NEWCASTLE UNIVERSITY

Neural Networks-on-Chip for Hybrid Bio-Electronic Systems

by

Graeme Coapes

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the

Faculty of Science, Agriculture and Engineering
School of Electrical and Electronic Engineering

July 2016

"I cannot live without brain-work. What else is there to live for?."

The Sign of Four, Sherlock Holmes

Sir Arthur Conan Doyle

Abstract

By modelling the brains computation we can further our understanding of its function and develop novel treatments for neurological disorders. The brain is incredibly powerful and energy efficient, but its computation does not fit well with the traditional computer architecture developed over the previous 70 years. Therefore, there is growing research focus in developing alternative computing technologies to enhance our neural modelling capability, with the expectation that the technology in itself will also benefit from increased awareness of neural computational paradigms.

This thesis focuses upon developing a methodology to study the design of neural computing systems, with an emphasis on studying systems suitable for biomedical experiments. The methodology allows for the design to be optimized according to the application. For example, different case studies highlight how to reduce energy consumption, reduce silicon area, or to increase network throughput.

High performance processing cores are presented for both Hodgkin-Huxley and Izhikevich neurons incorporating novel design features. Further, a complete energy/area model for a neural-network-on-chip is derived, which is used in two exemplar case-studies: a cortical neural circuit to benchmark typical system performance, illustrating how a 65,000 neuron network could be processed in real-time within a 100mW power budget; and a scalable high-performance processing platform for a cerebellar neural prosthesis. From these case-studies, the contribution of network granularity towards optimal neural-network-on-chip performance is explored.

Acknowledgements

First and foremost I would like to thank my project supervisor, Dr Patrick Degenaar, for his guidance and wisdom throughout the course of my studies. His management and leadership of the Neuroprosthesis Lab has developed it from nothing to a multi-million pound research facility.

My thanks must also go to Prof. Alex Yakovlev for his leadership of the group and his assistance whenever anything is asked of him, and to Dr. Terrence Mak for starting me along this path.

The support of staff and PhD students throughout the School has been invaluable; particulaly the research collaboration with Jun Wen Luo and the lunchtime club of Lucy, Kartheek, Sandip and many others.

Laura, my beautiful wife to be, has provided many moments of inspiration, often without realising, that have contributed hugely to this thesis.

Finally, thanks must go to my family, who have always supported and believed in me, particularly my Mum and Dad.

Contents

List of Figures

List of Tables

Symbol	Description
	Hodgkin-Huxley Equation Symbols
C	Capacitance of neuron's membrane
$E_K, E_l, E_N a$	Potential of ion channel when at resting state
$g_K g_l, g_N a$	Conductance of the ion channel, either Potassium, Leak or Sodium channel
h, m, n	Parameter defining how open/closed the ion channel is
$I_K, I_l, I_N a$	Current through the ion channel
V_m	Membrane voltage of the neuron
	Izhikevich Equation Symbols
a, b, c, d	Parameters affecting neuron configuration
I	Injected input current to neuron
u	Intermediate voltage level
v	Neuron output voltage
	Neural-Network-on-Chip Symbols
a	Silicon area required to implement a single neuron
c	Manhatten distance between processing cores within a 2-dimensional mesh network-on-chip
C_{mesh}	The capacitance of all physical wires connecting each processing core within the network-on-chip
C_0	Technology dependent parameter defining average capactiance of wire per unit length
$\bar{\delta_n}$	Average rate at which neurons produce spikes
D_{ave}	Average Manhatten distance that each spike packets travel through the network-on-chip to reach its destination
$\bar{f_{s_{max}}}$	Throughput of packets that will saturate the bottleneck channel within a network-on-chip

f_n	Mean operating frequency of link between two cores within a network-on-chip
H_0	The distance travelled by each packet to reach its destination assuming no contention
k	Number of rows/columns in a 2-dimensional mesh network-on-chip
L	Number of interconnections between processing cores within network-on-chip
$\bar{l}$	The length of physical wire connecting each processing core within the network-on-chip
l_{mesh}	The total physical length of all wires connecting each processing core within the network-on-chip
m	Total number of processing cores within network-on-chip
n	Granularity - Number of neurons allocated to each processing core
$\bar{N}_d$	Average degree of connectivity between neurons
P_{arch}	The theoretical power consumption of communication between processing cores within the network-on-chip
$R_\square$	Technology dependent parameter defining average resistance of wire per unit length
s	Total number of neurons within neural network model
T_0	The latency, in clock cycles, of each packet arriving at its destination assuming no contenion
U_{arch}	The utilisation of the communication channels within the network-on-chip, depending upon the architecture chosen
$\bar{w}$	The width of physical wires connecting each processing core within the network-on-chip
W_p	The number of physical wires connecting each processing core within the network-on-chip

To Uncle Ben...

Chapter 1

Introduction

"For nothing clears up a case as much as stating it to another person."

Silver Blaze, The Memoirs of Sherlock Holmes

Sir Arthur Conan Doyle

1.1 Motivation

The brain is one of the most complex and powerful systems in nature, and to unravel it is one of the great challenges of the 21st century [1]. The brain is at the centre of all functionality within the body, from control of limbs to sensory perception and higher level cognitive thinking.

A popular investigating technique into the brain's operation is to create and verify models replicating its function. Through the development of accurate models the brain's structure can be interpreted and its computation decoded. This furthers our understanding and allows for the development of novel treatments for neurological disorders.

Unfortunately, modelling the brain's operation is a non-trivial task. A typical human brain contains 10^{11} neurons with 10^{14} connections. This scale of computing far exceeds typical computing performance, and even the world's highest performing supercomputers struggle. Henry Markram suggests that "computational power needs to increase about 1-million-fold before we will be able to simulate the whole human brain." [2]

An effective computational platform for neural modelling must correctly replicate the brain's operation, which is thought to be encoded within:

- the generation within neurons of action potentials (AP), otherwise known as "spikes"
- and, the communication of these spikes between neurons.

However, neurons compute spikes using highly non-linear techniques and the connectivity between neurons is extraordinarily complex; resulting in signifcant computation, communication and memory overheads for an electronic platform.

Therefore, there is great interest in developing novel electronic architectures to enhance neural modelling capabilities. These projects scale from transistor-level designs targeting single neuron modelling [3] to alternative supercomputer architectures designed to mimic large-scale network operation [4].

Through expanding our knowledge of brain operation by the development of neural models many novel applications have been created, such as electronic neural prostheses that directly interface with the brain. These devices aim to repair, restore or replace lost neurological functionality, whether it be within sensory, motor or cognitive operation. They have had tremendous success in treating a range of conditions from loss of hearing [5] to Parkinson's [6].

The next-generation of neural prostheses will involve closed-loop systems, whereby the electronic platform will record, interpret and stimulate brain tissue in real-time. These systems will rely upon efficient and portable, but high-performing computational platforms, that can accurately model neural operation. These prosthetic devices could potentially replace impaired neural circuits, damaged from conditions such as stroke [7] or epilepsy [8].

However, electronic neural platforms have not typically been developed with the specific constraints imposed by a neural prosthetic. As such, the intention of this thesis is to extend the significant existing research into electronic neural modelling platforms towards systems capable of integration with neural prosthetics.

The project's original aims and objectives, outlined in November 2011, are defined below.

Aim

To investigate the design of efficient, high-performance electronic systems that can integrate with networks of biological neurons.

Objectives

Analysis of existing network structures to determine behavioural patterns.

Implementation of a network model in hardware to investigate performance and areas for optimization.

Derivation of a methodology for design of networks for interfacing with biological systems..

Application of methodology within a practical biological example.

1.2 Contributions

The key contributions of the thesis are outlined below:

- The highest performing specialised digital circuit to date to implement a Hodgkin-Huxley neuron model is described. Using the design over 100,000 neurons can be implemented upon a single modern FPGA.
- Similarly, a dedicated digital circuit for the Izhikevich neuron model is provided. This design is compared upon an FPGA with the Hodgkin-Huxley implementation and estimations are obtained for ASIC performance, which show increased performance over comparative options, including analogue neuromorphic design.
- A network-on-chip structure is shown to be the optimal design choice for implementing large-scale neural processing platforms. An empirical model is developed to evaluate the optimal design parameters of a network-on-chip depending upon the target application and the desired neural system.
- A benchmark cortical neural system is evaluated using the empirical model. The model highlights the importance of obtaining the correct network granularity in order to reduce area and power overheads by a factor of between 3x and 5x.

- Correct performance of the cortical neural system is verified using a software traffic model. This system highlights that communication challenges can be overcome and that memory overheads should be the primary concern within any implementation.
- Further, the empirical model is utilized in the design of a platform for a silicon cerebellar model containing 100,000 neurons. This platform can be used in high-performance accelerated mode, processing 1 second of real-time in 25ms, or in real-time for use in closed-loop cerebellar experiments and neural prosthesis.

1.3 Contributions to Literature

The work described within this thesis has contributed to the list of publications outlined below. Throughout the text where work has been published it is highlighted.

Journal Publications

- Junwen Luo*, **Graeme Coapes*** Terrence Mak, Tadashi Yamazaki, Chung Tin, Patrick Degenaar, "***Real-time Simulation of Passage-of-Time Encoding in Cerebellum Using a Scalable FPGA-based System***", *IEEE Trans. Biomed. Circuits Syst.*, Accepted with minor revisions, 2015

International Conferences

- Junwen Luo, **Graeme Coapes**, Patrick Degenaar, Terrence Mak, Tadashi Yamazaki, Chung Tin, "***A Real-time Silicon Cerebellum Spiking Neural Model based on FPGA***," *in Proc. International Symposium on Integrated Circuits*, Singapore, 2014
- Junwen Luo*, **Graeme Coapes***, Terrence Mak, Tadashi Yamazaki, Chung Tin, Patrick Degenaar, "***A Scalable FPGA-based Cerebellum for Passage-of-Time Represenation***," *in Proc. 36th Annu. Int. Conf. IEEE Engineering in Medicine and Biology Society*, Chicago, 2014
- Junwen Luo, Patrick Degenaar, **Graeme Coapes**, Alex Yakovlev, Terrence Mak, Peter Andras, "***Towards reliable hybrid bio-silicon integration***

*Both authors contributed equally to this work

using novel adaptive control system," *in Proc. 2013 IEEE International Symposium on Circuits and Systems*, Beijing, 2013

- **Graeme Coapes**, Terrence Mak, Jun Wen Luo, Alex Yakovlev, Chi-Sang Poon, "***A Scalable FPGA-based Design for Field-Programmable Large-Scale Ion Channel Simulations***," *in Proc. 22nd Int. Conf. on Field Programmable Logic and Applications*, Oslo, 2012

Local Conferences

- **Graeme Coapes**, Terrence Mak, Alex Yakovlev, Patrick Degenaar, ''***A Neural-Network-on-Chip for Next-Generation Neural Prosthesis***", Annual Research Conference, School of Electrical and Electronic Engineering, Newcastle University, 2014
- **Graeme Coapes**, Terrence Mak, Alex Yakovlev, Patrick Degenaar, ''***A Spiking Neural Network-on-Chip Platform for Hybrid Bio-Electronic Networks***", Annual Research Conference, School of Electrical and Electronic Engineering, Newcastle University, 2013
- **Graeme Coapes**, Terrence Mak, Alex Yakovlev, ''***Silicon Neural Node Capable of Large-Scale Simulations of Bio-Physically Realistic Ion Channels***", Royal Society's Young Researchers Meeting on Neural Engineering, 2012
- **Graeme Coapes**, Terrence Mak, Junwen Luo, Alex Yakovlev, ''***Trustworthiness in Hybrid Bio-Silicon Systems for Next-Generation Neural Prosthetics***", Workshop on Trustworthy Cyber-Physical Systems, 2012

1.4 Organization

The rest of thesis is structured as described below. The structure is visualized in Figure 1.1.

Chapter 2 - Neural Prosthesis

This chapter provides an introduction to neural prosthesis to provide justification for the development of a neural modelling processing platform for prosthetic applications. Initially a brief history of the progress of neural prosthetics is provided,

which details how the trend has moved towards closed-loop implementations. Next, a system-level specification is derived by evaluating the current technology, literature, and platforms available.

Chapter 3 - *Silicon Neuron Systems*

A thorough literature review is conducted to evaluate emerging and established neural model platforms. Comparisons are made between alternative options, including analogue VLSI (aVLSI), GPU and FPGAs. It is concluded that a dedicated digital design is the most appropriate for the application.

Chapter 4 - *Single Neuron Processing*

Digital techniques are investigated for the design of a neuron processing core, for both Izhikevich and Hodgkin-Huxley neuron models. The results of this section are used within the network-on-chip model outlined in the following chapters.

Chapter 5 - *Network Methodology*

This chapter compares the available communication options. A network-on-chip approach is deemed most suitable. Networks-on-chip design strategies are introduced and compared theoretically. An empirical model is described to allow for the network-on-chip design parameters to be analysed and optimized.

Chapter 6 - *Neural Network-on-Chip Case Study 1*

The previously described empirical model is utilized with a popular cortical benchmark to determine standard operating performance. Correct operation of the network is verified using a traffic software model.

Chapter 7 - *Neural Network-on-Chip Case Study 2*

The empirical model is utilized again to determine the optimum design of cerebellar model platform. This platform is fully implemented to allow for use in closed-loop experiments. A hypothetical neuroprosthetic example is demonstrated.

Chapter 8 - *Conclusions*

The contributions of the thesis are summarized along with directions for future work proposals.

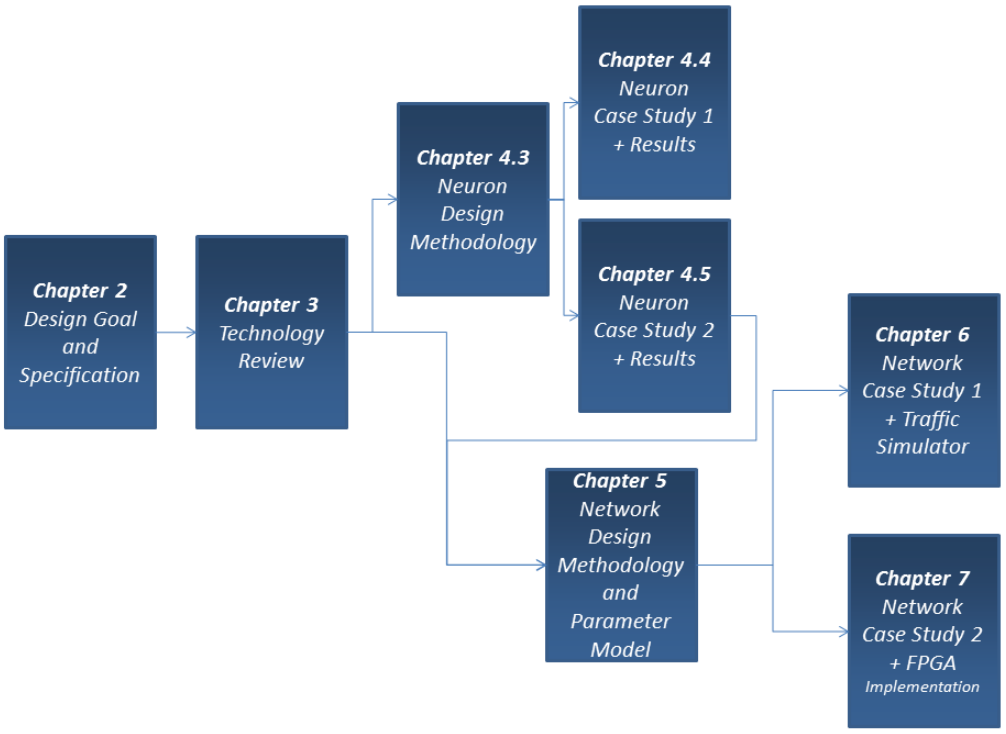


FIGURE 1.1: Structure of thesis. Chapter 2 provides background information on the history of neural prosthesis, and from this a design goal is developed. The available technologies are reviewed in Chapter 3. Chapter focuses on a sub-component of neural network-on-chips. The chapter initially develops a method for designing neurons before two example neurons are studied using the design method. Chapter 5 develops a methodology for studying the complete design of neural network-on-chips before this methodology is demonstrated using two specific biological neural networks in Chapter 6 and Chapter 7.

