1–2:13, jan. 2009. ISSN 1687-7195. Available from Internet: <http://dx.doi.org/10.1155/2009/908740>.

HEMANI, A. et al. Network on chip: An architecture for billion transistor era. In: PROCEEDING OF THE IEEE NORCHIP CONFERENCE. **Proceedings...** IEEE, 2000. Available from Internet: <http://www.imit.kth.se/~axel/papers/2000/norchip-noc.pdf>.

HERTZBERG, J.; JAEGER, H.; SCHöNHERR, F. **Learning to Ground Fact Symbols in Behavior-Based Robots**. 2002.

HODGKIN, A.; HUXLEY, A. Propagation of electrical signals along giant nerve fibres. **Proceedings of the Royal Society of London. Series B, Biological Sciences**, v. 140, n. 899, p. 177–183, Oct 1952.

HOPFIELD, J. J.; BRODY, C. D. What is a moment? Transient synchrony as a collective mechanism for spatiotemporal integration. **Proceedings of the National Academy of Sciences**, National Academy of Sciences, v. 98, n. 3, p. 1282–1287, jan. 2001. ISSN 1091-6490. Available from Internet: <http://dx.doi.org/10.1073/pnas.031567098>.

HOWARD, J. et al. A 48-core ia-32 message-passing processor with dvfs in 45nm cmos. In: 2010 IEEE INTERNATIONAL SOLID-STATE CIRCUITS CONFERENCE - (ISSCC). **Proceedings...** Beijing: IEEE, 2010. p. 108–109. ISSN 0193-6530.

HULEA, M.; CARUNTU, C. Spiking neural network for controlling the artificial muscles of a humanoid robotic arm. In: SYSTEM THEORY, CONTROL AND COMPUTING (ICSTCC), 2014 18TH INTERNATIONAL CONFERENCE. **Proceedings...** Sinaia: IEEE, 2014. p. 163–168.

IZHIKEVICH, E. M. Simple model of spiking neurons. **IEEE Trans. Neural Networks**, p. 1569–1572, 2003.

IZHIKEVICH, E. M. Which model to use for cortical spiking neurons? **Neural Networks, IEEE Transactions on**, v. 15, n. 5, p. 1063–1070, Sept 2004. ISSN 1045-9227.

IZHIKEVICH, E. M. **Dynamical systems in neuroscience : the geometry of excitability and bursting**. Cambridge, Mass., London: MIT Press, 2007. (Computational Neuroscience). ISBN 0-262-09043-0. Available from Internet: <http://opac.inria.fr/record=b1125242>.

IZHIKEVICH, E. M.; EDELMAN, G. M. Large-scale model of mammalian thalamocortical systems. **Proceedings of the National Academy of Sciences**, v. 105, n. 9, p. 3593–3598, 2008. Available from Internet: <http://www.pnas.org/content/105/9/3593.abstract>.

JAEGER, H. **The ''echo state'' approach to analysing and training recurrent neural networks.** 2001. There is an Erratum Note for this techreport. Available from Internet: <http://www.faculty.jacobs-university.de/hjaeger/pubs/EchoStatesTechRep.pdf>.

JAEGER, H. **Short term memory in echo state networks.** 2001. Available from Internet: <http://minds.jacobs-university.de/sites/default/files/uploads/papers/STMEchoStatesTechRep.pdf>.

JAEGER, H.; HAAS, H. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. **Science**, American Association for the Advancement of Science, v. 304, n. 5667, p. 78–80, 2004. ISSN 0036-8075. Available from Internet: <http://science.sciencemag.org/content/304/5667/78>.

JANTSCH, A.; TENHUNEN, H. (Ed.). **Networks on Chip**. Hingham, MA, USA: Kluwer Academic Publishers, 2003. ISBN 1-4020-7392-5.

JOHANSSON, C.; LANSNER, A. Towards cortex sized artificial neural systems. **Neural Netw.**, Elsevier Science Ltd., Oxford, UK, UK, v. 20, n. 1, p. 48–61, jan. 2007. ISSN 0893-6080. Available from Internet: <http://dx.doi.org/10.1016/j.neunet.2006.05.029>.

JOSHI, P.; MAASS, W. Movement generation and control with generic neural microcircuits. In: BIOLOGICALLY INSPIRED APPROACHES TO ADVANCED INFORMATION TECHNOLOGY: FIRST INTERNATIONAL WORKSHOP, BioADIT 2004, Lausanne, Switzerland, January 29-30, 2004, Revised Selected Papers. **Proceedings...** Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. p. 258–273. ISBN 978-3-540-27835-1. Available from Internet: <http://dx.doi.org/10.1007/978-3-540-27835-1_20>.

JU, H. et al. Spatiotemporal memory is an intrinsic property of networks of dissociated cortical neurons. **The Journal of Neuroscience**, v. 35, n. 9, p. 4040–4051, March 2015.

JU, H.; XU, J. X.; VANDONGEN, A. M. J. Classification of musical styles using liquid state machines. In: THE 2010 INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS (IJCNN). **Proceedings...** Barcelona: IEEE, 2010. p. 1–7. ISSN 2161-4393.

KANDEL, E. R. et al. (Ed.). **Principles of neural science**. New York, Chicago, San Francisco: McGraw-Hill Medical, 2013. ISBN 978-0-07-139011-8. Available from Internet: <http://opac.inria.fr/record=b1135227>.

KERR, D. et al. A novel approach to robot vision using a hexagonal grid and spiking neural networks. In: NEURAL NETWORKS (IJCNN), THE 2012 INTERNATIONAL JOINT CONFERENCE ON. **Proceedings...** Brisbane, QLD: IEEE, 2012. p. 1–7. ISSN 2161-4393.

KULKARNI, S.; BAGHINI, M. Spiking neural network based asic for character recognition. In: NATURAL COMPUTATION (ICNC), 2013 NINTH INTERNATIONAL CONFERENCE ON. **Proceedings...** Shenyang: IEEE, 2013. p. 194–199.

KUMAR, A.; ROTTER, S.; AERTSEN, A. Spiking activity propagation in neuronal networks: reconciling different perspectives on neural coding. **Nat Rev Neurosci**, v. 11, n. 9, p. 615 – 627, 2010.

LEE, H. G. et al. On-chip communication architecture exploration: A quantitative evaluation of point-to-point, bus, and network-on-chip approaches. **ACM Trans. Des. Autom. Electron. Syst.**, ACM, New York, NY, USA, v. 12, n. 3, p. 23:1–23:20, may 2008. ISSN 1084-4309. Available from Internet: <http://doi.acm.org/10.1145/1255456.1255460>.

LENG, X. et al. Implementation and simulation of a cluster-based hierarchical noc architecture for multi-processor soc. In: IEEE INTERNATIONAL SYMPOSIUM ON COMMUNICATIONS AND INFORMATION TECHNOLOGY, 2005. ISCIT 2005. **Proceedings...** Beijing: IEEE, 2005. v. 2, p. 1203–1206.

LIU, D.; YUE, S. Spiking neural network for visual pattern recognition. In: MULTISENSOR FUSION AND INFORMATION INTEGRATION FOR INTELLIGENT SYSTEMS (MFI), 2014 INTERNATIONAL CONFERENCE ON. **Proceedings...** Beijing: IEEE, 2014. p. 1–5.

LUKOSEVICIUS, M.; JAEGER., H. **Overview of Reservoir Recipes**. Dissertation (Technical Report) — Jacobs University, Bremen, 2007. Available from Internet: <http://www.jacobs-university.de/imperia/md/content/groups/research/techreports/reservoiroverview_techreport11.pdf>.

LUKOSEVICIUS, M.; JAEGER, H. Survey: Reservoir computing approaches to recurrent neural network training. **Comput. Sci. Rev.**, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 3, n. 3, p. 127–149, aug. 2009. ISSN 1574-0137. Available from Internet: <http://dx.doi.org/10.1016/j.cosrev.2009.03.005>.

MAASS, W. Lower bounds for the computational power of networks of spiking neurons. **Neural Computation**, v. 8, n. 1, p. 1–40, Jan 1996. ISSN 0899-7667.

MAASS, W. Networks of spiking neurons: The third generation of neural network models. **Trans. Soc. Comput. Simul. Int.**, Society for Computer Simulation International, San Diego, CA, USA, v. 14, n. 4, p. 1659–1671, dec. 1997. ISSN 0740-6797. Available from Internet: <http://dl.acm.org/citation.cfm?id=281543.281637>.

MAASS, W. Computing with spikes. **Special Issue on Foundations of Information Processing of Telematik**, v. 8, n. 1, p. 32–36, 2002.

MAASS, W. Motivation, theory, and applications of liquid state machines. In: **Computability in Context: Computation and Logic in the Real World**. [S.l.]: Imperial College Press, 2011.

MAASS, W.; NATSCHLäGER, T.; MARKRAM, H. Real-time computing without stable states: A new framework for neural computation based on perturbations. **Neural Comput.**, MIT Press, Cambridge, MA, USA, v. 14, n. 11, p. 2531–2560, nov. 2002. ISSN 0899-7667. Available from Internet: <http://dx.doi.org/10.1162/089976602760407955>.

MAGUIRE, L. P. et al. Challenges for large-scale implementations of spiking neural networks on fpgas. **Neurocomputing**, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 71, n. 1-3, p. 13–29, dec. 2007. ISSN 0925-2312. Available from Internet: <http://dx.doi.org/10.1016/j.neucom.2006.11.029>.

MAHOWALD, M. F. **VLSI analogs of neuronal visual processing: a synthesis of form and function**. Thesis (PhD) — Calif. Univ. Pasadena, Pasadena, CA, 1992. Presented on 12 May 1992.

MARCON, C. et al. Tiny - optimised 3d mesh noc for area and latency minimisation. **Electronics Letters**, v. 50, n. 3, p. 165–166, January 2014. ISSN 0013-5194.

MARKRAM, H. The blue brain project. **Nature Rev. Neuroscience**, v. 7, n. 2, p. 153–159, 2006. Available from Internet: <http://dx.doi.org/10.1038/nrn1848>.

MARKRAM, H. et al. Regulation of synaptic efficacy by coincidence of postsynaptic aps and epsps. **Science (New York)**, v. 275, n. 5, p. 213–5, 1997.

MATOS, D. et al. Performance evaluation of hierarchical noc topologies for stacked 3d ics. In: 2015 IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS (ISCAS). **Proceedings...** Lisbon: IEEE, 2015. p. 1961–1964. ISSN 0271-4302.

MATOS, D. et al. A power-efficient hierarchical network-on-chip topology for stacked 3d ics. In: 2013 IFIP/IEEE 21ST INTERNATIONAL CONFERENCE ON VERY LARGE SCALE INTEGRATION (VLSI-SoC). **Proceedings...** Istanbul: IEEE, 2013. p. 308–313. ISSN 2324-8432.

MCKENNOCH, S.; LIU, D.; BUSHNELL, L. G. Fast modifications of the spikeprop algorithm. In: INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS. **Proceedings...** Vancouver, BC: IEEE, 2006. p. 3970–3977.

MEHRING, C. et al. Activity dynamics and propagation of synchronous spiking in locally connected random networks. **Biological cybernetics**, v. 88, n. 5, p. 395–408, 2003.

MELLO, A. V. de et al. **Evaluation of Routing Algorithms on Mesh Based NoCs**. [S.l.], 2004.

MEROLLA, P. et al. A multicast tree router for multichip neuromorphic systems. **Circuits and Systems I: Regular Papers, IEEE Transactions on**, v. 61, n. 3, p. 820–833, March 2014. ISSN 1549-8328.

MISRA, J.; SAHA, I. Artificial neural networks in hardware: A survey of two decades of progress. **Neurocomputing**, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 74, n. 1-3, p. 239–255, dec. 2010. ISSN 0925-2312. Available from Internet: <http://dx.doi.org/10.1016/j.neucom.2010.03.021>.

MORAES, F. et al. Hermes: an infrastructure for low area overhead packet-switching networks on chip. **Integration, the {VLSI} Journal**, v. 38, n. 1, p. 69 – 93, 2004. ISSN 0167-9260. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0167926004000185>.

NEDJAH, N.; MOURELLE, L. de M. Routing in network-on-chips using ant colony optimization. In: **Hardware for Soft Computing and Soft Computing for Hardware**. Springer International Publishing, 2014, (Studies in Computational Intelligence, v. 529). p. 173–198. ISBN 978-3-319-03109-5. Available from Internet: <http://dx.doi.org/10.1007/978-3-319-03110-1_11>.

NEUGRID. **Brains in Silicon Website**. 2006. <https://web.stanford.edu/group/brainsinsilicon>. Accessed November, 2015.

OGRAS, U. Y.; MARCULESCU, R. **Modeling, Analysis and Optimization of Network-on-Chip Communication Architectures**. [S.l.]: Springer Publishing Company, Incorporated, 2013. ISBN 9400739575, 9789400739574.

PAGIAMTZIS, K.; SHEIKHOLESLAMI, A. Content-addressable memory (cam) circuits and architectures: a tutorial and survey. **IEEE Journal of Solid-State Circuits**, v. 41, n. 3, p. 712–727, March 2006. ISSN 0018-9200.

PAUGAM-MOISY, H.; BOHTE, E. Computing with spiking neuron networks. In: **Handbook of Natural Computing**. [S.l.]: Springer Verlag, 2012. ISBN 978-3-540-92909-3.

POTJANS, W.; MORRISON, A.; DIESMANN, M. A spiking neural network model of an actor-critic learning agent. **Neural Comput.**, MIT Press, Cambridge, MA, USA, v. 21, n. 2, p. 301–339, feb. 2009. ISSN 0899-7667. Available from Internet: <http://dx.doi.org/10.1162/neco.2008.08-07-593>.

POUGET, A.; DAYAN, P.; ZEMEL, R. Information processing with population codes. **Nature reviews. Neuroscience**, Department of Brain and Cognitive Sciences, Meliora Hall, University of Rochester, Rochester, New York 14627, USA. alex@bcs.rochester.edu, v. 1, n. 2, p. 125–132, nov. 2000. ISSN 1471-003X. Available from Internet: <http://dx.doi.org/10.1038/35039062>.

QIAO, N. et al. A re-configurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses. **Frontiers in Neuroscience**, v. 9, n. 141, 2015. ISSN 1662-453X.

REID, D.; HUSSAIN, A.; TAWFIK, H. Spiking neural networks for financial data prediction. In: NEURAL NETWORKS (IJCNN), THE 2013 INTERNATIONAL JOINT CONFERENCE ON. **Proceedings...** Dallas, TX: IEEE, 2013. p. 1–10. ISSN 2161-4393.

REINBRECHT, C. R. W. **Desenvolvimento e avaliação de redes-em-chip hierárquicas e reconfiguráveis para MPSoCs**. Dissertation (Master) — Universidade Federal do Rio Grande do Sul, 2012.

ROY, A.; SCHAFFER, J. D.; LARAMEE, C. B. Evolving spike neural network sensors to characterize the alcoholic brain using visually evoked response potential. **Procedia Computer Science**, v. 20, p. 27 – 32, 2013. ISSN 1877-0509. Complex Adaptive Systems. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S187705091301034X>.