Chapter 2

Neural Prosthesis

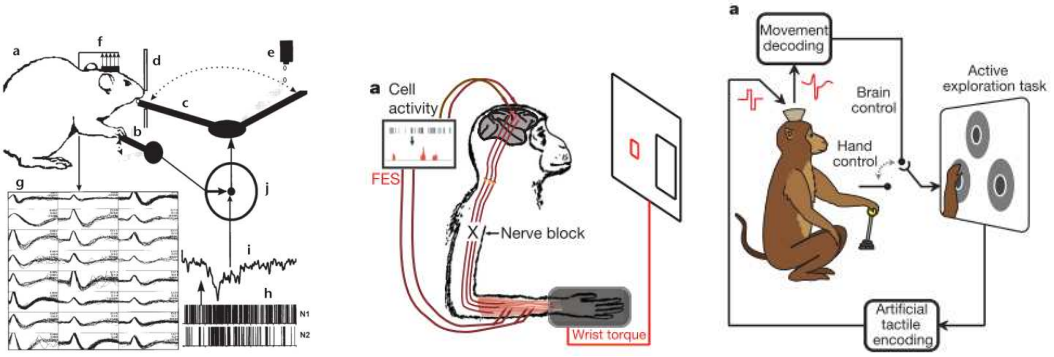
"There is nothing new under the sun. It has all been done before."

A Study in Scarlet, Sherlock Holmes

Sir Arthur Conan Doyle

Neural modelling has led to an increased understanding of the operation of the brain. However, as neural investigations grow in complexity more advanced electronic systems are required. This combination of an increased understanding of neural function and advancements in electronics has allowed for the development of neural prosthetics to treat previously incurable diseases.

This chapter provides an introduction to neural prosthetics, first of all by detailing their progress and highlighting likely areas of future growth. It is shown how neuroprosthesis are developing from open-loop unidirectional systems to more complex closed-loop systems which involve directly recording from and stimulating neural tissue in order to restore or repair lost functionality. The second part of the chapter studies the challenges associated with implementing electronic neuroprosthesis, from which a technical specification is developed.



(A) (B) (C)

FIGURE 2.1: Progress of brain-machine interfaces. (A) 1D robotic arm controlled by the neural activity of a rat [9]. (B) Direct control of arm muscles through a machine connected to the brain of a monkey [10]. (C) Volitional control of a computer cursor and a restored perception of touch [11]

2.1 Progress of Neural Prosthesis

2.1.1 Brain-Machine

There has been a long-term interest in the development of neural prosthesis to treat medical conditions incurable through drugs alone, such as tetraplegia. The research hypothesis is that a computer can be used to bypass the damaged nerves to allow the brain to directly interact with the outside world. This system is known as a brain-machine interface (BMI).

In 1969 E. Fetz at Washington University developed the first example of a simple cognitive brain-machine interface [9]. A single electrode was inserted into the motor cortex of a monkey and the rate of neural activity recorded. When the rate reached a certain activation level the monkey was fed banana flavoured pellets. The experiment showed that the test subjects were capable of learning to specifically control defined brain regions for augmented purposes. Although the study demonstrated a successful interface the limited technology available at that time prohibited more complex and useful implementations.

However, by 1999 multi-neuron recording techniques allowed for Chapin and Nicolelis et al. to create a machine capable of interpreting populations of neural signals in order to control a simple robotic arm [12]. Arrays of microwires were inserted into the motor cortex's of 6 rats who were trained to use a control lever in order to rotate the position of a 1D robotic arm in order to feed themselves water (Figure 2.1a). During the trials the neural activity was recorded and various processing models developed, such as artificial

neural networks, to produce an approximation of the robotic arm position. Interestingly, after multiple trials the attempted use of the control lever by the rats diminished suggesting the rats were aware of the volitional control.

A year later, Wessberg, Chapin and Nicolelis et al. extended this study to non-human primates, specifically owl monkeys. Again by studying a wide range of neural populations a processing model was developed to approximate the position of a robotic arm, this time in both 1 and 3 dimensions [13].

Velliste et al. [14] in 2008 developed the first fully interactive volitionally controlled robotic arm capable of movement in 3D space. During this experiment the arms of a primate were restrained and replaced with a synthetic model controlled purely through neural activity.

2.1.2 Brain-Machine-Body

Following this, Moritz et al. [10] have demonstrated a novel application involving the first brain-machine-body interface. Within this study the primates own limbs were controlled through electrical stimulation of the limb's nerves (Figure 2.1b). The investigators blocked the natural nervous system connections between the subject's brain and its arms in order to reproduce the conditions involved in patients suffering from tetraplegia. These connections were then replaced with artificial systems allowing for the control of the muscles to be regained by the animal. The study used neural data from only single cells, whereas the more recent study by Ethier et al. used data from large neural populations [15], perhaps allowing for multiple degrees of freedom to be regained in the limb.

2.1.3 Practicality

The above systems have all suffered from size and portability issues - limiting their practicality as research tools and their potential as neuroprosthesis. However, Borton et al. [16] have recently developed a portable and wireless interface to allow for recording of neural activity in moving primates. This will allow investigators to study neural data from normal primate activities, as opposed to constrained laboratory experiments, and also reduce the overheads involved in a long-term human neural prosthesis.

2.1.4 Human Results

Early work has started in translating the animal-based experiments described above to trials involving human patients. For example, Hochberg et al. created a successful BMI for a human patient suffering from tetraplegia [17][18]. A multi-electrode array with 100 probes was surgically inserted into the brain and the neural activity was used to control an on-screen cursor. This project was extended in 2012 to allow for the human patients to be able to control a complete robotic arm [19].

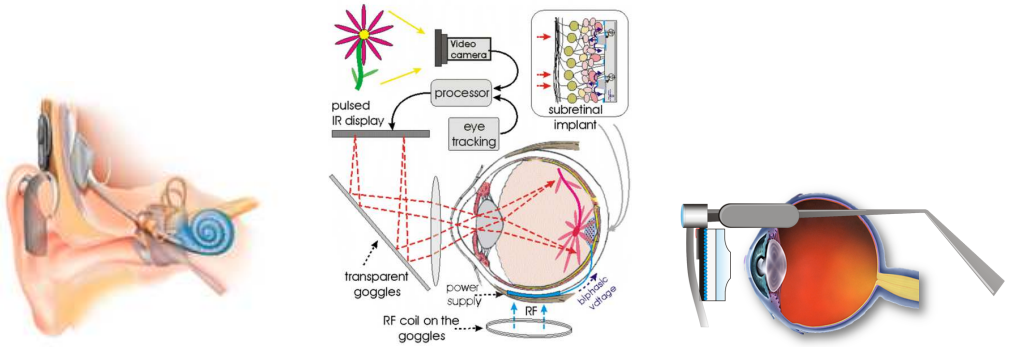
Similar work has been completed by Wang et al. [20], who used an electrocorticographic interface, and Collinger et al. [21] who created a volitionally controlled 7 degree of freedom prosthetic arm. All of these studies involve at most two human patients, suggesting that the research is still in the very early stages of development. Complications ranging from engineering challenges, to surgical and ethical overheads have limited the development and progression of the technology so far.

2.1.5 Machine-Brain

Electrical signals can also be injected into the brain and this significantly extends the range of neurological conditions that can be treated by prosthesis to include ailments such as chronic pain [22], Parkinsons [23], obsessive compulsive disorder [24], Tourette's [25] and a multitude of sensory disorders, including loss of hearing or sight.

For instance, in 1968 Brindley et al. [26] demonstrated a BMI capable of restoring a sense of sight to a blind patient. An array of electrodes was implanted into the visual cortex of a 52-year old woman and when an electrode was stimulated she experienced singular spots of white light in a fixed position depending upon the chosen electrode. The authors proposed that this demonstrated the potential of visual neural prosthesis in curing blindness.

The most successful neural prosthetic currently available is the cochlea implant. The development of cochlea implants began in the 1960s [27], and they were fully approved for medical use in the 1980s. By 2008 they had restored a sense of hearing to over 120,000 patients [5]. The system involves externally recording and processing sound waves before wireless transmission to surgically implanted electrodes (Figure 2.2a) These electrodes



(A) A standard cochlear prosthetic design [28]

(B) The retinal prosthetic design proposed by [29]

(C) Optogenetic retinal prosthetic

FIGURE 2.2: Designs of machine-brain interfaces for sensory neural prostheses. (A) A standard cochlea prosthetic design [28]. (B) The retinal prosthetic design proposed by [29]. (C) An optogenetic retinal prosthetic. External LEDs are used to stimulate genetically modified light-sensitive nerve cells.

are able to stimulate the surrounding nerve cells, which the patient is able to interpret as sound [5].

Recently, there has been great research interest in extending the work of Brindley et al. [26] and using the knowledge gained in the development of cochlea disorders to treat retinal diseases [29][30][31][32]. Commonly these devices have involved an external video camera being used to record an image, which is passed to a processor that interprets the signal and sends commands to an implant to stimulate the appropriate nerve cells. Typical visual prosthetic devices are shown in Figure 2.2b and Figure 2.2c.

The region of the brain that is stimulated to overcome blindness depends upon the cause of the condition. For instance, glaucoma causes damage to the optic nerve, so stimulation must occur beyond the optic nerve in the visual cortex [33]. Alternatively, retinitis pigmentosa is a degeneration of the light-sensitive cells within retina. As such, by stimulating the remaining cells within the retina, either subretinally or epiretinally a sense of sight can be restored [34].

Another successful neural prosthetic is the deep brain stimulation (DBS) system used in the treatment of a wide-range of movement disorders. In DBS, electrodes are implanted into a targeted brain region, such as the thalamus, through which electrical pulses are injected. The device acts as a pacemaker and regulates the activity of the brain region, which can be afflicted by the underlying neurological condition [6]. The amplitude and duration of the electrical pulses is typically user controlled and refined through experimentation.

2.1.6 Brain-Machine-Brain - Closing the Loop

Extending upon the experiments that simply recorded neural activity described above, in 2011 O'Doherty and Nicolelis et al. [11] showed the first brain-machine-brain interface that restored motor and sensory functionality. This was achieved by feeding various electrical signals back into the brain of the animal depending upon the animal's actions (Figure 2.1c). Over time the animal learned to associate certain frequencies of signals with a positive reward.

The majority of systems described above provide minimal external processing- most of the learning is encapsulated within the existing intact neural matter of the patient. However, intelligent closed loop systems, whereby a computer controls the interaction with the neural tissue depending upon the signals received, have been demonstrated and described recently.

In 2012, McLaughlin et al. [35] proposed extending a cochlea implant to include closed-loop control. Traditional open-loop systems require a time-consuming configuration session in order to effectively setup the implant. However, over time the patients' auditory system adapts to the implant, consequently, the configuration may need updating to ensure optimal performance. McLaughlin et al. suggest that by monitoring the auditory nerve response to the stimulation unit this configuration process can be automated and performed online.

Further, recently efforts have been made to improve upon the efficacy of DBS by introducing closed-loop platforms. For instance, Little et al. [36] highlight an experimental study whereby the local field potential (LFP) was recorded by the implanted electrodes and used in the calculation of the stimulating pulses. They found that the performance of the closed-loop DBS improved by over 25%. Also, the closed-loop system allowed for a reduction in stimulation time, reducing the overall energy consumption and doubling the lifetime of the battery.

Control of epileptic seizures in real-time using neural prosthesis has become the focus of many research groups [8][37]. These projects aim to monitor the neural circuits for onset of seizures and to adaptively stimulate the appropriate neural tissue to suppress the symptoms.

Similar closed-loop techniques have been proposed to treat cognitive disorders [38]. For example, Berger et al. [39] have postulated that by mimicking the signals involved in neural functionality of the hippocampus, an area susceptible to neural abnormalities such as stroke, cognitive abilities could be restored.

Although Berger's approach has had some success in mice experiments, it may suffer from the technique that has been adopted. The challenge has been treated as purely a digital signal processing problem, whereby the injected signals are generated from the received inputs using a multi-input multi-output (MIMO) model. This will require a separate model to be developed for each system, a new model developed each time the system grows and reduces the capability of the system to learn and adapt, which in silicon is usually accomplished through neural network implementations. Perhaps a more influential approach could be to generate artificial networks of neurons to replace the regions that have been damaged. The function and operation of these networks could be designed to coincide with the natural biological operation of the circuits they are replacing and they could allow for adaptation over time.

Even in many common signal processing algorithms neural networks are often chosen to complete the function. For instance, pattern recognition is often completed using artificial neural networks [40]. For pattern recognition learning is often completed offline.

The concept and an example of using replicas of neural networks for cognitive repair is discussed in [41], where it is proposed that small real-time silicon networks, initially running upon FPGAs, can integrate with biological tissue in vitro.

Recently devices have been demonstrated that influence the cognitive behaviour of mice [42][43][44]. Within this experiment a mouse was taught to fear a particular scenario. When this scenario occurred the mouse instinctively froze and became aware of danger. Liu et al. showed how by stimulating the same neurons in the hippocampus that were involved in the initial memory formation the mouse once again froze and became wary of danger. This proved how targeted stimulation of a select group of neurons is able to influence cognitive behaviour. Similar experiments were completed by Pais-Vieira et al. [45] who managed to transmit cognitive behaviour and response between two separate mice.

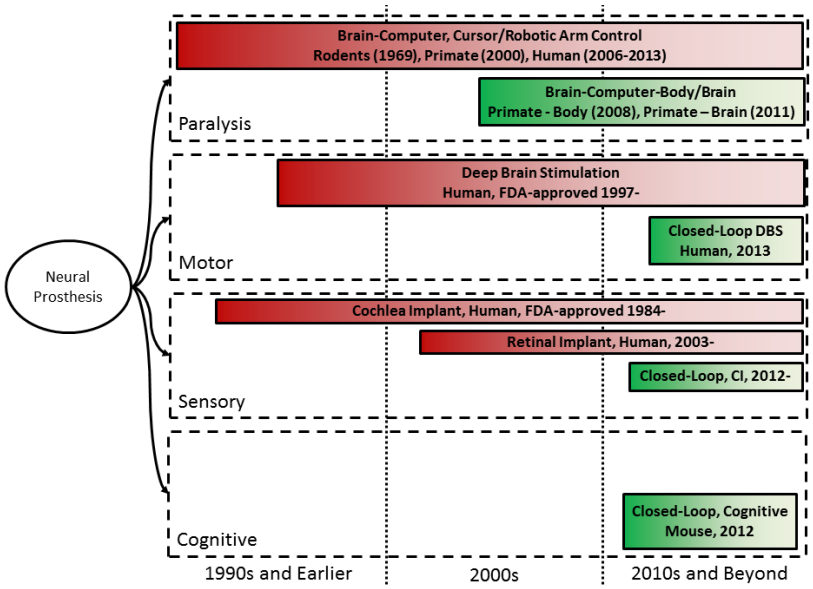


FIGURE 2.3: History of neural prosthetic development

2.1.7 Summary

The trend in neural prosthetic technology is moving towards closed-loop systems that both record and stimulate from neural tissue, as summarized by Figure 2.3. Closed-loop systems have been shown to increase the capability of the devices, whilst also improving their performance, such as the reduced energy consumption demonstrated by a closed-loop DBS device [36].