SALMINEN, E.; KULMALA, A.; HAMALAINEN, T. D. On network-on-chip comparison. In: DIGITAL SYSTEM DESIGN ARCHITECTURES, METHODS AND TOOLS, 2007. DSD 2007. 10TH EUROMICRO CONFERENCE ON. Lubeck: IEEE, 2007. p. 503–510.

SCHÄFER, M. et al. Simulation of spiking neural networks - architectures and implementations. **Neurocomputing**, v. 48, n. 1-4, p. 647–679, 2002.

SCHEMMEL, J. et al. A wafer-scale neuromorphic hardware system for large-scale neural modeling. In: CIRCUITS AND SYSTEMS (ISCAS), PROCEEDINGS OF 2010 IEEE INTERNATIONAL SYMPOSIUM ON. **Proceedings...** Paris: IEEE, 2010. p. 1947–1950.

SCHMIDT, D. **Automated Characterization of a Wafer-Scale Neuromorphic Hardware System**. Dissertation (Master) — University of Heidelberg, Germany, 2014.

SCHRAUWEN, B. et al. Compact hardware liquid state machines on {FPGA} for real-time speech recognition. **Neural Networks**, v. 21, n. 2–3, p. 511 – 523, 2008. ISSN 0893-6080. Advances in Neural Networks Research: {IJCNN} '072007 International Joint Conference on Neural Networks {IJCNN} '07. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0893608007002353>.

SCHRAUWEN, B.; VERSTRAETEN, D.; CAMPENHOUT, J. V. An overview of reservoir computing: theory, applications and implementations. In: PROCEEDINGS OF THE 15TH EUROPEAN SYMPOSIUM ON ARTIFICIAL NEURAL NETWORKS. **Proceedings...** Bruges, Belgium: [s.n.], 2007. p. 471–482.

SEGEV, I. Lecture notes: Brain excitements for the 21st century. **Synapses, Neurons and Brains**, Hebrew University of Jerusalem / Coursera, 2016. Available from Internet: <https://d396qusza40orc.cloudfront.net/bluebrain/Lesson%201/Coursera_lecture%231.pdf>.

SHADLEN, M. N.; NEWSOME, W. T. Noise, neural codes and cortical organization. **Current Opinion in Neurobiology**, v. 4, n. 4, p. 569 – 579, 1994. ISSN 0959-4388. Available from Internet: <http://www.sciencedirect.com/science/article/pii/0959438894900590>.

SINGER, W. Putative Functions of Temporal Correlations in Neocortical Processing. In: **Large-Scale Neuronal Theories of the Brain**. 2008. chp. 10, p. 201–237.

SIVILOTTI, M. A. **Wiring Considerations in Analog VLSI Systems, with Application to Field-programmable Networks**. Thesis (PhD), Pasadena, CA, USA, 1991. UMI Order No. GAX91-37292.

SMARAGDOS, G. et al. Fpga-based biophysically-meaningful modeling of olivocerebellar neurons. In: 2014 ACM/SIGDA INTERNATIONAL SYMPOSIUM ON FIELD-PROGRAMMABLE GATE ARRAYS. **Proceedings...** New York, NY, USA: ACM, 2014. (FPGA '14), p. 89–98. ISBN 978-1-4503-2671-1. Available from Internet: <http://doi.acm.org/10.1145/2554688.2554790>.

SPINNAKER. **SpiNNaker Project Website**. 2005. <http://apt.cs.manchester.ac.uk/projects/SpiNNaker/>. Accessed November, 2015.

STROGATZ, S. H. Exploring complex networks. **Nature**, v. 410, n. 6825, p. 268 – 276, 2001.

SUN, Q. et al. Implementation study of an analog spiking neural network for assisting cardiac delay prediction in a cardiac resynchronization therapy device. **Neural Networks, IEEE Transactions on**, v. 22, n. 6, p. 858–869, June 2011. ISSN 1045-9227.

THOMAS, D.; LUK, W. Fpga accelerated simulation of biologically plausible spiking neural networks. In: FIELD PROGRAMMABLE CUSTOM COMPUTING MACHINES, 2009. FCCM '09. 17TH IEEE SYMPOSIUM ON. **Proceedings...** Napa, CA: IEEE, 2009. p. 45–52.

VAINBRAND, D.; GINOSAR, R. Scalable network-on-chip architecture for configurable neural networks. **Microprocessors and Microsystems**, v. 35, n. 2, p. 152 – 166, 2011. ISSN 0141-9331. Special issue on Network-on-Chip Architectures and Design Methodologies. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0141933110000517>.

VANGAL, S. et al. An 80-tile 1.28tflops network-on-chip in 65nm cmos. In: 2007 IEEE INTERNATIONAL SOLID-STATE CIRCUITS CONFERENCE. **Proceedings...**. San Francisco, CA: IEEE, 2007. p. 98–589. ISSN 0193-6530.

VERSTRAETEN, D. **Reservoir Computing: computation with dynamical systems**. 190 p. Dissertation (PhD thesis) — Ghent University, Gent, 2009.

VERSTRAETEN, D. et al. An experimental unification of reservoir computing methods. **Neural Networks**, v. 20, n. 3, p. 391 – 403, 2007. ISSN 0893-6080. Echo State Networks and Liquid State Machines. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S089360800700038X>.

VERSTRAETEN, D. et al. Isolated word recognition with the liquid state machine: a case study. **Information Processing Letters**, v. 95, n. 6, p. 521 – 528, 2005. ISSN 0020-0190. Applications of Spiking Neural Networks. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0020019005001523>.

WANG, Q.; JIN, Y.; LI, P. General-purpose lsm learning processor architecture and theoretically guided design space exploration. In: BIOMEDICAL CIRCUITS AND SYSTEMS CONFERENCE (BIOCAS), 2015 IEEE. **Proceedings...** Atlanta, GA: IEEE, 2015. p. 1–4.

WANG, X. et al. Formation control for multiple mobile robots based on the spiking neural network. In: ROBOTICS AUTOMATION AND MECHATRONICS (RAM), 2010 IEEE CONFERENCE ON. **Proceedings...** Singapore: IEEE, 2010. p. 447–452.

YAMAZAKI, T.; TANAKA, S. The cerebellum as a liquid state machine. **Neural Networks**, v. 20, n. 3, p. 290 – 297, 2007. ISSN 0893-6080. Echo State Networks and Liquid State Machines. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0893608007000366>.

ZAMARRENO-RAMOS, C. et al. Multicasting mesh aer: A scalable assembly approach for reconfigurable neuromorphic structured aer systems. application to convnets. **IEEE Transactions on Biomedical Circuits and Systems**, v. 7, n. 1, p. 82–102, Feb 2013. ISSN 1932-4545.

ZEFERINO, C. A. **Redes-em-chip: Arquiteturas e Modelos para Avaliação de Área e Desempenho**. Thesis (PhD) — Universidade Federal do Rio Grande do Sul, 2003.

ZEFERINO, C. A.; KREUTZ, M. E.; SUSIN, A. A. Rasoc: a router soft-core for networks-on-chip. In: DESIGN, AUTOMATION AND TEST IN EUROPE CONFERENCE AND EXHIBITION, 2004. **Proceedings...** [S.l.]: IEEE, 2004. v. 3, p. 198–203 Vol.3. ISSN 1530-1591.

ZEFERINO, C. A.; SUSIN, A. A. Socin: a parametric and scalable network-on-chip. In: 16th Symposium on Integrated Circuits and Systems Design, 2003. SBCCI 2003. **Proceedings...** São Paulo, Brazil: IEEE, 2003. p. 169–174.

ZHANG, Y. et al. A digital liquid state machine with biologically inspired learning and its application to speech recognition. **Neural Networks and Learning Systems, IEEE Transactions on**, v. 26, n. 11, p. 2635–2649, Nov 2015. ISSN 2162-237X.

APPENDIX A — RESUMO EM PORTUGUÊS

DHyANA: uma Arquitetura Digital Neuromórfica Hierárquica para Máquinas de Estado Líquido.

A.1 Resumo

Redes Neurais têm sido um tema de pesquisas por pelo menos sessenta anos. Desde a eficácia no processamento de informações à incrível capacidade de tolerar falhas, são incontáveis os mecanismos no cérebro que nos fascinam. Assim, não é nenhuma surpresa que, na medida que tecnologias facilitadoras tornam-se disponíveis, cientistas e engenheiros têm aumentado os esforços para o compreender e simular.

Em uma abordagem semelhante à do Projeto Genoma Humano, a busca por tecnologias inovadoras na área deu origem a projetos internacionais que custam bilhões de dólares, o que alguns denominam o despertar global de pesquisa da neurociência.

Avanços em hardware fizeram a simulação de milhões ou até bilhões de neurônios possível. No entanto, as abordagens existentes ainda não são capazes de fornecer a densidade de conexões necessária ao enorme número de neurônios e sinapses.

Neste sentido, este trabalho propõe DHyANA (Arquitetura Digital Neuromórfica Hierárquica), uma nova arquitetura em hardware para redes neurais pulsadas, a qual utiliza comunicação em rede-em-chip hierárquica. A arquitetura é otimizada para implementações de Máquinas de Estado Líquido.

A arquitetura DHyANA foi exaustivamente testada em plataformas de simulação, bem como implementada em uma FPGA Stratix IV da Altera. Além disso, foi realizada a síntese lógica em tecnologia 65nm, a fim de melhor avaliar e comparar o sistema resultante com projetos similares, alcançando uma área de 0,23mm^2 e potência de 147mW para uma implementação de 256 neurônios.

A.2 Introdução

A.2.1 Redes Neurais Pulsadas

Um neurônio é uma célula extremamente especializada, que gera sinais elétricos em resposta a agentes químicos e elétricos, e dissemina-os através de seus axônios para outras células (DAYAN; ABBOTT, 2005). Ele é composto principalmente pelo corpo celular, ou soma, os dendritos e o axônio, o qual pode atravessar grandes porções do cérebro ou mesmo de todo o corpo. Suas conexões, que podem ser vistas como um tipo de resistor químico, são denominadas sinapses.

O potencial elétrico entre o meio intra- e extracelular de um neurônio, que em condições de repouso (polarizadas) é de cerca de -70mV, é o sinal relevante para o sistema nervoso. Este potencial é modificado por correntes elétricas inseridas através das mais de 10.000 entradas presentes em um neurônio comum. Tais alterações no potencial elétrico transmembrana, denominadas Potenciais Pós-Sinápticos (PSP, do inglês *postsynaptic potential*), podem ser grandes o bastante para, sob determinadas circunstâncias, serem amplificadas pelos canais sensíveis a tensão embutidos na membrana neuronal, gerando

assim um potencial de ação ou espiga.

Existem diversos modelos de neurônios pulsados, cada um variando em níveis de complexidade, intensidade computacional e precisão biológica, e uma boa visão geral deles pode ser visto em (IZHIKEVICH, 2004). O mais biologicamente preciso e um dos modelos matemáticos mais utilizados do neurônio é o desenvolvido por Hodgkin e Huxley (1952). Ao pesquisar o comportamento de neurônios de lula (cujo tamanho é 100 a 1000 vezes maior do que aqueles no cérebro humano), eles desenvolveram um modelo matemático que pode reproduzir todos os tipos de neurônios com boa precisão em termos de forma de espiga e atividades de disparo complexas (AMBROISE et al., 2013).

Um dos modelos mais simples é o Integra-e-Dispara com Vazamento (LIF, do inglês *Leaky Integrate-and-Fire*), o qual utiliza apenas uma equação diferencial para modelar o comportamento de um neurônio. Este modelo basicamente idealiza um neurônio como tendo corrente ôhmica de fuga e correntes reguladas por tensão desativadas quando em repouso. Apesar dos benefícios relativos à simplificação, este modelo é falho ao não ser tão biologicamente plausível.

Outro modelo matemático de neurônio comumente utilizado foi proposto por Izhikevich (2003), o qual pode ser considerado um meio termo entre complexidade e plausibilidade biológica. Apesar de não ser tão simples quanto o modelo LIF, modelo de Izhikevich pode simular várias respostas não-lineares de neurônios biológicos, tornando-o quase tão versátil quanto o modelo de Hodgkin-Huxley em uma fração de seu custo computacional, sendo este, portanto, o modelo escolhido para este trabalho.

O modelo de neurônio idealizado por Izhikevich é descrito pelas duas equações diferenciais acopladas a seguir.

$$\frac{dv}{dt} = 0.04v^2 + 5v + 140 - u + I_{Izh} \tag{A.1}$$

$$\frac{du}{dt} = a(bu - v) \tag{A.2}$$

A variável *v* denota o potencial da membrana, enquanto *u* representa o parâmetro de recuperação da membrana. *I* é a entrada sináptica, e *a*, *b* são parâmetros que controlam o comportamento dinâmico do modelo neural. Há também uma condição de reset, controlada pelos parâmetros *c*, *d*, e definida pela equação (A.3).

$$v \geq 30mV \Rightarrow \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases} \tag{A.3}$$

A.2.2 Máquinas de Estado Líquido

O conceito de Computação por Reservatórios foi introduzido em duas publicações independentes, as quais, embora usando abordagens diferentes, apresentaram a ideia de um reservatório dinâmico recorrente de unidades de processamento em estado transitório, enquanto a camada de saída está sujeita a um aprendizado supervisionado. A primeira publicação, por Jaeger (2001a), as Redes com Estado de Eco foram introduzidas. O estudo explorou aplicações de redes recorrentes de neurônios sigmoidais conectados aleatoriamente a tarefas complexas de previsão de séries temporais. A segunda, por Maass, Natschläger e Markram (2002), uma abordagem mais biologicamente orientada, considerou reservatórios de neurônios pulsados estruturados e funcionando de uma forma inspirada

pelas propriedades de colunas neocorticais do sistema nervoso central de mamíferos.

A motivação intuitiva por trás deste conceito pode ser explicada imaginando o uso de um líquido (como um copo de água) para executar a computação, como explicado por Cao e Pipa (2010). Do ponto de vista de um sistema dinâmico, isso não faz muito sentido, pois o único estado estável ao qual o líquido pode convergir após uma perturbação (por exemplo, uma queda no líquido) é um estado "morto" em que o líquido estará perfeitamente imóvel. Entretanto, a ideia de Computação por Reservatórios é que os estados transitórios do líquido em determinado momento ainda mantêm informações relevantes sobre uma perturbação que ocorreu no momento anterior. Uma vez que o líquido tenha a capacidade de produzir estados transitórios significativamente diferentes sob duas entradas diferentes, é então teoricamente possível extrair informação diretamente de medidas sobre o estado do líquido, uma vez que este pode armazenar informações sobre o passado (por exemplo, imagens sucessivas de um copo de água previamente perturbado).

A Máquina de Estado Líquido de Maass, Natschläger e Markram (2002) permite a realização de computações exigentes com circuitos de neurônios pulsados e sinapses dinâmicas, e, por ser mais biologicamente orientada, foi escolhida como foco deste trabalho. Ela foi motivada pela hipótese de que a capacidade de aprendizagem de um dispositivo de processamento de informação é o seu aspecto mais delicado, e que a disponibilidade de exemplos de treinamento suficientes é um gargalo primário para a aprendizagem orientada por objetivos (isto é, supervisionado ou baseado em recompensas) (MAASS, 2011).