There is growing research interest in the development of cognitive prosthetics to treat neurological conditions. Berger et al. proposed a closed-loop MIMO processing platform to augment the damaged neural circuits to repair functionality. But, with the growing research into mimicking neural circuits in silicon perhaps the damaged circuits can be replaced directly. To ensure a successful prosthetic consideration must be made to reduce the size and energy consumption of any processing platform in order to make the system as portable as possible, reducing the impact upon any patient.

Further, accurate real-time electronic models of neural circuits will allow for prediction of a response to any neural stimulation, thereby reducing the risk and maximizing the effectiveness of many neural prosthetic devices.

Progress of Brain-Machine Interfaces: Key Points

- ***Brain-machine interfaces*** (BMI) involve computers directly interfacing with neural circuits.
- BMI can be used to treat previously incurable neurological conditions. These systems are known as ***neuroprosthesis***
- There is a growing trend towards ***closed-loop*** BMI, whereby the response from the machine is dependent upon the state of the neural circuits.
- Certain neurological conditions involve faulty neural circuits. Can these faulty circuits be ***replaced*** with machine equivalents?

2.2 A Systems Perspective

As for most electronic systems, closed-loop neuroprostheses require real-time acquisition of input signals, on-board processing and computation, and finally an output operation. For a neuroprosthesis to be an effective medical treatment these components need to be combined into a portable and integrated solution. A proposition for an integrated solution, perhaps using state-of-the-art 3D-IC technology is shown in Figure 2.4. Alternatively, either a traditional 2D-IC, such as that illustrated in [37], or even multiple ICs distributed across a PCB could be used depending upon the overall system objective and limitations.

Interface electrodes are placed close to the neural tissue to record local field potentials arising from surrounding neural cells. These electrodes require supporting analogue circuitry to amplify and filter the received signals before passing to an analog-to-digital converter. Once the signals are digitized they may undergo a spike sorting process whereby neural events are attributed to specific cells to aid in decoding the function of the neural network.

Once the biological network state has been decoded as much as possible, processing may be completed to translate the received signal into a meaningful action. As with most systems, this processing may require the use of on-board memory. In addition, an external communication channel would allow the state of the system to be interrogated; particularly important during early stage research and trials.

The result of the computation may be returned to the neural tissue through integrated microelectrodes, thereby *closing-the-loop*.

Finally, all system components will require energy supplied either from internal or external sources. For effective neural prosthesis this power source will be required to operate reliably and independently for a number of years.

In the following sections of the thesis, these individual components are described in more detail with a focus upon the latest published research.

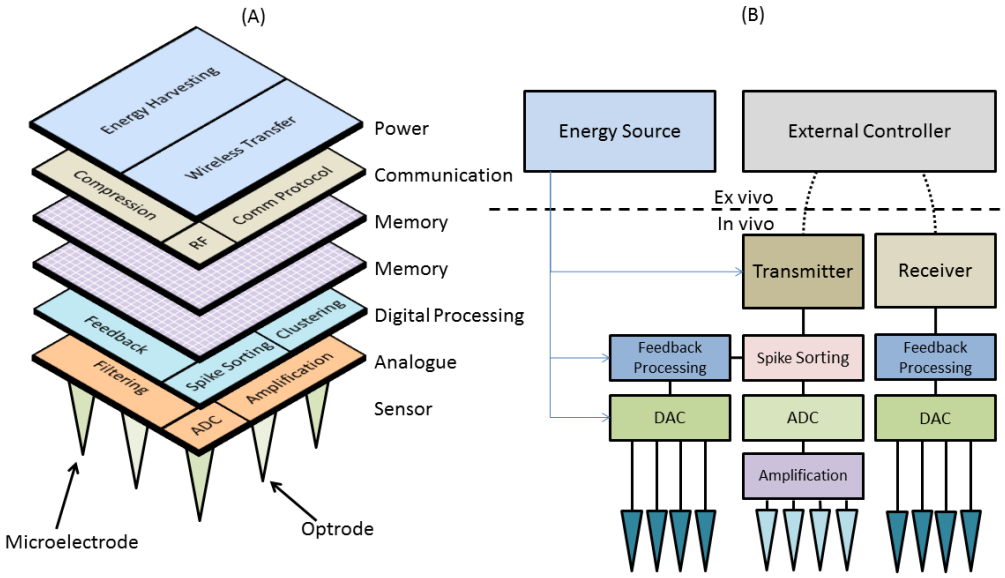


FIGURE 2.4: Example BMI system. (A) 3D Chip containing all the necessary components of the system. (B) Flow-chart illustrating how the components interact.

2.2.1 System Platform

For closed-loop neural prostheses electronics are needed to record, process and stimulate the brain. Recording and stimulating electronics have to be placed close to the neural circuits, but often, especially in clinical research environments, processing is performed on desktop/PC systems. For instance, Simeral et al. used commercial software running on general purpose computers for their study investigating a human controlling a computer cursor volitionally [46]. However, for prostheses the processing must be portable to allow for the patient to experience as normal a life as possible.

The processing could be performed externally, such as upon the ASIC[1] within the head-mounted unit in a cochlear prosthetic, as seen in Figure 2.2a. Alternatively, Al-Atabany et al. [47] investigate performing their required computation upon a generic mobile phone processing platform which offers low power benefits.

However, experience with cochlea implant patients has shown that external components are undesirable and there is a trend towards developing fully implantable systems [48][49]. Implantable systems will alleviate cosmetic, portability, reliability and security concerns whilst imposing strict constraints upon energy and area consumption [50].

[1] Application-specific integrated circuit.

A complete device may be fully implanted within the neural region that is being targeted. However, this introduces energy consumption issues, as discussed in section 2.2.2 and size limitations. Therefore, the electronic systems are typically either placed subcutaneously, as in [16], or within the chest cavity underneath the clavicle bone, as in all DBS systems. A typical DBS stimulator device occupies a volume of 22 cm^3 [51].

2.2.2 Energy Delivery

The capability of a successful neuroprosthetic device is determined by how much energy it has available and how efficiently it utilizes that energy. Non-invasive BMI systems, such as the EEG-controlled device illustrated in [52], can be powered directly through battery systems if desired. But, within invasive systems it is best to avoid transcutaneous wired connections to reduce the risk of infection and failure [53] and as such, the energy must be available locally. This allows for either energy harvesting, battery systems to be implanted alongside the actual device, or wireless power transfer techniques to be used.

There is significant energy available for harvesting in the human body in the forms of heat, chemical and kinetic. The challenge lies in the conversion of these sources into usable electrical energy, which is fairly easy in a regular environment, but to do so inside the body a device is severely constrained by size, heat dissipation and bio-compatibility [54].

Examples of devices have been demonstrated converting glucose into electrical energy and producing meaningful amounts of electricity, up to $280\mu W cm^{-2}$ [55]. A recent study demonstrated the first ever in-vivo example of such a bio-fuel cell generating meaningful power from a snail [56]. Also, in 2013 Zebda et al. [57] demonstrated a glucose biofuel harvester generating $38.7\mu W$ from a rat. However, these biofuel cells have a number of limitations, including a short-term life span measured in hours [58] and the size to power ratio of current implementations would not allow for inclusion into a neural implant capable of performing any reasonable operation. Similar issues of power density restrict implementations of thermal and piezoelectric harvesting techniques [59].

Traditionally, deep-brain stimulation devices have used non-rechargeable batteries to provide energy to the neurostimulator. Medtronic, a leading manufacturer of such devices, state that non-rechargeable batteries typically last between 3-5 years depending upon the rate and amplitude of stimulus that is applied [60] - typically in the range of 0-25.5mA [61]. Unfortunately replacing a DBS battery requires further surgery and therefore there is a desire to improve battery lifetime.

Medtronic have recently developed rechargeable variants of their DBS systems. Unfortunately the energy capacity of rechargeable variants is much smaller than non-rechargeable variants and therefore these devices typically require recharging every 10-30 days depending upon the stimulus program applied [62]. They are recharged via wireless inductive techniques. Also, rechargeable batteries still have a limited lifetime which is dependent upon the number of recharge cycles and as such they still require replacing, typically every 9 years [62].

Cochlea implants were the first commercial prosthetic device to have wireless charging [63]. Typically an external battery along with a transmitting RF coil is attached to the side of the head. Power at a rate of 20-40mW [5] is transferred to the implanted unit. The external batteries require recharging every 1-5 days [5]. Cochlea implants have the luxury of being a non-life-critical device, meaning they can be disabled to allow for the batteries to be swapped and no internal batteries are required. Perhaps for a motor or cognitive prosthetic this will not be acceptable. For instance, a patient suffering from a motor disability may struggle to swap batteries whilst the prosthetic is switched off.

Wireless power mechanisms have also been demonstrated for systems recording neural activity from cortical areas. Harrison et al. [53] were one of the first groups who demonstrated a device capable of recording and transmitting data from 100 electrodes. Their device used only 13.5mW. More recently Borton and Yin et al. highlighted the first portable neural activity monitor to be tested long-term within primates [16][64]. They implanted a 200mAh battery subcutaneously that allowed for their neural recording device to operate for up to 7 hours, consuming 30mA at 3V [64]. Ideally such devices should last for 24 hours between recharging - to allow for a human recipient of a neural prosthetic to only require recharging once per day. The interval between recharging can be extended by either reducing the energy requirement of the device or by increasing the charge capacity of the battery.

Interestingly, the amount of energy consumed by a neuroprosthetic is not just constrained by the amount of energy that can be delivered, but also the effect any heating of the device will have upon the surrounding tissue. An investigation described in [65] defined a rise of $1°C$ in the electronics as enough to damage surrounding brain tissue. Using a standard electrode array design they found the limit of power consumption within a 5x6mm^2 area is 35mW.

Brain temperature is known to rise to combat fever [66], but the $1°C$ and 35mW figures have become a common baseline for engineering specifications [37][67][68] in order to reduce the impact of any implant upon the neural tissue. However, it has been stated that progress with neural implants in clinical use may be delayed until further research into acceptable temperature implications is conducted and international safety standards are adopted [69].

To limit the impact of this constraint only the minimum amount of electronics required should be placed close to the neural tissue; other components should be placed between the skin and the skull, or as in a DBS unit within the chest cavity. Only systems associated with either recording or stimulating the neural tissue should be placed near the brain. The recording electrodes typically require approximately $20\mu W$/channel [70] for analogue front end processing, analogue-to-digital conversion and spike sorting. For stimulating electrodes the energy consumption is dependent upon the rate and amplitude of stimulation, as stated previously. Typically this has a value of between $100\mu W$ and 400 μW [71].

In a closed-loop neural prosthesis the online processing capability is likely to be a considerable consumer of energy. Unfortunately, most current implementations are not yet at a stage to consider power efficiency. For instance, the brain-machine-brain prosthetic illustrated by O'Doherty et al. [11] makes no mention of the processing platform, let alone its power efficiency. Also, Berger et al. who are developing closed-loop cognitive prostheses using a VLSI[2] implementation admitted in a paper in 2001 that they are yet to consider power efficiency of their design [72] and there has been no reference to power in any of their papers since [73][39][38][7]. Also, Bamford et al. [74] have introduced a VLSI design for closed-loop neural prosthesis and although power is discussed, no serious efforts have yet been considered to reduce it.

[2]Very-large-scale integration

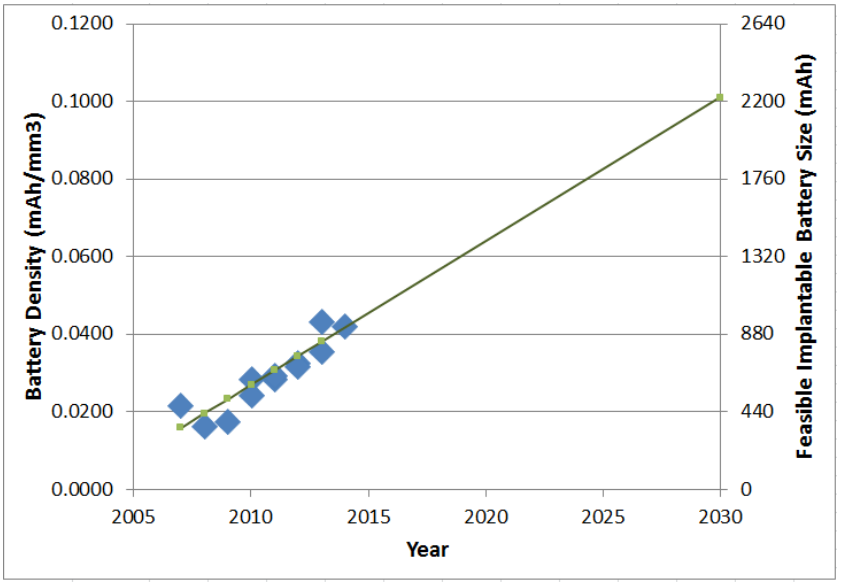


FIGURE 2.5: Recent progress in battery technology

As such, instead of comparing power efficiency of any proposed design with previous implementations, we need to consider the target application and develop a system-level estimated energy budget suitable to meet the long-term objectives.

To define this specification some assumptions need to be made:

- Due to the size of the commercial market we can assume that the state-of-the-art in battery technology is prevalent within smartphones. Current technology is approximately at $0.04 mAh/mm^3$ and is expected to grow to $0.10 mAh/mm^3$ by 2030 (see Figure 2.5).
- The implantable battery volume will remain constant at the same level as current DBS implantable batteries, at $22 cm^3$ [51].
- The electronics associated with stimulating and recording from the neural tissue is significant, but must be limited to 30mW, or 10mA, due to heating issues as the stimulation and recording must be located close to the neural tissue. The remaining energy available from battery source can be made available to the processing platform, which can be located away from the neural tissue.
- Only 90% of the battery's energy should be utilized in order to ensure safe operation.
- The processing platform is to be placed internally.

- The batteries should last a minimum of 24 hours between recharges.

Using these estimations currently the battery size is limited to 840mAh and is expected to rise to 2200mAh by 2030. As such, there is 21.5mA available to the processing platform currently, rising to 74mA by 2030. Due to this limitation in available energy, which Furber and Temple predict will be become the true cost of computing as opposed to processing capability [75], a system-level design perspective needs to be considered and to ensure that the scalability of a proposed design remains within the scalability of the energy supply.

2.2.3 Computation

Within closed-loop systems computation is required to interpret the input received from the neural systems and to calculate a response. As noted previously, most current BMI systems utilize desktop/PC systems upon which implement mathematically heavy algorithms, such as Kalman Filters [76]. Recently however, Dethier et al. [50] used a spiking neural network in order to decode signals for a BMI, claiming that a neuromorphic implementation would be able to save multiple orders of magnitude in energy.

Further, cognitive prosthetics must compute a response based upon the decoded input. This response must mimic the correct response of the damaged neural circuits in order to replace or restore the lost functionality. For instance, if a neurological disability is caused by a faulty pacemaker circuit, we may wish to utilize a model of the correct pacemaker circuit in the derivation of the feedback to be injected to the brain.