Sua arquitetura é composta por duas camadas. Uma camada é um reservatório de neurônios pulsados interagindo em uma topologia recorrente que mapeia entradas em um estado dinâmico, gerando um padrão de atividades espaço-temporais de neurônios chamado "estado líquido". A rede tem de gerar estados líquidos diferentes para diferentes sinais de entrada, uma vez que o desempenho de uma Máquina de Estado Líquido depende principalmente da qualidade dos padrões de atividade. A outra camada consiste de neurônios de "leitura", que recebem o estado líquido e os sinais de instrução, comparam sua saída com uma saída alvo no procedimento de treinamento e adaptam seus pesos sinápticos utilizando uma regra de aprendizado. Deste modo, uma vez que apenas a camada de leitura precisa ser treinada, e não toda a rede neural recorrente, a aprendizagem é realizada de forma rápida e robusta.

A.2.3 Redes em Chip

A crescente complexidade e quantidade de transistores em dispositivos microeletrônicos, nos quais centenas de núcleos de processamento (*IP cores*) são necessários para executar múltiplos processos simultâneos em um único chip, colocou tremenda pressão sobre a arquitetura de comunicação em Sistemas em Chip (SoCs, do inglês *Systems-on-Chip*), mudando toda sua metodologia de projetos, agora baseada em comunicação.

Os núcleos dos sistemas foram tradicionalmente interligados utilizando-se dois diferentes esquemas de comunicação: canais multipontos compartilhados, ou barramentos, e canais ponto-a-ponto (P2P) dedicados. As arquiteturas de comunicação P2P podem fornecer o máximo em desempenho de comunicação, porém com o curso de utilização de canais dedicados entre todos os pares de comunicação, o que gera alta complexidade, custo e esforço de design em termos de escalabilidade. Por outro lado, as arquiteturas baseadas em barramento podem conectar algumas dezenas de núcleos de forma econômica, reduzindo a complexidade do projeto e eliminando os fios dedicados exigidos pelas arquiteturas P2P. Entretanto, os barramentos não conseguem fornecer escalabilidade o

suficiente para energia e desempenho dos sistemas.

A abordagem baseada em Redes-em-Chip (NoCs, do inglês *Networks-on-Chip*) representa uma solução promissora para os problemas de comunicação intra-chip, sendo melhor escaláveis em termos de área, desempenho, consumo de energia e esforço de projeto global. Uma rede é constituída por roteadores, interconectados por enlaces (*links*) ponto a ponto, formando assim sua topologia. Os roteadores estão ligados aos elementos de processamento que constituem o sistema através de uma interface de rede (NI, do inglês *network interface*). Assim, a informação gerada por um núcleo de processamento, que pode ser dividida em partes menores, ou pacotes, é enviada pela rede através do roteador, anexado ao núcleo por meio do NI (OGRAS; MARCULESCU, 2013).

A capacidade da rede para disseminar eficientemente as informações depende em grande parte da sua topologia, a qual possui um efeito primordial na largura de banda da rede, latência, *throughput*, área total, tolerância a falhas e consumo de energia, bem como um papel importante na concepção da estratégia de roteamento e de mapeamento dos núcleos da rede (OGRAS; MARCULESCU, 2013). Existe um vasto espaço de parâmetros quando se trata da infra-estrutura de comunicação do NoC.

O design do roteador envolve a determinação das técnicas de controle de fluxo, número de canais virtuais, organização de buffer, design de switch, estratégia de pipelining enquanto aderindo à freqüência de clock de destino e orçamentos de energia. Cada parâmetro afeta os sistemas em diferentes métricas, como desempenho, consumo de energia e área, sendo assim tarefa fundamental no projeto de NoCs explorar eficientemente o espaço de design em relação a todos esses aspectos. Para uma explicação detalhada de todos os parâmetros importantes para uma infra-estrutura de rede-em-chip, o leitor é indicado às pesquisas de Bjerregaard e Mahadevan (2006) e de Ogras e Marculescu (2013).

A.2.4 Estado da Arte

Existe uma vasta gama de Redes Neurais Pulsadas encontrada na literatura. Várias destas redes foram desenvolvidos em hardware, como demonstram as pesquisas de Maguire et al. (2007), de Misra e Saha (2010) e de Cassidy, Georgiou e Andreou (2013). As arquiteturas propostas mais relevantes para este trabalho são os recentes esforços neuromórficos altamente escaláveis, os quais foram implementados em hardware e utilizam estratégias de rede em chip como principal estrutura de comunicação.

Os projetos Neurogrid da Universidade de Stanford (BENJAMIN et al., 2014), e BrainScales da Universidade de Heidelberg (SCHEMMEL et al., 2010), são formados por sistemas multi-chip neuromórficos de sinal misto. Ambos os projetos, embora diferentes em muitos aspectos, usam computação analógica para emular dinâmicas neurais e comunicação digital para realizar as conexões sinápticas.

O projeto SpiNNaker, da Universidade de Manchester, tem como objetivo fornecer uma plataforma para o processamento paralelo de alta performance, integrando um sistema baseado em microprocessador, o qual contem 18 núcleos ARM968 em um único *die* (FURBER et al., 2013). Cada nó do sistema consiste em um Sistema-em-Pacote, o qual contém um processador de núcleos ARM e uma SDRAM *off-die* de 128Mbyte empilhada em seu topo. Os nós do sistema, formados por Sistemas-em-Pacote, são interligados por uma rede em chip contendo seis links, envolvidos em uma rede triangular.

O projeto EMBRACE da Universidade de Ulster e da Universidade Nacional da Irlanda (HARKIN et al., 2009), e o processador neuromórfico ROLLS da Universidade de Zurique e da ETH Zurique (QIAO et al., 2015) são abordagens de sinal misto de baixa

potência. A arquitetura EMBRACE propõe RNPs baseadas em uma matriz hierárquica de roteadores, com base em uma topologia híbrida malha-estrela. Já o processador ROLLS usa circuitos de lógica digital assíncrono para configuração de diferentes configurações de rede.

O projeto IBM SyNAPSE desenvolveu o chip TrueNorth, no qual 4096 núcleos neurossinápticos formam uma matriz bidimensional, formando um sistema com 1 milhão de neurônios digitais e 256 milhões de sinapses. Sua arquitetura também utiliza a comunicação hierárquica, contendo um *crossbar* de alto *fanout* para comunicação local e uma rede em chip para comunicação de longa distância. O sistema possui também sincronização global do sistema para garantir a operação em tempo real.

A.3 Arquitetura Proposta

DHyANA é uma Rede Neural composta por uma rede em chip hierárquica em dois níveis. Os núcleos de processamento, modelos de neurônio de Izhikevich digitais, bem como um Controlador são conectados localmente através de um barramento compartilhado no nível de Cluster. A interface entre os dois níveis é realizada através de uma conexão entre o Controlador e seu Roteador correspondente. No nível Global, uma rede-em-chip bidimensional do tipo grelha (malha, ou *mesh*) realiza a conexão entre Clusters de neurônios.

A visão de um sistema no Nível Global está disponível na Figura A.1. Ele mostra o tamanho do protótipo do sistema para a comunicação global, uma topologia de malha 4x4.

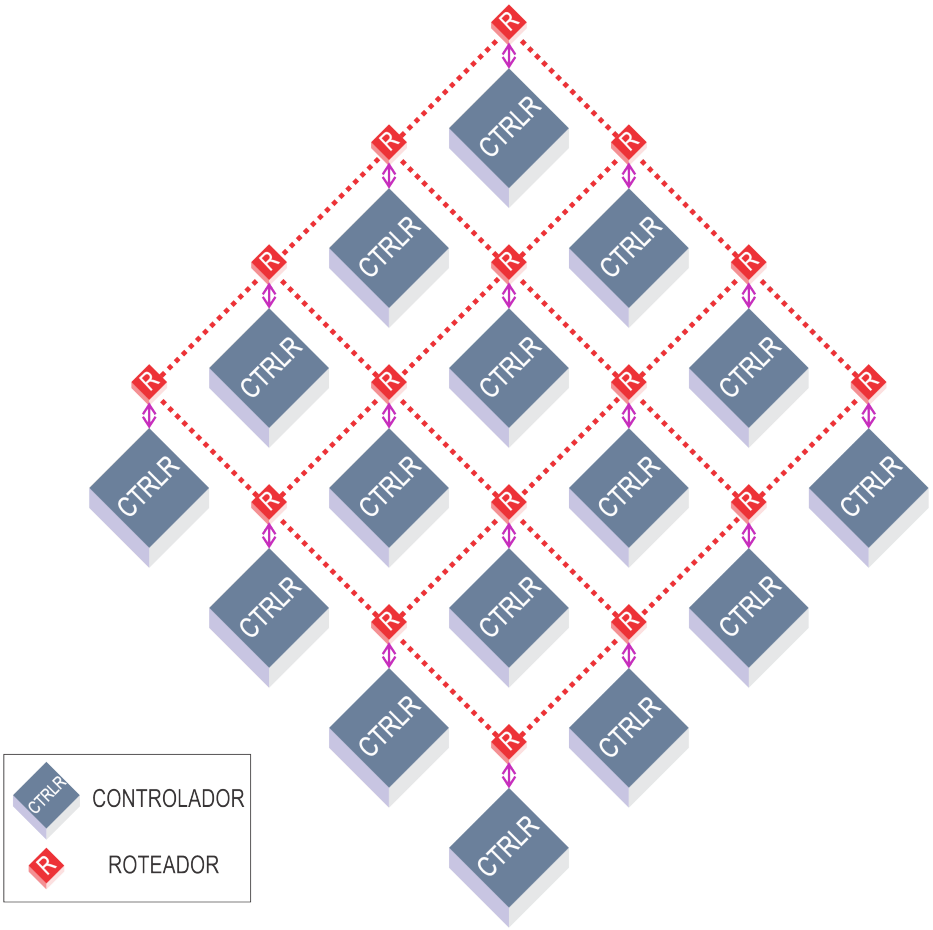
A motivação para a escolha de uma topologia híbrida foi impulsionada por alguns pontos importantes: simplicidade, desempenho, otimizações de energia e, principalmente, localidade. Em termos de chips multiprocessadores, o conceito de localidade foi definido por Das et al. (2009) como "a porcentagem de pacotes injetados por um nó que são satisfeitos por seus vizinhos imediatos na rede". Assim, aplicações com localidade alta tendem a ter alto tráfego local no padrão de comunicação com vizinhos próximos. Em sistemas biológicos neuronais, o conceito é semelhante. Na escala de aproximadamente um milímetro cúbico de córtex, contendo cerca de 100.000 neurônios, o trabalho de Mehring et al. (2003) sugere que a probabilidade de conexão diminui de uma forma Gaussiana com a distância entre neurônios.

Assim, com base nas premissas de localidade supramencionadas, pode-se inferir que a comunicação inter-cluster (global) possui menor atividade durante a execução em comparação com a comunicação intra-cluster (local), sendo o barramento compartilhado uma escolha conveniente para tal configuração, além de ser esta uma solução simples e eficiente em termos de energia. Porém, estas poucas mensagens inter-cluster exigem um alto *throughput* para lidar com rajadas de disparos dos neurônios, sendo então a topologia de rede em grelha uma ótima opção para evitar estrangulamentos independentemente da taxa de mensagem, além de ser uma opção viável em termos de escalabilidade.

A.3.1 Nível Global

A rede-em-chip em grelha utilizada para o Nível Global de DHyANA foi a *System-on-Chip Interconnection Network* (SoCIN) (REINBRECHT, 2012; ZEFERINO, 2003), uma rede escalável baseada em uma arquitetura de roteador parametrizável para NoCs de baixo custo. A rede, baseada no algoritmo de roteamento XY, emprega chaveamento

Figure A.1: Nível Global da DHyANA. Por questões de clareza visual, o Nível de Cluster é omitido, como se estivesse sob os Controladores.



Fonte: A autora

por pacotes do tipo *wormhole* e controle de fluxo por handshake, arbitragem dinâmica distribuída e memorização por FIFOs na entrada.

O enlace do Nível Global de DHyANA é composto por dois canais unidirecionais *simplex*, cada um dos quais formado por n bits de dados e dois bits de banda lateral, utilizados para indicar, respectivamente, o começo do pacote (bop - *begin-of-packet*) e o fim do pacote (eop - *end-of-packet*). Um flit no sistema é formado pelo tamanho da largura física do canal, possuindo assim $n + 2$ bits. Neste trabalho $n = 8$, portanto um flit possui 10 bits de largura. Além disso, cada canal inclui um par de sinais necessários para controlar o fluxo, que são utilizados para validar dados no canal (val) e para reconhecer os dados recebidos (ack).

Os pacotes no nível global são compostos por três flits: o cabeçalho (*header*; o endereço do neurônio (NA, do inglês *neuron address*); e o flit de término (cauda ou *tail*). O cabeçalho é o primeiro flit do pacote e inclui as informações necessárias para estabelecer o caminho do pacote na rede. O flit NA contém o endereço dos neurônios dentro de seu respectivo Cluster. A cauda, flit que define o final do pacote, é utilizada para recuperar o endereço do Cluster do neurônio que originou a espiga.

Além dos dois bits de banda lateral, o restante do flit de cabeçalho é reservado para

os bits de Informação de Rota (RI, do inglês *Routing Information*), contendo informações necessárias para o roteamento do pacote. O RI é formado por quatro campos: Xdir, Xmod, Ydir e Ymod. A descrição de cada campo está incluída na Tabela A.1.