There have been many spiking neural network models developed that recreate neural circuits. For instance, the model by Yamazaki and Tanaka [77] replicates the operation of the cerebellum, Traub et al. [78] have developed a thalamocortical model and Kopell et al. [79] a hippocampus model. Many more examples are available at the ModelDB website[3].

Interestingly, when creating artificially intelligent or learning-based systems designers typically look to implement neural network models [80][81][82]. Therefore, in developing cognitive prosthetics that may involve learning we may naturally consider neural

[3] https://senselab.med.yale.edu/ModelDB/

networks. Implementation of a biological network model could automatically include any learning which is required. In fact, Bamford et al. [74] and the ReNaChip project have demonstrated a biomimetic/neuromorphic chip for closed-loop prosthesis including learning of a new motor response.

To ensure complicity with the interacting neural circuits any implementation of the model should run in the same time-domain.

Although, the brain may consist of up to 100 billion individual neurons a closed-loop prosthetic will interact with only small defined regions. As such, a neural network model implemented for a prosthesis application is not required to scale all the way to a full brain. Instead the emphasis should be upon the efficient implementation of a defined network size. This is counter to the aims and objectives of many large neural platform projects, such as BlueBrain [2] and SpiNNaker [4]. A system containing between 10,000 and 100,000 neurons can be expected to produce meaningful results as is explored later in section 6.1 and section 7.1.

2.2.4 Neural Interface

Neural activity is recorded by inserting an electrode close to the desired neurons. When these neurons produce action potentials they change the local field potential, which is detected by the electrode.

A singular electrode is able to discriminate between multiple local neurons through a process known as spike sorting [83]. Each neuron has a distinctive pattern that is produced for each action potential. The spike sorting algorithms are able to distinguish between different patterns to allocate events to particular neurons. Spike sorting is a very active area of research, with most studies focusing upon improving the accuracy whilst reducing the power consumption [84][85][86].

Multiple electrode arrays (MEA) allow for observing a greater number of neurons. Recent neural prostheses [16][53] have tended to use the 100-electrode Utah array which is commercially available. This allows for between 300-400 individual neurons to be observed. Ideally this value will be much larger to allow for observation of the distributed processing of a biological neural network. For instance, Berger states in [7] that they can observe from only 16 different neural sites, in a region containing up to 450,000 neurons.

Recent publications have illustrated much larger arrays. However, the amount of data that these arrays generate introduce a great processing and energy burden. The 11011-electrode array illustrated by Frey et al [87] can in fact only record from 126 sites simultaneously. This may offer greater flexibility in choice of location but it does not yet produce much more information than the common Utah MEA.

The machine-brain interface may also utilize electrical signals. For instance, both the cochlear implant and DBS systems involve neurostimulators injecting current into defined brain regions. However, this process is limited in that specific neurons are unable to be targeted and neurons can only be excited.

A new approach known as optogenetics [88] offers great potential. This optical stimulation technique relies upon the genetic adaptation of a neuron's structure to include a light sensitive ion channel, known as channelrdhodspin-2 (ChR2). When a bright light source is targeted at a neuron expressing this ion channel the neuron may produce an action potential. This allows for a population of specific neurons to be targeted, allows for inhibition as well as excitation, and vastly improves the resolution offered. This approach has been used previously by Liu et al. [42] within a cognitive brain-machine interface.

High density optoelectrode arrays are being developed [89] for neural prosthesis, but mainly for external issue, such as within retinal prosthetics [47]. When implanting devices, the optoelectrode arrays are limited in size by their energy consumption and temperature dissipation. Therefore, typically we may expect a closed-loop prosthesis to be limited to only 10-100 electrical/optical input and output channels.

2.2.5 Data Transfer

To monitor the neural activity and the health of the implanted device a communication interface is required. Wireless techniques are often preferred to avoid transcutaenous connections.

Harrison et al. [53] combined the wireless data transmission with the power protocol and achieved a data rate of 330kb/s. Alternatively, Borton et al. [16] used a dedicated RF channel and obtained an increased data rate at a greater cost in power. They utilized their increased channel bandwidth to transmit the full resolution of the recorded signal

from all 100 electrodes. With each electrode sampled at 12 bits at a rate of 20,000 samples / second this equates to a bandwidth of 24Mb/s. To reduce energy consumption in the data transfer protocol spike sorting can be used to decrease the required data rate. For instance, Karkare et al. [70] achieved a data compression of 240x at a cost of only $5uW$/channel.

The utilisation of this high bandwidth could perhaps be put to more effective use by reducing the ADC sample resolution from 12 bits as this infers an exceptional noise floor. Rizk et al. [90] suggest that only 8-bits are needed if an adaptive scheme is introduced to match the dynamic range of the input signal; this may provide up to a 33% increase in performance over Borton et al. [16].

To transmit the status of a complete neural network model of between 10,000 and 100,000 neurons a data rate of between 4.9Mb/s and 43Mb/s is required. This is within the data rate achievable by current technology. However, in order to reduce energy consumption transmitting the status of the neural network should only be required during early initialisation and testing stages.

2.2.6 Reliability

Stable operation over a number of years is required for any implanted prosthetic device. For instance, pacemakers are expected to last for 5-10 years [91]. Studies have previously demonstrated how electrode arrays are able to function for extended periods of time. Within [92] no discernible degradation, either qualitative or quantitative, was found in the signals received from an electrode array implanted within the motor cortex of macaque monkeys for up to 1.5 years. Also, [46] validated the reliable performance of a 100 probe electrode array after 1000 days implanted within a tetraplegia patient. The patient was able to successfully use the prosthetic device with a 91% accuracy 1000 days after surgery.

Although both these studies showed that sustained operation is possible they both highlighted how reliability could be affected in the long term. Over extended periods of time recordings may become unstable due to cell death surrounding the implant [93] or actual physical displacement of the implant [83]. To overcome this BMI algorithms must be

flexible so as to be able to adapt to changes in signal characteristics and requirements over time.

2.2.7 Summary

The review provided by this section has developed a system-level specification for a closed-loop cognitive neuroprosthesis, as described in Table 2.1. The specification considers the key and unique constraints detailed in this section whilst requesting a minimum level of performance for an effective medical device.

2.3 Chapter Summary

Neuroprosthetics provide the opportunity to treat a wide-range of conditions, many of which have previously been thought of as incurable or untreatable. To increase the functionality, reduce the risk and improve the efficiency of neuroprosthetics there is trend towards developing closed-loop systems involving recording, interpreting and stimulating in real-time. These closed-loop systems will come to rely upon accurate, effective and portable electronic neural modelling platforms.

TABLE 2.1: System specification for a closed-loop cognitive prosthetic.

Component	Specification
Size	$30mm^3$ (brain), $2500mm^3$ (chest)
Power	10-100mW, Battery, Rechargeable every 24h
Processing	Spiking Neural Network, 10-100k neurons
Time	Real biological time, 10-100 events/second/neuron
Interface	10-100 electrodes/optrodes
Data Rate	5-50Mb/s
Platform	Implanted and Portable
Lifetime	10 years

Chapter 3

Silicon Neuron Modelling

"Eliminate all other factors, and the one which remains must be the truth."

The Sign of Four, Sherlock Holmes

Sir Arthur Conan Doyle

Modelling of neural systems in silicon is a growing area of research. This is primarily motivated by two features:

- the desire of neuroscientists to simulate brain functionality to increase their understanding.
- the requirement of engineers to design high-performance electronic systems.

The computational performance of a brain far exceeds that of even the most advanced computer systems in many disciplines [94], most impressively in the area of power efficiency, a popular topic in embedded systems research. For instance, simulation of a full human brain with current technologies would require hundreds of MWs of power [95], as opposed to the brains consumption of 20-30W [96]. As such, it is hoped that large-scale study of brain function could provide insights into how to reduce power consumption and improve other features including robustness and artificial intelligence.

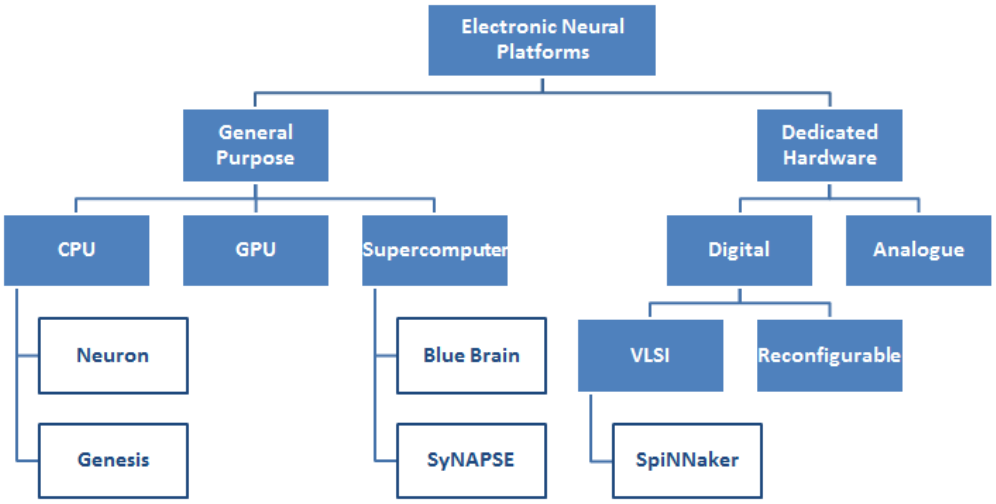


FIGURE 3.1: Different platform options for electronic neural modelling. Some well known projects are provided as examples.

3.1 Platform Options

Due to the benefits provided by neural modelling, many research groups have focused upon developing efficient implementation platforms. However, there is a wide choice of platform options depending upon the focus and requirements of the application. These choices range from software on a standard desktop PC to complete dedicated hardware design at a transistor-level. The range and relationship of these options are illustrated in Figure 3.1. In this section each of these options are evaluated in terms of their suitability with regards to the specification outlined in Table 2.1.

3.1.1 General Purpose Platforms

By far the most popular implementation route for neural modelling is to use general purpose platforms. This is because of the general availability of machines such as personal computers (PC)s and graphics processing units (GPU)s and their relative ease in programmability. However, as will be described, the performance of these options is not always satisfactory, particularly for a neural prosthetic.

PC Traditionally, there has been great emphasis on software-based approaches to neural modelling, such as the NEURON [97] and GENESIS [98] packages. Both of these options offer powerful capabilities, with a wide range of freely available libraries, which

include highly accurate bio-physical neuron models and network topologies. The ease of programming and their flexibility also encourage users, particularly from the neuroscience community.

However, the software-based approach is inherently limited by the computational bottlenecks of the underlying platform, which in most cases is a standard desktop PC. The sequential Von-Neumann architecture of the desktop PCs limits the scalability of the neural models and execution time increases exponentially with increasing neural model size. In fact, the software model described in Chapter 7 no longer executes beyond a certain network size due to the excessive memory overhead filling the available resources (cache, RAM, virtual memory).

When considering a closed-loop platform, software approaches also suffer from timing issues caused by their non-deterministic operation [99]. Ideally feedback to the neural circuits will occur at a guaranteed time interval. But, software approaches share hardware resources with other processes, introducing the opportunity for delays and variation of the time interval between updates. Real-time operating systems are available to overcome this burden [100] but they introduce additional complexity, negating the main benefit of a software approach.

GPU Software-based designs can be extended to run upon GPUs. GPU processors operating with hundreds of processing cores have been shown to complete spiking neural network simulations 20-100x quicker than the equivalent centrail processing unit (CPU) implementation [101][102][103] and they scale linearly with increasing number of neurons within the network. The computing of neurons can be efficiently shared between processing elements within a GPU, but they are inherently limited by memory and communication bandwidth constraints, which are the main tasks in a spiking neural network. Although a GPU offers an extremely high raw memory bandwidth this is difficult to achieve in practice and requires adhering to strict memory access patterns [101].

Similarly to software approaches, GPU designs will suffer from timing-related issues. Also, the only way to scale a GPU is to include more processors within the core, which still doesn't overcome the memory and communication bottlenecks. GPUs are undoubtedly high-performance components, but they are commercially driven with a particular

application focus, as such, their architecture is unlikely to be altered to become more suitable for the non-profitable market of neural modelling.

Supercomputer The Blue Brain [2] and SyNAPSE [104] projects are both examples of systems looking to exploit the computing resources available to some of the world's quickest supercomputers, specifically the IBM Blue Gene computer, which has hundreds of thousands of processors and hundreds of terabytes of available memory.

The SyNAPSE project has previously demonstrated simulation of over 1 billion neurons with over a trillion synapses running on Blue Gene [104], although much slower than biological real-time. They expect to be able to simulate the whole human brain by 2018 [105].

The investigators behind the Blue Brain project, led by Henry Markram, believe that a greater level of detail is required than the simple leaky integrate and fire model used by others. Once again they aim to simulate the whole human brain, but aim to include the complex morphology of the complete neuron structure [2]. Hence, they still believe a considerable jump in computing resources is required for full brain simulation.

The performance potential of supercomputers is undoubted. However, they come at a great cost, in terms of size, energy, financial and complexity. It could also be argued that the architecture itself is not the most efficient for neural modelling due to limitations in the communication protocol. For example, the one-second simulation of the thalamocortical system by Izhikevich and Edelman[106], which involved 1 million multi-compartmental neurons and 500 million synapses, required 11 minutes to run. And, the SyNAPSE project reported their simulation in time in relation to the average firing frequency (643x slower than real biological time per spike event) [104]. It is likely that by using the supercomputer approach this time will grow exponentially due to bottlenecks in the communication architecture.

3.1.2 Dedicated Hardware

General purpose platforms are not designed for use with neural applications. Therefore, they are not always the suitable for the application. Recently, there has been significant interest in developing specific electronic platforms to cater for the demands of neural

systems. These platforms include both dedicated analogue designs, where the individual specifications of transistors are manipulated, and digital designs, where standard digital design techniques are used to design an efficient communication and computation infrastructure.

Although dedicated hardware may provide increased levels of performance, the complexity of design has some negative affects, including:

- increased cost in materials and development time.
- inflexibility in operation.
- requires specialist knowledge and experience.

Within the following paragraphs the potential performance improvements of different dedicated hardware approaches are described and the impact of the negative affects listed above are considered.

SpiNNaker Due to the bottlenecks of general purpose supercomputers, there is great interest behind the potential of an alternative system, known as SpiNNaker [4]. Ultimately, this system may resemble a supercomputer in its appearance, but it has been developed from the ground up with the objective of neural modelling in mind. As such, the network architecture is optimized for the transmission of neural packets, the memory system is honed for efficient use, and the central ARM processors offer reduced energy consumption [107]. The SpiNNaker system will be able to transmit packets across its 1 million nodes in under 1ms, taking advantage of the silicon/biology time inequality.

An overview of the SpiNNaker platform is illustrated in Figure 3.2. Each SpiNNaker processor contains 18 ARM cores, each capable of modelling between 100-1000 neurons, depending upon the number of synaptic connections and the neural model used [4][108]. They estimate that each processor requires between 1W [4] and 2W [109]. To model a single Izhikevich neuron, without regard for synaptic connectivity, requires 100uW [109] and approximately $1000\mu m^2$.