Table A.1: Descrição dos bits RI

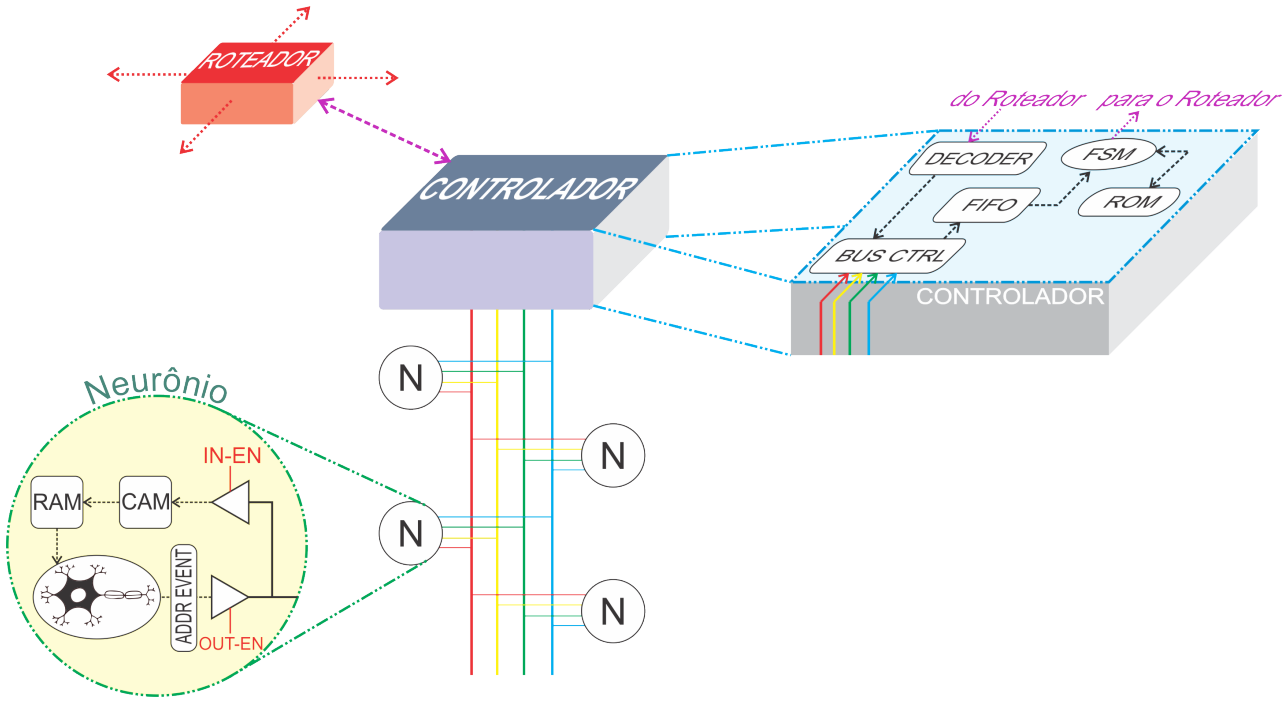
Campo	Significado
Xdir	O pacote deve ser encaminhado na direção Leste/Oeste
Xmod	Número de enlaces restantes a serem percorridos na direção X (East/West)
Ydir	O pacote deve ser encaminhado na direção Norte/Sul
Ymod	Número de enlaces restantes a serem percorridos na direção Y (Norte/Sul)

Fonte: Adaptado de Zeferino (2003)

A.3.2 Nível Local

O Nível Local é responsável por duas tarefas muito importantes dentro da arquitetura DHyANA: conectar os neurônios dentro do mesmo *cluster*, além de construir os pacotes para a comunicação entre os níveis na hierarquia. Ele é composto principalmente, conforme mostrado na Figura A.2, pelos módulos Neurônio e Controlador Local.

Figure A.2: Nível Local da DHyANA.



Fonte: A autora

O módulo Neurônio é formado pelo Izhikevich *Simple Model* (ISM), modelo de neurônio utilizado neste trabalho, além de um Codificador de Endereço-Evento (AE, do inglês *Address-Event*), e duas memórias que juntas simulam as sinapses.

Uma vez que um neurônio dispara, o Codificador AE é responsável por enviar o endereço do neurônio para o barramento. O módulo utiliza a abordagem denominada Representação de Endereço-Evento (AER, de *Address Event Representation*), na qual um endereço é atribuído a cada célula de neurônio no chip, de modo que quando uma espiga ocorre, seu endereço é transmitido para todos os determinados nós computacionais (MAHOWALD, 1992; SIVILOTTI, 1991).

As duas células de memória no módulo Neuron são uma Memória Endereçável por Conteúdo (CAM, do inglês *Content Addressable Memory*) e uma Memória de Acesso Aleatório (RAM, do ingês *Random Access Memory*). As duas memórias representam as sinapses da seguinte maneira: a memória CAM testa conectividade sobre um AE sendo transmitido no barramento; em casos positivos, envia para a RAM o endereço no qual está armazenado o valor daquela determinada sinapse (corrente vezes peso); Os valores de inicialização e arquivos *.mif* para as memórias são gerados aleatoriamente de modo offline por um programa em Python, seguindo as restrições impostas pela rede desejada a ser implementada.

A saída de RAM é conectada ao sinal de entrada de 18 bits *I* do ISM. No entanto, os dois módulos foram concebidos para ter domínios de relógio diferentes (com o ISM sendo muito mais lento do que o resto do sistema). Assim, a fim de evitar perda de dados e metaestabilidade, um registador de deslocamento de dois estágios foi implementado na interface entre os dois módulos.

O Controlador Local é composto por uma memória ROM, um buffer FIFO, uma Máquina de Estados Finitos, um Decodificador e um Controlador de Barramento.

O Controlador de Barramento é utilizado para anular a possibilidade de dois neurônios enviarem seu AE simultaneamente para o barramento, através de um protocolo *handshake* e um codificador de prioridade. Além disso uma FIFO, com profundidade 4 neste trabalho, foi implementada para lidar com alta taxa de transferência e evitar a perda de dados.

A ROM contém os seguintes dados: um FLAG para cada neurônio dentro de um Cluster, contendo o número de Clusters com que se conecta; e os bits RI, dados necessários para montar o cabeçalho do pacote Global. Seus valores são gerados off-line por um programa em Python, o qual define as conexões da rede e, em seguida, monta o RI para o pacote Global de acordo com as premissas descritas na Tabela A.1.

A Máquina de Estados Finitos é o módulo que controla a comunicação entre o Controlador Local e o Roteador Global. Ela monta o pacote Global e mantém os controles de temporização e sincronismo para a transmissão de cada flit ser bem-sucedida.

O módulo Decodificador interpreta a informação recebida de outro Cluster para torná-lo legível para os neurônios de um determinado Cluster. Para isto, aproveitou-se o fato de que o flit de término não é modificado ao longo do caminho na rede, ao contrário do flit de cabeçalho. Assim, quando um novo pacote chega através do Roteador Global, os flits NA e *tail* vão para o Decodificador, o qual recupera o AE original do neurônio que disparou e permitindo assim que este seja distribuído no barramento para que cada módulo Neurônio teste sua conectividade.

A.3.3 Núcleos de Processamento: Neurônios

Para DHyANA, foi utilizado o modelo de neurônio Izhikevich *Simple Model* (ISM) implementado digitalmente em hardware, proposto em (BANDEIRA et al., 2015). A escolha do modelo de Izhikevich deveu-se principalmente ao fato de sua implementação se encontrar no meio-termo entre complexidade e plausibilidade biológica. Tal escolha, porém, vem sem perda de generalidade. Assim, sempre que se verifique que um outro modelo de neurônio é mais adequado para uma determinada aplicação, este pode ser incorporado ao sistema.

Para sua implementação, foi escolhida uma representação de em ponto fixo e complemento de dois, uma abordagem mais adequada para implementações digitais do que ponto flutuante. Os sinais, inclusive de entrada e saída, possuem 18 bits, quantidade adequada para permitir o uso do hardware disponível sem comprometer a precisão.

Cada neurônio recebe os parâmetros (*a*, b*, c, d**), valor modificados das equações originais de Izhikevich, de um módulo superior, e armazena localmente os valores para *v* e *u**. Os valores iniciais de *v* e *u** são, respectivamente, -70 mV e -15.63 mV. O tempo incremental é $\Delta t = h = 0.78125$ ms (milissegundos). Os parâmetros e as correntes injetadas são exibidos na Tabela A.2

Table A.2: Parâmetros para cada característica neurocomputacional e corrente injetada utilizada na implementação.