Primarily, the SpiNNaker objective is to run in real-time with reduced energy consumption. Using the analysis provided by [109] we can estimate that a simulation of

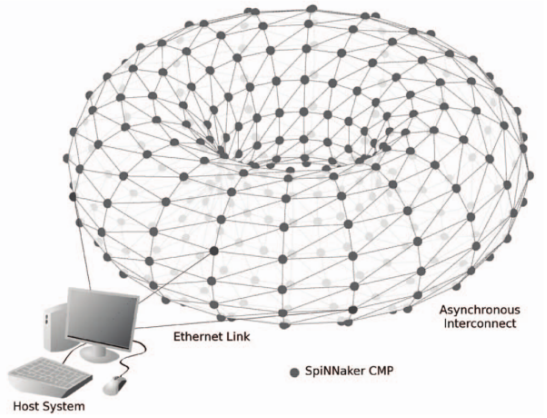


FIGURE 3.2: An overview of the SpiNNaker machine [4].

1 biological second of 10 billion neurons firing at 2Hz will require 3.72MJ, as opposed to the SyNAPSE project running upon a commercial supercomputer requiring 8.4GJ (whilst running 1000x slower than real-time) .

Although the SpiNNaker project is an improvement upon software-, GPU-, and supercomputer-based approaches a single chip is unable to meet the required specification defined in Table 2.1 in terms of network size or power performance.

VLSI Analogue design approaches are typically used when designers are interested in developing neural-like computation with reduced power consumption and low area overheads. Frequently designers choose to exploit the similarities between transistor and neuron characteristics, something first proposed by Carver Mead [110] and demonstrated by Mahowald and Douglas [111] in 1991.

A more recent, and perhaps more elegant, design was offered by Farquhar et al. in 2005 [3]. This design exploited the sub-threshold properties of CMOS[1] transistors to recreate Hodgkin-Huxley neurons using only 6 transistors. This design is illustrated in Figure 3.3. Unfortunately, they neglect to give any detailed design results in terms of area or power performance of their design.

Similar approaches have been illustrated by multiple groups, including Wijekoon and Dudek [112], Vogelstein et al [113], Hynda and Boahen [114] and Indiveri et al. [115]. A complete review and a description of the differing technologies is presented in [116].

[1]Complementary metal-oxide semiconductor

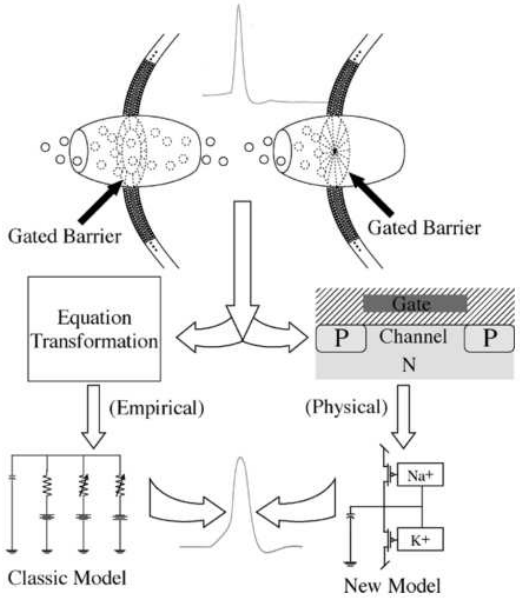


FIGURE 3.3: Neuromorphic silicon neuron [3].

Unfortunately, despite the perceived benefits of utilizing *neuromorphic* technology the realities of such systems are not as generous.

Firstly, a silicon neuron may only require 9pJ of energy per spike [112]. But, this neglects the communication of spikes between different neurons. This communication is impractical in analogue techniques so a mixed-signal approach is required. Spikes produced by the analogue neuron are packeted and transmitted over a digital communication infrastructure, commonly using the address-event representation protocol [113]. Vogelstein et al [113] estimate that each synaptic event equates to 0.6nJ, a number which is likely to rise with increasing network dimensions (43nJ in the SpiNNaker system analysed in [109]).

Secondly, the design of an individual neuron requires only 10s of transistors, but, when the supporting circuitry is included the area efficiency is not as beneficial. The most complete silicon neuron circuit demonstrated to date [112] requires approximately $2800\mu m^2$ per neuron, as opposed to the estimated SpiNNaker neuron size of $1000\mu m^2$.

The area of analogue neurons is unlikely to reduce in size with decreasing technology fabrication node sizes due to transistor mismatch and variation. To overcome these non-deterministic parasitics transistor sizes are much larger than their digital equivalents. Some neuromorphic designers claim that this variation can be utilized to represent noise and errors within the biological circuits [113][116] but, this is yet to be demonstrated

in practice. Neuroscience modellers would also request an element of control over the noise and variation within their spiking neural network simulations which would be unachievable using analogue methods.

Analogue neuron circuits require extensive experience to design and once manufactured provide very little reconfigurability. If reconfigurability is included it will introduce area and energy overheads, further negating the supposed benefits. Lack of reconfigurability will limit the potential pool of neural systems that can be mimicked.

Further, neuromorphic systems are limited to the time-domain that is targeted at design-time. For instance, the silicon neuron proposed by Wijekoon and Dudek [112] operates 1000 times quicker than biology. Similarly, the FACETS project [117][118] have developed a mixed-signal approach designed to also operate at 1000x. By operating at faster than biological time they are able to reduce the capacitance sizes in their designs, significantly improving the silicon area utilisation [117].

Alternatively, digital neurons have also been demonstrated [119][120][121], although perhaps due to cost issues most are only implemented upon FPGAs [122][123][124][125]. Emery et al. described an optimized leaky integrate and fire design that uses only $700\mu m^2$ per neuron in 90nm, comparable to analogue design sizes. Whereas Seo et al [120] used $2500\mu m^2$ in 45nm. Imam et al. [121] developed an Izhikevich digital model in 65nm which consumed $30000\mu m^2$.

Digital neurons have a number of advantages over analogue versions, including:

- Digital neurons are highly flexible and configurable. Parameters to the neuron can be efficiently stored in memory, which can be accessed and updated by a user. Analogue neurons are either fixed at build-time or require complex configuration circuitry [94]
- They can be operated in multiple time-domains. A single implementation could be operated in accelerated mode or in real biological time [121].
- They show deterministic behaviour. The state of a digital neuron is predictable and unlikely to be affected by anomalous variations within the physical circuit [126].

- Digital design is much quicker, easier and requires less experience in specific design and manufacture processes [94].

Reconfigurable Digital Due to the rapid developments in reconfigurable digital circuit technology, such as field-programmable gate arrays (FPGA), and the fundamental parallelism and flexibility they offer, reconfigurable circuits can provide the performance to satisfy the increasing requirements of many neural models. Large networks of Izhikevich neural models have been illustrated by Thomas and Luk [123], Moore and Fox [125] and Cassidy et al. [127] amongst others. Whereas, Mak et al. [124] and Weinstein et al. [128] chose to implement small networks of biophysically accurate models for hybrid bio-electronic applications.

The size and complexity of FPGA-based neural models have been limited by available memory upon the FPGA [129][123]. Moore and Fox [125] utilize large off-chip memories, but these require the design of dedicated architectures so that the available memory bandwidth is fairly and efficiently allocated between processing nodes. Cassidy et al. [127] also utilize external SRAM.

FPGAs offer a rapid prototyping environment, but their inherent reconfigurability limits their performance in terms of area and energy. Most parties agree that without the issue of financial cost a digital VLSI platform would be preferred to reduce both of these factors [130][127].

Due to the agreed high energy consumption of FPGAs this factor is rarely reported in FPGA-based neural model papers.

3.2 Design Considerations

In this section, some key design principles are extracted from the options highlighted above and the impact of those principles are evaluated. All of the systems above are capable of the fundamental processing of individual neurons, but when the scale and complexity of the brain is considered memory and communication bottlenecks begin to become a problem.

3.2.1 Memory Requirements

Memory limitations are prevalent in all of the design options listed in section 3.1.1. A typical simplified neuron, such as an Izhikevich neuron, requires 7 parameters, each at a minimum 16-bit resolution [108] [2] in order to maintain system stability and accuracy. This equates to 11.2Mb for 100,000 neurons. Assuming that each neuron is connected to on average 1000 other neurons, each with a 32-bit identifier then a further 3.2Gb is required to store the synaptic connectivity. Further, assuming that a mean neuron firing rate is 10Hz and that each neuron model is updated every 1ms, using a simple estimation the system will require a memory bandwidth of 43Gb/s. These memory requirements impose significant strain on both area and energy resources.

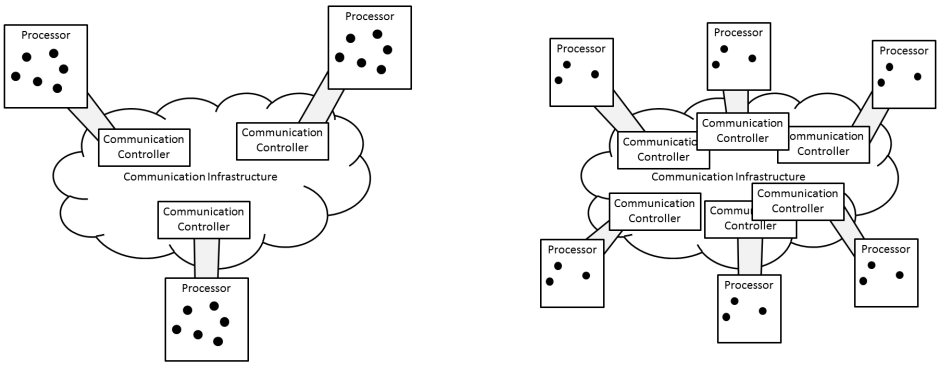
3.2.2 Communication Requirements

Similar estimations can be made for the communication requirements. With 100,000 neurons firing at 10Hz, there will be 1M events per second and 1000M synaptic updates required.

Most systems simplify the transmission of spike along an axon to a binary event. Nearly all of the projects listed above use a variation of a scheme known as address-event representation (AER). This allows for multiple virtual axons to be multiplexed across a single silicon connection, taking advantage of the biology/silicon time inequality.

Within AER, when a neuron produces a spike, a packet is generated containing the neuron's identifier. This packet is transmitted to all required post-synaptic neurons using a variety of interconnection protocols. For instance, SpiNNaker [4] connects multiple processing cores in a torus topology and uses a multicasting packet-switching methodology to route packets to the correct location. The EMBRACE [131] project uses a hierarchical arrangement of routers to take advantage of clustering of communication. Emery et al. [119] utilize a hybrid approach, whereby a subset of connections are transmitted in a unicast fashion over a mesh topology; the remaining connections are directly connected in configurable blocks. Alternatively, the Neurocore chip [120] disregards a packet-based approach and uses a crossbar matrix and a circuit switching methodology.

[2] 16-bit resolution was chosen as it was shown to provide a similar level of accuracy and stability as a floating-point implementation. Reducing the resolution further may impact upon the result of the calculations significantly.



(A) 3 processors each implementing 6 neurons

(B) 6 processors each implementing 3 neurons

FIGURE 3.4: Illustration of varying granularity of a neural network platform. Each variation computes 18 neurons.

Further information about interconnection protocols is provided in section 5.2.1.

3.2.3 Granularity

The communication of events between neurons requires multiple communication controllers. Each communication controller is responsible for packetizing information from a subset of neurons and transferring this information to other controllers throughout the system.

Across all the projects listed previously the size of this subset of neurons allocated to each controller varies significantly. For instance, within EMBRACE [132] a single controller is used for 10 neurons. Whereas, for the Bluehive [130] project it is 16000 neurons.

This parameter can be defined as the granularity of a system, and is illustrated within Figure 3.4. If a neural network has s neurons implemented upon m processing cores then each processing core has a granularity of $s/m = n$.

Vainbrand and Ginosar [133] studied silicon implementation of neural networks from a theoretical perspective, but left a study of granularity as *future research*. The issue of granularity has only previously been analyzed by Cassidy et al. [127]. They developed an analytical model to determine the optimal granularity of a neuron processor for two different criteria, delay and area. They found that to optimize both speed and area each processor should contain approximately 8000 neurons. Unfortunately, within their model

TABLE 3.1: Summary of silicon neural model design approaches. For cost, size and power, numerical rankings are provided, with 1 being the best and 7 being the worst.

Option	Cost	Size	Power	Real-time	Neuron Number	Implantable
PC	1	5	5	Limited	10^3	If using microprocessor [3]
GPU	2	4	4	Limited	10^6	If using mobile GPU [4]
Supercomputer	7	7	7	No	10^9	No
SpiNNaker	6	6	6	Yes	10^9	No
aVLSI	5	1	1	Yes	10^3	Yes
dVLSI	4	2	2	Yes	10^6	Yes
FPGA	3	3	3	Yes	10^6	Yes

they haven't included consideration for the energy consumption, the communication overheads, and the synaptic connections.

It is hypothesized that all three of these factors will have a more significant impact upon the optimal design of a neural network platform. The issue of granularity is considered at each stage of the design process within this thesis. For instance, Figure 4.23 shows how the area and power change for a neuron processing core depends upon the granularity and similarly Figure 6.11c shows the area and power variation for a complete neural-NoC for a typical cortical column.

3.3 Summary

The different design options are outlined in Table 3.1. Software and GPU-based approaches are flexible and easy to implement but are not powerful enough to perform the required operations in real-time. Supercomputer based approaches are obviously not suited for closed-loop systems due to their size, energy inefficiency and timing constraints. Single chip VLSI and FPGA options can meet the performance requirements, but they are costly and time-consuming to implement.

Analogue VLSI neuromorphic implementations are popular, but still require significant digital circuitry to operate. FPGA options are flexible and cheap and easy to implement, but they are not as energy efficient as dVLSI and do not offer the same level of performance. As such, a dVLSI approach should be preferred for closed-loop neural network prosthesis. Although this area has been studied in detail previously there has

[3] Limits performance capability.
[4] See previous footnote.

been very little research focusing upon single-chip implementations which are suitable for long-term implantation for a neural prosthesis.

Chapter 4

Single Neuron Processing

"It has long been an axiom of mine that the little things are infinitely the most important."

A Case of Identity, Sherlock Holmes

Sir Arthur Conan Doyle

Neurons form the fundamental building block of the brain and they are often considered equivalent to a logic gate within a digital circuit [75] as they receive inputs from multiple sources and produce a single output. However, unlike logic gates the function to translate the input to the output is often highly complex, not a simple binary operation.

A generic neural system is outlined within Figure 4.1. A neuron may receive multiple inputs from other neurons within the network. These inputs are summated over time, typically with a non-linear function and a time-decaying parameter. When the summated value crosses a threshold the neuron will produce an output signal of its own. This signal will then be propagated to all other connected neurons within the network.

This network of neurons is capable of computing a certain neural function, in the same way that a series of logic gates may be connected to calculate addition or multiplication. For instance, within a pattern recongition function, the connections and individual computation of neurons will be tuned such that depending upon a certain criteria of inputs a set output pattern is produced.

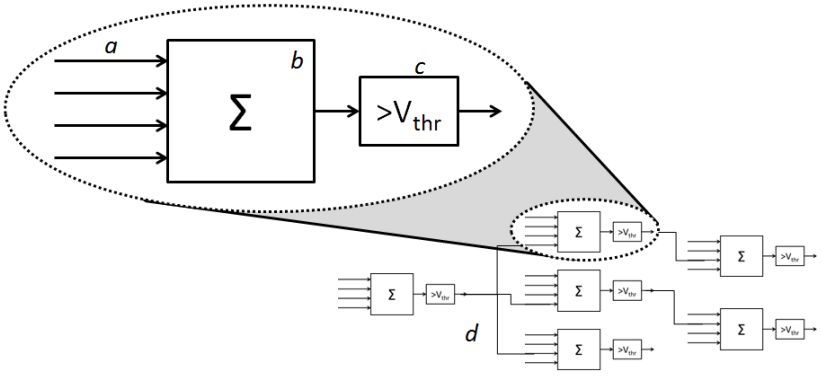


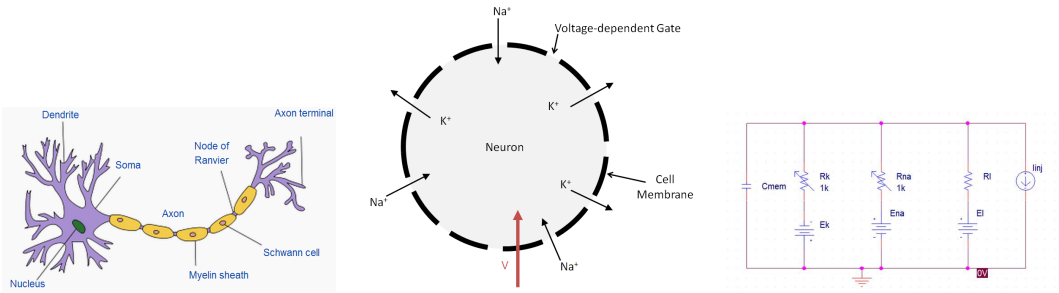
FIGURE 4.1: Generic neural system showing interaction between single neurons and the complete network. (a) represents the synaptic inputs coming from a list of other neurons. (b) represents the summation of these synaptic inputs over time; this is typically a non-linear summation with a time-decaying parameter. (c) represents the threshold that must be crossed for the neuron to produce an output "spike". (d) shows how multiple neurons may be interconnected to form a network.

In the same way that large circuits rely upon the logic gate, any large-scale implementation of a silicon neural network must begin with an efficient and effective design for a single neuron. This design process is illustrated within this chapter. In the following section the detailed structure of a neuron is described, including how action potentials are generated biologically and how these can be replicated electronically. Simplified neuron models that are commonly used are then described before the design methodology is presented. This is followed by two case studies, where the efficient designs of neuron models are illustrated.

4.1 Neuron Structures

The output of a neuron, which is generated within the soma (see Figure 4.2a), is known as an action potential (AP) or a spike. This spike is propagated along the neuron's axon towards other neurons within its local network. It is this communication of spikes that is thought to be the foundation of computation and the basis for all functionality within developed brains.

The activation of a neuron is highly dependent upon the stimuli received from its inputs. Neurons receive inputs from branches known as dendrites, which are connected to another neuron's axon through a synapse. The morphology of these dendrites is often complex and can sometimes involve 1000 connections [75]. This number of connections is far greater than typical electronic circuits implement with point-to-point techniques.



(A) (B) (C)

FIGURE 4.2: The structures of a neuron. (A) View of complete neuron structure, including soma, dendrites and axon. [134] (B) Structure of a typical soma. (C) Representation of a soma as a simple electronic circuit.

The spike itself is generated through a complex electro-chemical process, which was first discovered and documented by Hodgkin and Huxley [135]. Their model has become the foundation of many later cell models due to its highly accurate bio-physical detail.

The structure of the central part of the neuron is shown in Figure 4.2b. Alternatively, you can represent this as a very simple electronic circuit as seen in Figure 4.2c. As shown by Figure 4.2b, the central part of the neuron is enclosed by a semi-permeable membrane. This membrane creates two separate concentrations of chemical ions, an intracellular concentration and an extracellular concentration. Since the membrane is an insulator which is surrounded by two conductors it can be thought of as a capacitor, such as that shown in Figure 4.2c.

The chemical ions are able to diffuse through the membrane due to the presence of ion channels consisting of many chemical gates. The diffusion of ions causes an electrical current to be generated, as represented by the current source in Figure 4.2c. These gates are able to open or close to control the conductance of the ion channel and therefore control the amount of current. Hence, these gates are illustrated in Figure 4.2c as resistors. When the system is in dynamic equilibrium, meaning the flow of currents sums to zero, the system is at a resting voltage, as defined by the presence of the batteries within Figure 4.2c.

The gates which allow for the chemical ions to flow are activated depending upon the voltage across the cell membrane, in a similar way in which the conductance of a channel in a transistor is controlled by the voltage on its gate. When an input causes the voltage across the membrane to increase, the gates will either activate or deactivate, causing a succession of currents to flow and further voltage changes, which produces

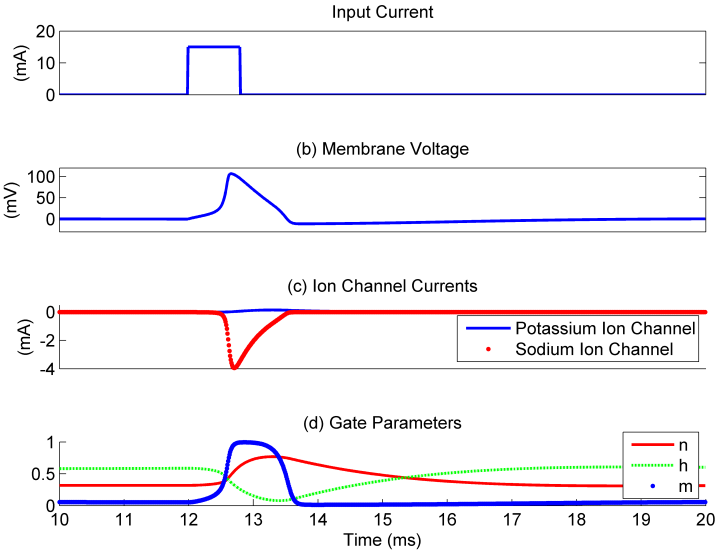


FIGURE 4.3: The characteristic values of a Hodgkin-Huxley neuron.

the spike. This process is illustrated by Figure 4.3. In Figure 4.3(a) an input current is generated, the voltage membrane is illustrated by Figure 4.3(b), the ion channel currents by Figure 4.3(c), and the gate parameter values by Figure 4.3(d).

Hodgkin and Huxley defined this process mathematically in [135]. The required equations to model a complete neuron from a squid are illustrated in Table 4.1. By tuning the parameters involved in these equations most neuron responses or behaviours can be represented.

As can be seen the model is mathematically heavy and therefore computationally expensive, typically requiring 1.2MFlops[1] per neuron [136]. Therefore, often simpler neuron models are used to increase performance. A comparison of the complexity of these neuron models is provided in Figure 4.4. Two of the most popular simplified neuron models are the leaky integrate and fire model (LI&F) and the Izhikevich model.

The leaky integrate and fire model requires only 5kFlops per neuron [136]. Within this model all input currents are summed over time and when a threshold is reached a spike is generated. The summation of currents contains a leakage element to remove long-term memory from the system. Although the model is very simple to implement it is unable to display all types of neural behaviour.

[1]Floating-point operations per second

TABLE 4.1: The complete set of equations required to model a single Hodgkin-Huxley neuron. To update the h, m and n the parameter update equations are used. All of the parameters and variables are included in the neuron update equation, which updates the overall voltage of the membrane of the neuron.

Hodgkin-Huxley Parameter Values:			
Ion	Gate Parameter, p	$\alpha(V)$	$\beta(V)$
Sodium	m	$\frac{2.5-0.1*V_m}{e^{2.5-0.1*V_m}-1}$	$4e^{\frac{-V_m}{18}}$
Sodium	h	$0.07e^{\frac{-V_m}{20}}$	$\frac{1}{e^{3-0.1*V_m}+1}$
Potassium	n	$\frac{0.1-0.01*V_m}{e^{1-0.1*V_m}-1}$	$0.125e^{\frac{-V_m}{80}}$
Parameter Update Equations: $p_\infty = \frac{\alpha_p(V_m)}{\alpha_p(V_m)+\beta_p(V_m)}$ $\tau_p = \frac{1}{\alpha_p(V_m)+\beta_p(V_m)}$ $\frac{dp}{dt} = \frac{1}{\tau_p}(p_\infty - p)$			
Neuron Update Equation: $C\frac{dV_m}{dt} = I(t) - (g_{na}m^3h(V-E_{na}) + g_kn^4(V-E_k) + g_l(V-E_l))$			

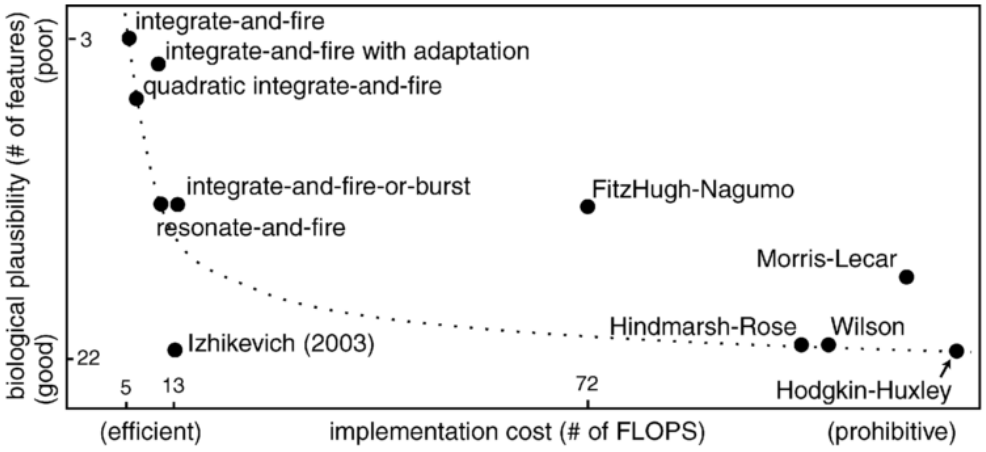


FIGURE 4.4: Comparison of neuron models illustrating their performance versus implementation cost relationship [136].

In 2003, Izhikevich [137] developed a simple model of a neuron using bifurcation theory that is able to reproduce all neural behaviours yet requires only 13kFlops [136]. The model and its design is illustrated by Figure 4.5. It has been used within large-scale network simulations [106] and is becoming increasingly popular due to its excellent performance versus cost relationship.

In section 4.3 and later, the electronic designs of both a HH and an Izhikevich neuron are

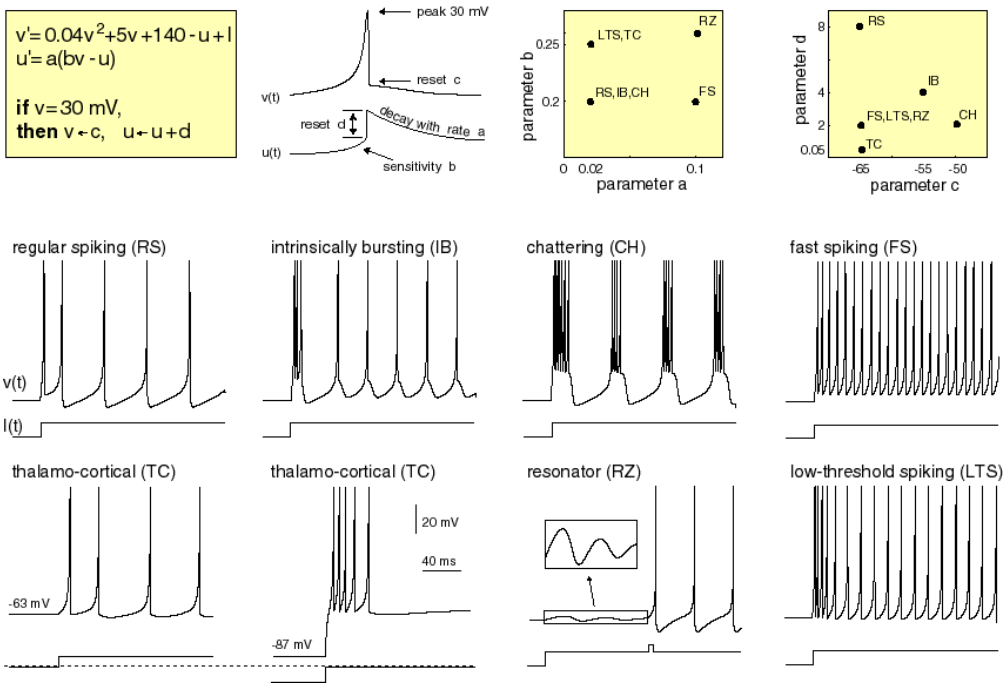


FIGURE 4.5: Izhikevich neuron model design. Electronic version of the figure and reproduction permissions are freely available at www.izhikevich.com.

described and their performance/cost ratio compared against the predictions provided by Figure 4.4.

4.2 Neural Encoding

As shown in the previous section, single neurons produce action potentials with a pattern as defined by their morphology or type. But, the full computation of a neural network is encoded within the communication of these patterns of spikes between neurons. As each neuron type may produce spikes at different times or diferent rates depending upon its input stimuli, it is important to model both the single neuron model and the complete network.

There is significant debate about how the encoding operates.

Historically rate-based coding has been the most popular technique. This coding scheme relates the number of spikes in a certain time interval to the result of the computation.

However, rate-based coding involves counting events over a period of time, which in a lot of scenarios is not feasible. For example, in reflex motions the body needs to react to sensory stimulus in the scale of milliseconds. Therefore, temporal-based coding has

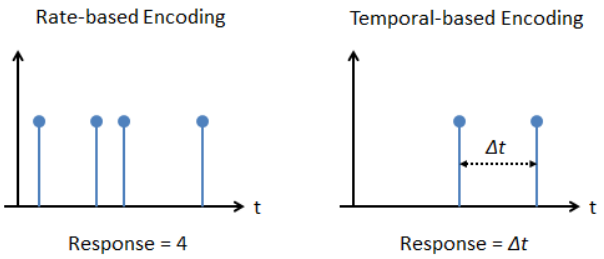


FIGURE 4.6: Differenece in neural encoding techniques.

become popular. This scheme proposes that the precise timing of spikes as well as the number of spikes may encode information.

For instance, the time to first spike hypothesis suggests that information may be encoded in the latency of a spike after the onset of a stimulus. Similarly, in correlation encoding the information may be represented as the time different between two spikes.

Figure 4.6 briefly illustrates the differences between rate- and temporal- based encoding. The choice of encoding will impact the parameter selection for a neuron model.

4.3 Methodology

Digital silicon neurons rely upon implementing a set of defined differential equations using arithmetic hardware. These equations may be implemented using software and a microprocessor, such as within SpiNNaker [108], but this introduces area, energy and performance overheads. By using a specific architecture and defined datapath the mathematical operations can be streamlined to reduce these overheads. This approach has been adopted by multiple groups, utilizing both FPGA and ASIC devices [122][138][130][119].

The high-speed operation of silicon circuits in comparison to biology allow for multiple virtual neurons to be multiplexed across a single processing core datapath. Each neuron is then allocated a specific time frame for its operation through the datapath. A neuron's parameters are stored in memory whilst other virtual neurons are being updated. This design concept is illustrated in Figure 4.7(a).

The number of virtual neurons allocated to a single processing core will impact the area, energy, latency and throughput of the system [127]. This parameter has been defined as the system granularity previously in section 3.2.3 and will be referred to as n.