		Comportamento Neural					
		Spike Tônico	Spike Fásico	Rajada Tônico	Rajada Fásico	Modo Misto	Spike com Freq. Adaptativa
Entrada	I*	10.9375	5	11.71875	4.6875	9.375	23.4375
Parametros	a*	0.015625	0.015625	0.015625	0.015625	0.015625	0.0078125
	b*	0.15625	0.1953125	0.15625	0.1953125	0.1953125	0.1953125
	c	-65	-65	-50	-55	-55	-65
	d*	4.6875	4.6875	1.56125	0.0390625	3.125	6.25

Fonte: Bandeira et al. (2015)

A.4 Resultados

Sem perda de generalidade, um protótipo de NoC 4 x 4 com um número variável de neurônios em cada cluster foi descrito em linguagem mista, VHDL e Verilog. A validação da arquitetura proposta foi realizada em duas etapas: avaliação funcional de hardware através de simulação e implementação em FPGA para análise de desempenho em tempo real. Ademais, foi realizada uma síntese lógica de 65 nm para avaliação e comparação com projetos semelhantes do estado-da-arte.

A avaliação funcional da arquitetura DHyANA foi realizada utilizando-se o Modelsim SE 10.1c da Mentor Graphics e o Quartus Prime 16.0 da Altera. A validação foi realizada em diferentes estágios: primeiro a comunicação entre neurônios por conexão direta foi simulada; então, foram testados Clusters de diferentes tamanhos; posteriormente

foi estabelecida a conexão inter-nível (Controlador Local para Roteador Global) para verificar os protocolos de comunicação; e por fim, Clusters com tamanhos diferentes foram conectados a Roteadores Globais e a funcionalidade da arquitetura foi completamente verificada.

Após o cuidadosos testes funcionais e avaliações da arquitetura proposta, a arquitetura DHyANA foi implementada em uma FPGA Altera Stratix IV EP4SGX230KF40C2, a fim de se verificar o desempenho em tempo real do sistema em uma plataforma de hardware. Foram analisadas diferentes configurações da arquitetura, tendo-se observado um crescimento linear na utilização da FPGA com o aumento de um neurônio por Cluster. A FPGA foi capaz de admitir até 13 neurônios por Cluster (99% de sua capacidade de utilização lógica), um total de 208 neurônios no sistema. Para fins de comparação, o mesmo chip FPGA pôde ser preenchido com até 364 ISMs em (BANDEIRA et al., 2015).

Tal número é considerado bom o suficiente para algumas tarefas de classificação, como o reconhecimento da palavras faladas em (HOPFIELD; BRODY, 2001) e outras implementações de Máquinas de Estado Líquido, uma vez que o artigo que marcou sua origem implementou tais aplicações utilizando-se 135 neurônios Integra-e-Dispara para o líquido e mais 51 neurônios de leitura.

Um crescimento linear semelhante é também esperado para configurações de Clusters mais numerosos. Entretanto Clusters superiores a 20 ou 25 neurônios podem vir a apresentar problemas de temporização, sendo assim vantajoso considerar-se o uso de outras topologias de rede para o nível mais baixo da hierarquia, como o *crossbar* em (REINBRECHT, 2012), a estrela como em (CARRILLO et al., 2013), a grelha como sugerido em (VAINBRAND; GINOSAR, 2011) ou até a criação de um terceiro nível na hierarquia.

Uma síntese lógica de 65nm foi também realizada, utilizando-se ferramentas da Cadence Design Systems, para 256 neurônios, (rede em grelha 4 x 4 com 16 neurônios por Cluster), a fim de se avaliar e ter meios de comparação com trabalhos similares. Para a configuração estabelecida, a síntese estabeleceu uma potência de 147mW para o sistema, e uma área de chip de 0.23mm^2.

A tabela A.3 resume alguns dos recursos presentes na arquitetura DHyANA, bem como dados de outras plataformas neuromórficas já citadas neste trabalho. Vale ressaltar, no entanto, que, embora relacionadas, a maioria das medições não utilizam os mesmos *benchmarks* nem utilizam a mesma tecnologia, portanto, não podem ser comparadas diretamente.

Table A.3: DHyANA compada a Arquiteturas Relacionadas

Projeto	Modelo de Neurônio	A/D	Neurônios	Potência	Potência/ Neurônio	Área mm^2	Área/ Neurônio	Processo CMOS	Synapses por Neurônio	Topologia
SpiNNaker	IF or IZHI	DS	16x10^3	1W	6,25E-05	102	6,38E-03	130nm	1x10^3	Treliça triangular*
Neurogrid	Quad. IF	A	983,040	5W	5,08E-06	168	1,71E-04	180nm	6x10^9	Estrela
BrainScaleS	Exp. IF	A	512	1kW	1,95	430	8,40E-01	180nm	112x10^3	Barramentos hierárquicos**
EMBRACE	IF	A	400	13.16mW	3,29E-05	0.587	1,47E-03	65nm	400	Híbrida (estrela-malha)
TrueNorth	-	-	1x10^6	65mW	5,60E-08	430	4,30E-04	28nm	256	Malha
ROLLS	Exp. IF	A	256	4mW	1,56E-05	51.4	2,01E-01	180nm	256	Matriz de comutação programável
DHyANA	IZHI	D	256	147mW	5,74E-04	0.23	8,98E-04	65nm	256	Híbrida (barramento-malha)

A = Analógico, D = Digital, DS = Digital (Software), IF = Integra-e-Dispara, IZHI = Izhikevich. *dobrado em uma superfície toroidal. **2D Toro (wafer). SpiNNaker, BrainScaleS, TrueNorth dados por chip; EMBRACE data por cluster.

Fonte: A autora

A.5 Conclusão e Trabalhos Futuros

Neste trabalho foi proposta DHyANA, uma Arquitetura Digital Neuromórfica Hierárquica. Com foco em implementações de Redes Neurais de Máquina de Estado Líquido, a arquitetura utiliza uma abordagem de rede em chip hierárquica para melhorar a escalabilidade da comunicação e do sistema.

Diferentes configurações do sistema foram implementadas, e os resultados experimentais mostraram que a utilização da FPGA cresceu linearmente com o número de neurônios por Cluster. A FPGA foi capaz de comportar até 13 neurônios por Cluster, 208 neurônios no total. Uma configuração de 256 neurônios, 16 por Cluster, foi sintetizada, alcançando uma área de 0,23mm^2 e uma dissipação de potência de 147mW.

Assim, embora não tenha alcançado grandes melhorias em termos de área e potência em relação ao estado da arte o projeto demonstrou grande potencial, a ser explorado em muitos aspectos diferentes em pesquisas futuras.

Para que o sistema se torne mais biologicamente realista, é importante investigar melhorias futuras. Por exemplo, a implementação de modelos de neurônios ainda mais complexos, como o modelo Hodgkin-Huxley, traria mais realismo, embora a custo de se aumentar área e potência. A modelagem da plasticidade sináptica e outros detalhes nos mecanismos dos neurônios também melhorariam o sistema nessa direção.

Além disso, algumas mudanças na arquitetura também trariam avanços para o sistema. Uma comunicação *multicasting* para o Nível Global, como em (ZAMARRENO-RAMOS et al., 2013), poderia trazer grandes melhorias na velocidade e *throughput*, mas só poderia ser implementada após análise cuidadosa da viabilidade do novo tráfego do sistema. Ademais, como já mencionado, a implementação de outra camada de hierarquia pode ser vantajosa, ou até mesmo necessária, para expansões do sistema. Expansões também poderiam se beneficiar da tecnologia de redes 3D, utilizando-se por exemplo a topologia em grelha de (MATOS et al., 2013; MATOS et al., 2015) e de (MARCON et al., 2014).

O desenvolvimento e futuras melhorias deste trabalho tem por objetivo abrir um grande potencial para pesquisas futuras, por exemplo na utilização do sistema para desenvolvimento de aplicações em engenharia e biologia, incluindo Interfaces Cérebro-Máquina, neuropróteses, reconhecimento de voz com grandes vocabulários, veículos autodirigidos, controle de robôs autônomos, processamento de sinais médicos, bem como experiências e análises de dados em neurofisiologia.