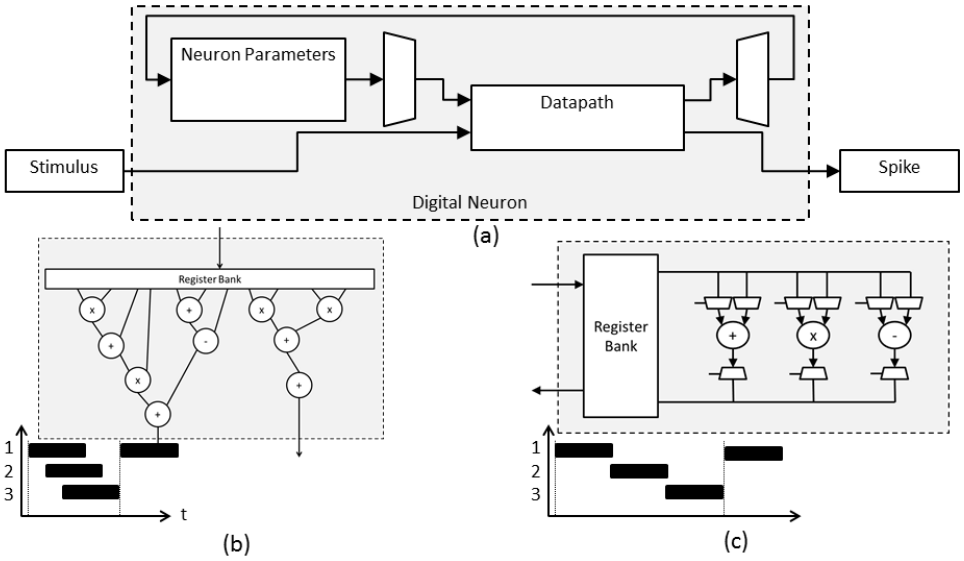


FIGURE 4.7: Digital neuron design. (a) Processing core may implement n virtual neurons through a time-multiplexing approach. (b) Maximum resource design. As many functional units as possible are utilized to reduce latency and provide maximum throughput. (c) Minimum resource design. Virtual neurons are updated sequentially, increasing system latency and reducing throughput, but utilizing less area resources.

The datapath structure consists of a combination of arithmetic logic units and memory elements. Each neural equation may be implemented using a variety of structures depending upon the design specification. For instance, if a high throughput is required with little concern for area and energy consumption then a maximum resource approach, such as illustrated by Figure 4.7(b), should be used [124]. Alternatively, to reduce area and energy consumption the datapath could be constrained to use a minimum number of resources (see Figure 4.7(c)). Next, the impact of the datapath structure and the granularity upon the system requirements is considered.

4.3.1 Implementation Parameters

Area As defined by Cassidy et al. [127] the area consumption of a neuron processing core is dependent upon the area of the datapath and the area of the parameter memory (4.1).

$$Area_{total} = Area_{datapath} + Area_{memory} \tag{4.1}$$

With increasing granularity n the area of the datapath will remain fixed, whilst the area of the memory will increase linearly. Therefore, the area per neuron will asymptotically

reduce to the memory area consumed by an individual virtual neuron's parameters. As such, to reduce the area overhead a large granularity should be utilized.

Further, the analysis of Cassidy et al. [127] can be extended to include an optimization function for the datapath structure. By utilizing a maximum resource approach the datapath can be fully pipelined, increasing the throughput and the amount of virtual neurons per processing core. Whereas, a minimum resource approach may reduce datapath size, but limit the number of virtual neurons, thereby, requiring more processing cores to be implemented for a given network size. This leads to the following relationship (4.2)

$$TotalArea = ProcessingCoreArea * \frac{NetworkSize}{n} \tag{4.2}$$

Equation (4.2) states that the total area consumed is the size of a processing core multiplied by the number of processing cores that are required to compute the function.

Area of the datapath may be estimated by calculating the total number of arithmetic units required. Area of the memory units may be estimated by calculating the number of bits required to represent a neuron's parameters.

Energy Energy dissipation is a combination of static and dynamic power [139].

Static power relates to parasitic leakage currents within transistors. Its impact can therefore be reduced by reducing the overall number of transistors in a circuit. Alternatively, power gate techniques can be used to fully disable transistors when they are not required.

Dynamic power is the power required to switch the gates in a circuit between voltage levels at a set frequency. Its relationship is shown in (4.3), where P is the dynamic power, C is the gate capacitance, V is the voltage of the transistor and f is the average switching frequency.

$$P = CV^2 f\alpha \tag{4.3}$$

Reducing the frequency of operation of the circuit will lead to power reduction. Also, a lower frequency allows for a lower voltage, which has a quadratic effect upon the dynamic power consumption.

Lowering the operating frequency will limit the number of virtual neurons per processing core. Therefore to reach a certain network size more processing cores will be required. This will increase the static power consumption.

A maximum resource datapath may utilize a slower operating frequency due to its higher throughput, reducing the dynamic power consumption. However, the increased number of transistors required will contribute towards increased static power consumption.

Finding the optimal energy performance will rely upon locating a sweet-spot in the relationship between static and dynamic power per neuron. This is illustrated later in the thesis by Figure 4.22, where the static and dynamic power performance of a neuron processing core are shown as the granularity of that processing core is varied.

Latency Latency is the time required to update a single virtual neuron. A maximum resource approach will provide a theoretical lower bound upon the latency.

To ensure working in real biological time the number of virtual neurons per neuron processing core should be constrained such that the total time to update all virtual neurons does not exceed the differential equation time step period. Equally, the simulation can be accelerated by reducing the latency.

Throughput Throughput is the rate at which virtual neurons are updated. A maximum resource approach will offer an upper bound on throughput of 1 virtual neuron updated per clock cycle if all operations can be pipelined. Reducing available resources will reduce throughput significantly, especially if the datapath is no longer pipelined. The global throughput can be increased by including more neuron processing cores.

In the following sections the optimization of two different neural models is considered: a Hodgkin-Huxley model with a focus upon closed-loop in vitro experiments, and an Izhikevich model suitable for large-scale network simulations.

4.4 Case Study 1. Hodgkin-Huxley for Dynamic Clamp

The application of dynamic clamping involves the connection of virtual neuron models to biological cells in vitro [140], forming a hybrid bio-electronic network. This allows for the functionality and behaviour of neurons to be investigated, controlled and analysed. This application is therefore of great interest to those studying the brain and significant effort is expended on the development of the electronic systems involved.

The software solutions demonstrated by Kispersky et al. [141] and Nowotny et al. [142] are flexible and easy to use, however, as has been previously highlighted section 3.1.1, software modelling neurons is not the most efficient, powerful or optimal solution.

Previously, FPGA-based designs have been utilized for dynamic clamping. For instance, in 2004 a Georgia Institute of Technology team described the simulation of conductance based models using a Xilinx Virtex 2 FPGA [143]. This work was extended in 2006 [128] and 2007 [144], whereby a methodology was illustrated that would allow for FPGAs to be used by the wider neuroscience dynamic clamping community. In an attempt to compete with the popular software-based solutions this design was then commercialized.

For complex and large-scale experiments neuroscientists may wish to model the specific and detailed morphology of multiple neurons and multiple neuronal compartments. From this perspective the designs offered by Lee et al. at Georgia have suffered due to their poor scalability, which limits their potential for simulating complex neural models. The results of the non-linear functions involved in the Hodgkin-Huxley model, described in Table 4.1, where stored in pre-determined lookup-tables (LUT). Although this allowed for quick and efficient access, it means that when the neural model involves a varied selection of neuron structures the design size grows rapidly.

An alternative approach was implemented by Mak et al. in 2006 [124]. This involved implementing the non-linear functions programmatically, allowing for the wide range of neuron structures that is often required in large-scale simulations. However, since this design utilized an iterative approach to the calculation, the throughput, and therefore number of neurons that could be implemented was reduced.

The same problem inhibits the performance of the approach offered by Zhang et al. [145]. Within this design a CORDIC [146] function is used for the non-linear functions, resulting in a latency of 170 cycles. Interestingly, Zhang et al. have utilized a floating-point architecture, whereas all other implementations have chosen to use fixed-pint arithmetic.

Luo and Mak et al. [138] demonstrated a complete bio-FPGA system, involving a Xilinx Virtex 5 FPGA connected to dissected nerve cells from the stomach of a crab. This model used a simplified version of the HH neuron, the Hindmarsh-Rose model [147]. The following sections provide details of an implementation designed to extend upon this work by integrating bio-physically realistic Hodgkin-Huxley neurons into the bio-FPGA system. In order to allow for this, it is imperative that an efficient and scalable FPGA-based design, which allows for programmable neuronal simulations, is used. This requires a system with the latency offered by [144] but with the flexibility of [124]. The conceptual analysis leading to the determination of the optimal design parameters is illustrated, followed by the design of the complete neuronal simulation IP block that can be incorporated into a wider system.

4.4.1 Parameter Investigation

The implementation of a Hodgkin-Huxley neuron requires equation (4.4) to be implemented, along with parameters and equations from Table 4.1.

$$C\frac{dV}{dt} = I(t) - (g_{na}m^3h(V - E_{na}) + g_kn^4(V - E_k) + g_l(V - E_l)) \tag{4.4}$$

For dynamic clamping the objective is in maximizing the number of neurons that can be implemented in real biological time using a single FPGA device. As mentioned in the previous section this is dependent upon the area occupied by each neuron and the throughput of each datapath. By selecting the correct datapath design the area overhead can be minimized and a suitable throughput of data provided.

To determine the correct datapath design an area-delay model was developed for each arithmetic component, based around a Xilinx FPGA implementation. Multiplication,

TABLE 4.2: Area and Delay costs for different arithmetic components implemented upon a Xilinx FPGA. Number of operations required of each type for a single Hodgkin-Huxley Neuron.

Arithmetic Unit	Delay Cost	Area Cost	Operations Required
Multiplication	3	1	23
Addition	1	1	13
Subtraction	1	1	17
Division	8	8	15
Exponential	8	8	6

addition and subtraction can all be implemented upon the pre-configured DSP components of a Xilinx FPGA. As such, this is used as a base component for area consumption. A state-of-the art Xilinx Virtex-7 FPGA contains 3600 DSP components [148].

Addition and subtraction typically require a single clock cycle to compute, whereas a multiplication requires 3 cycles.

Unfortunately, exponential and division operations require extra circuitry. It is estimated that each exponential and division block will require the equivalent of 8 DSP components and have a latency of 8 cycles (see section 4.4.2.1 for derivation).

This information is summarized in Table 4.2, along with the operation count of a typical HH neuron. This information can be used to determine the area/delay cost for three different implementations, and from this figure create a relative estimate of the number of neurons that can be implemented upon a single FPGA. This information is supplied in Table 4.3.

Approach A: Maximum Resource An individual arithmetic unit is instantiated for each operation. Requiring a total of 221 DSP units. The critical path is 36 clock cycles with a throughput of 1 neuron per clock cycle. Assuming a clock frequency of 100MHz this equates to a latency of 360ns, and in a biological time-step of 100us, 10,000 virtual neurons can be updated per processing core. Each FPGA may contain 16 neuron processing cores for a total of 160,000 computational neurons.

TABLE 4.3: Comparison of three different datapath structures for implementing a Hodgkin-Huxley neuron model.

Approach	Area (Units)	Latency (Cycles)	Throughput (Neurons / Cycle)	n, Virtual Neurons / Processing Core	No. Processing Cores	Total Neurons
A	221	36	1	10,000	16	160,000
B	19	120	0.01	83	189	15,000
C	43	38	0.33	3,333	84	280,000

Approach B: Minimum Resource In the minimum resource approach it is assumed that only one of each arithmetic component is instantiated. The critical path can be estimated by the longest delay through a single arithmetic unit. In this case, 15 division operations are required with a latency of 8 clock cycles, giving a total latency of 120 cycles. Since only one neuron may be updated within each frame the throughput is 1/120 neurons per cycle and therefore only 83 virtual neurons can be implemented. The total area consumption is 19 DSP units, allowing for 189 neuron processing cores per FPGA. With 83 virtual neurons per core this equates to approximately 15,000 computational neurons in total.

Approach C: Compressed The calculation of the Hodgkin-Huxley neuron can be compressed into the three identical sub-calculations (see section 4.4.2 for further details). As such, this sub-calculation can be implemented in a maximum resource approach and then three consecutive updates can be combined into a single neuron response. Using this technique the resources are significantly reduced to only 43 DSP units, whilst maintaining a relatively high throughput of 0.33 neurons per cycle. As such, 280,000 neurons can be implemented per FPGA.

4.4.2 Design

Clearly, the most efficient implementation option is approach C and a detailed description of this design is provided here. The design was implemented using Xilinx System Generator, commonly used by many FPGA-based neural models [124][128][138]. This

design tool is an extension to the Simulink platform and allows for rapid prototyping and an efficient design structure.

The primary Hodgkin-Huxley equation (4.4) can be simplified to Equation 4.5.

$$C\frac{dV}{dt} = I(t) - (I_{Na} + I_k + I_l) \tag{4.5}$$

where $I(t)$, I_{Na}, I_k, I_l are the injected, sodium channel, potassium channel and leakage currents respectively. The ion channel currents can in turn be calculated using (4.6). Although this appears simple, the $p_n(V_m)$ components are non-linear time varying rate functions as illustrated in Table 4.1. It is the calculation of $p_n(V_m)$ that is the most computationally expensive, and that has previously been enumerated into LUTs [144]. However, as mentioned LUTs do not scale effectively in terms of neural model complexity or in terms of precision. As such, it is beneficial to calculate these values on-the-fly.

$$I_c(t) = g_c * p_{n_1}(V_m)^{k_1} * p_{n_2}(V_m)^{k_2} * (V_m - E_c) \tag{4.6}$$

At each time step the values of $p_n(V_m)$ should be updated according to (4.7), which is a simplified version of that listed in Table 4.1. The α and β functions depend upon the gate type that is implemented and takes the form of either a sigmoid function or bell curve, as illustrated in equations (4.8) and (4.9). Different gate types, such as sodium or potassium, require different constants, p_1-p_7.

$$\frac{dp}{dt} = \alpha_p(V_m)(1-p) - \beta_p(V_m)p \tag{4.7}$$

$$\alpha_x(V) = \frac{p_1 - p_2 * V}{e^{p_3 - p_4 * V} - p_5} \tag{4.8}$$

$$\beta_x(V) = p_6 e^{p_7 * V_m} \tag{4.9}$$

The architecture for computing $p_n(V_m)$ forms the main datapath and is illustrated in Figure 4.8a. This gate updater datapath is integrated within the complete system as

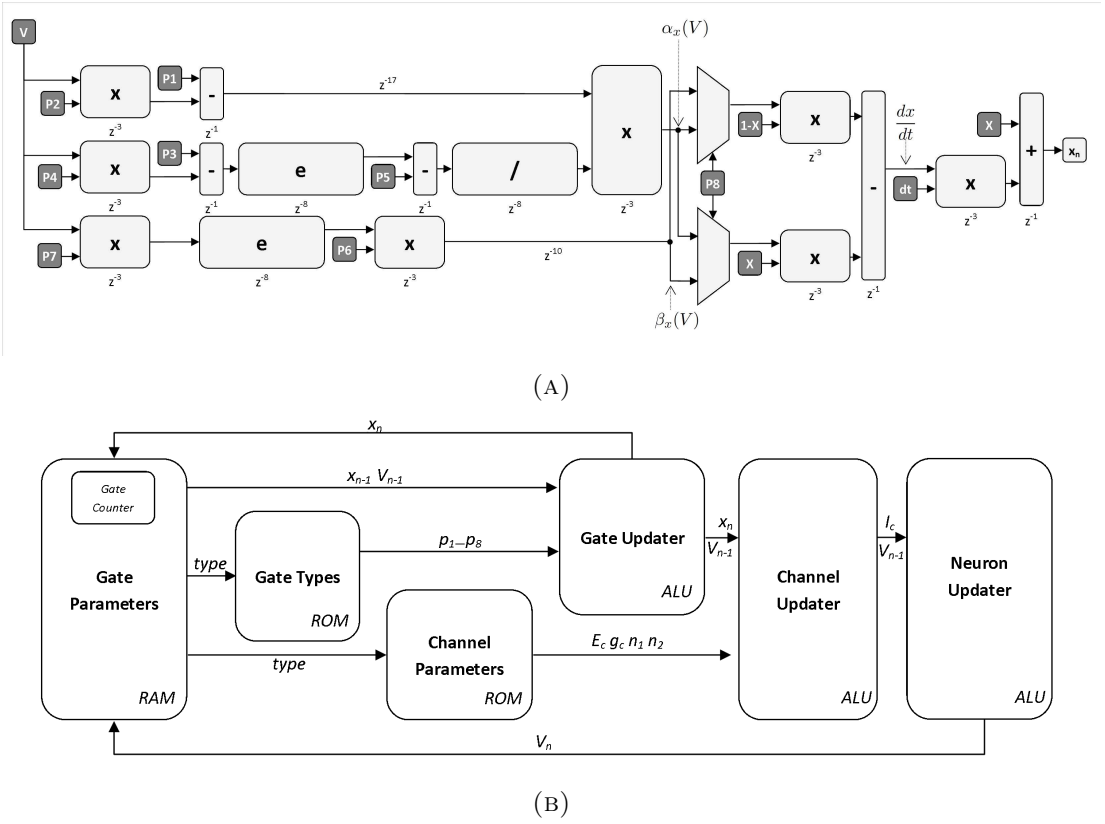


FIGURE 4.8: Hodgkin-Huxley FPGA-based implementation overview. (a) Gate updater arithmetic datapath. (b) Complete system.

illustrated in Figure 4.8b. The constants p_1-p_7 are delivered to the gate updater from the gate types ROM. For each gate within the complete system a separate list of variables is held within the Gate Parameters RAM. On each clock cycle a different set of gate variables, along with its associated gate type constants, is fed into the gate updater datapath.

The parameter p_8 in Figure 4.8a controls a switch between an activating or inactivating ion gate. These two opposing gate types are illustrated within the sodium ion channel equations in Table 4.1.

Following the updating of the gate values, multiple gates are combined in the Channel Updater datapath, and then multiple ion channels are combined in the Neuron Updater datapath. A typical HH neuron contains three gate types, m, n, h, and two ion channels, Sodium and Potassium. As such, each complete neuron is updated every three clock cycles.

The $\frac{dx}{dt}$ operations are performed using a simple Euler integration technique with a timestep of $25\mu s$. This timestep value was found to reduce the error in the calculations

by Bettencourt et al. [99].

By defining a list of gates and their associated neuron connections, it is possible to programmatically produce neuron models to be simulated using this dedicated hardware architecture. For large-scale simulations there may be a requirement to develop a compiler that is capable of mapping defined neural topologies to the correct memory locations.

4.4.2.1 Arithmetic Units

The exponential and division operations are a key component of a Hodgkin-Huxley-style neuron. There currently exists no fixed-point exponential or division function blocks within the System Generator library and previously it was thought the implementation of such a block would be too costly in terms of area and speed [143]. However, with the increase in resource availability on modern FPGAs and with the development of new algorithms this is no longer the case [149].

For the purposes of this application only an approximation of both the exponential and division operations are required. This is due to the approximation already introduced by fixed-point architecture.

It is imperative that the calculations of both complex functions are able to be fully pipelined and have a low latency to conform with the chosen design methodology. For these reasons a system of lookup tables (LUTs) was chosen to be the most suitable implementation. These LUTs were able to be reduced in size through the use of interpolation and judicious selection of input range, output precision and memory arrangement.

For example, the exponential function uses a system of three LUTs. The first LUT stores the result of the exponential function at regular intervals and a second LUT stores the gradients between these intervals. The result can then be computed using interpolation by combining the output of the two LUTs. However, this method has a tendency to overestimate the result because of the linearisation of gradients between intervals. It was observed that this error can be compensated for by adjusting the gradient by a ratio which depends upon the distance of the point to be calculated from the index value in the first LUT. Since the ratio values are constant throughout all the intervals it is possible to store these ratios within a third LUT.

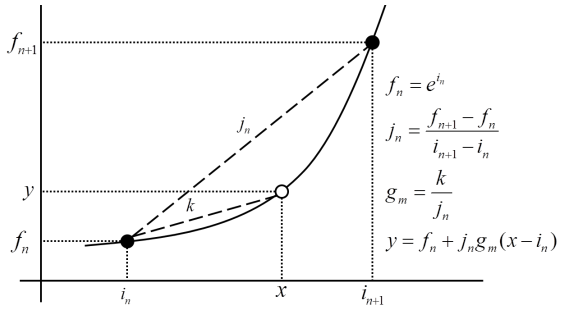


FIGURE 4.9: f
Illustration of the gradient error fix technique for implementing fixed-point exponential function. The values of f, j and g are pre-stored in LUTs.

This approach is expressed in (4.10), where the values f_n, j_n, and g_m are those that are stored within LUTs, n and m are indices to the LUTs, i_n is the value stored within the LUT that is nearest to the original input value, that is represented as x, y is the calculated result. This design is illustrated by Figure 4.9.

$$y = f_n + j_n g_m (x - i_n) \tag{4.10}$$

Table 4.4 shows a comparison between this technique, a pure LUT-based option and the optimized design presented by [149].

The GEF approach using LUTs was used in this design because:

- it uses a comparable number of resources as [149]
- it offers the reduced latency of the implementation, offering a higher throughput of neurons
- it conforms to the fixed-point design, whereas [149] uses floating-point
- the error introduced is negligible
- it conforms to the design tool approach used, that being Xilinx System Generator

4.4.3 Results

Initial simulations suggested a 28-bit signed fixed-point system with a 14-bit fraction provided a sufficient level of resolution and range for most operations, although, the

TABLE 4.4: Comparison of methods for implementing exponential function.

	Our Design	LUT	[150]
Format	28-bit Fixed Point		32-bit Floating Point
Arithmetic	2 Mult, 2 Add	0	1 Mult, 3 Add
Memory	16kbits	15Mbits	18kbits
Latency	8 Clocks	1 Clock	15 Clocks

design requires minimal rework to alter the number system used. Previously, Graas et al. have used a 32-bit fixed point value [143]. Within a dynamic clamping application the resolution of the input/output is constrained to between 12-16 bits depending upon the ADC/DAC used [138][151][152].

4.4.3.1 Synthesis Results

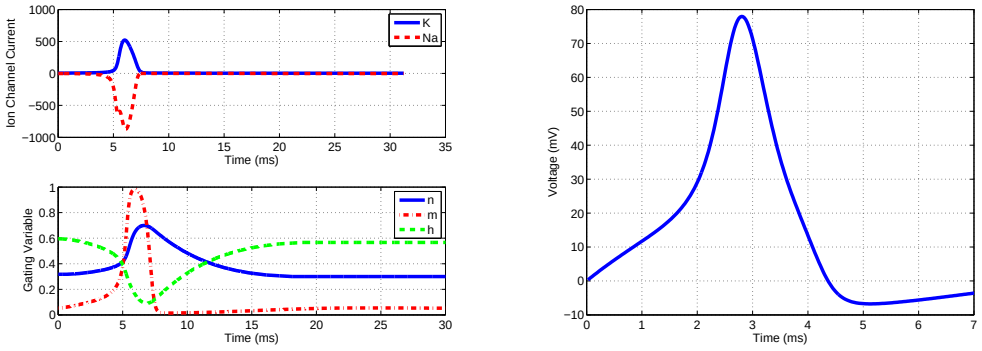
The design was synthesized for a Xilinx Virtex7 FPGA using the System Generator [153] and Design Suite [154] software tools provided by Xilinx. The key synthesis results are provided within Table 4.5. It can be seen that despite the addition of what was previously thought of as complex functionality, namely fixed-point exponential and division operations, the design uses minimal resources. After placing and routing the maximum clock frequency of the system was determined to be 240MHz.

Table 4.5 also provides estimated synthesis results for a LUT-based design, in the style of [144], with the same capabilities of the programmable design described in section 4.4.2. It can be seen that in order to simulate an equal number of neuron types the memory usage is significantly larger than the programmable approach. If a neural model was to include such a wide range of channel types the LUT-based approach would be severely limited in its capabilities, in fact the resources of even a modern FPGA may become exhausted.

Also shown in Table 4.5 are results from [124] where a similar design was implemented. The iterative nature of this design breaks the pipeline and has a significant affect upon the latency of the calculation, thus reducing the overall throughput that would be achievable on a modern FPGA.

TABLE 4.5: Resource usage comparison for different design approaches. The figures for the LUT-based approach are extrapolated from smaller models.

Resource	Prog.	LUT	[124]
Device	Virtex7	Virtex7	Virtex2
Slices	836	1200	2800
BRAM	11	2000	1
DSP48E1s	66	24	-
Clock Speed	240MHz	240MHz	68MHz
Latency (Clock Cycles)	40	10	73
Pipelined	Yes	Yes	No



(A) (B)

FIGURE 4.10: FPGA-based neuronal dynamics. (a) Simulated ion channel results (top) and gate variables (Bottom). (b) Action potential produced by FPGA-design.

4.4.3.2 FPGA-based Neuronal Dynamics

Figure 4.10 illustrates the intermediate values of the Hodgkin-Huxley equations along with the final output voltage of the neuron from a simulation of the completed design. A simulated voltage from a spiking neuron was fed into the ion channel simulator and the gating values and their associated currents were calculated. The ion channel currents are then combined to produce the membrane voltage response of the neuron.

Figure 4.11 shows the results from an experiment to investigate the impact of translating the Hodgkin-Huxley equations from a high-precision software-based platform to a fixed-point hardware implementation using low accuracy Euler integration techniques. In this figure, the x-axis shows the injected current into the simulated neuron, for each value of injected current the spike frequency, commonly thought to be a key measure of neural activity was recorded; this is plotted on the y-axis. It can be seen that the hardware and

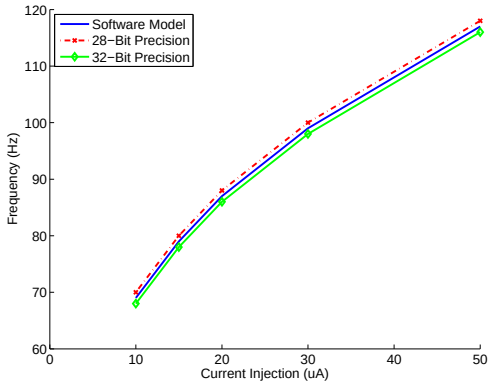


FIGURE 4.11: Spike frequency against current injection.

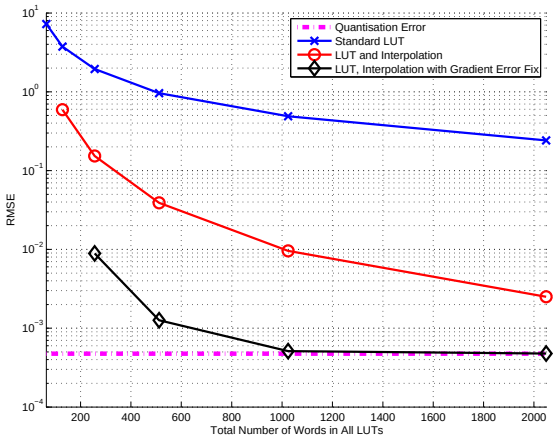


FIGURE 4.12: Comparison of error against the total size of LUTs used by three different methods to calculate the exponential function.

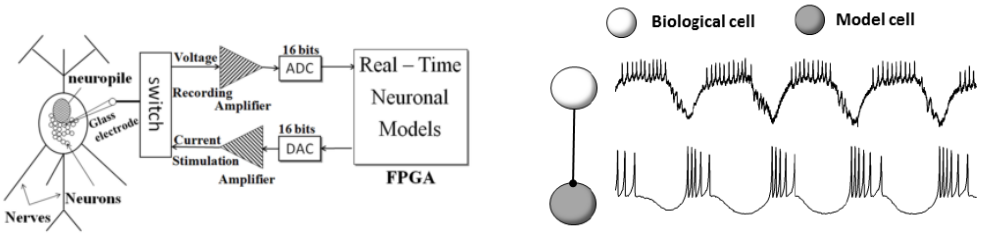
software variations broadly coincide, indicating an acceptable level of accuracy within the system. This analysis approach has previously been utilized within [108][155][156].

4.4.3.3 Exponential Unit Optimization

The improvement in the accuracy of the exponential function by using the GEF method is highlighted by Figure 4.12. It can be seen that by using GEF the error can be reduced to the same order of magnitude as the 28-bit quantisation error by using just 1024 words of memory.

4.4.4 Dynamic Clamp

The work illustrated in section 4.4.4 was completed by Jun Wen Luo, it is included here to illustrate the concept of an FPGA-based dynamic clamp



(A) (B)

FIGURE 4.13: Example FPGA-based dynamic clamp (a) experimental configuration as shown in [138], (b) experimental results taken by project colleague Junwen Luo

and the associated results which were obtained.

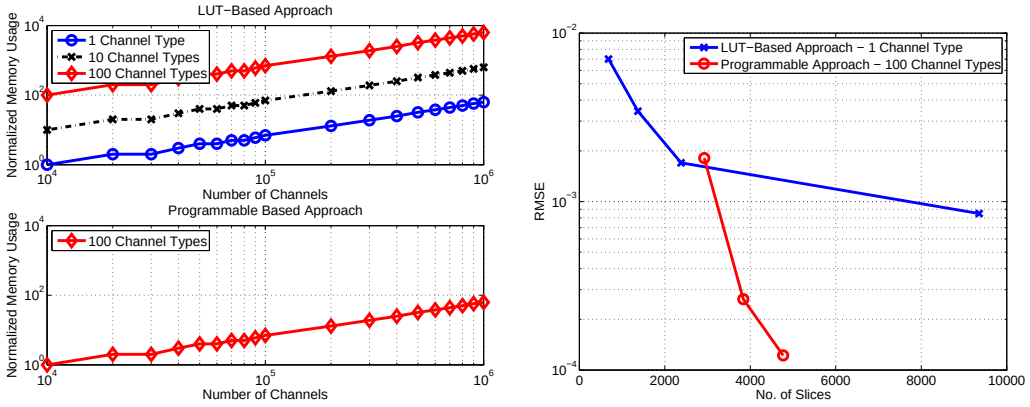
As mentioned at the start of this section, a dynamic clamp involves the connection of virtual neuron models to biological cells in vitro, forming a hybrid bio-electronic network with the intention of studying neuron behaviour at a cell and circuit level [157].

This virtual neuron can be implemented upon an FPGA, as highlighted in [138] and shown in Figure 4.13. In this experiment cells taken from a crab are connected to an FPGA using a single electrode. This electrode is used to monitor the voltage of the neuron, the voltage is then input into the real-time neuronal model, such as the Hodgkin-Huxley model described here. The model calculates an output current which is fed back to the neuron using the electrode. A sample response showing the synchronisation between the real biological cell and the model cell on the FPGA is shown in Figure 4.13b

4.4.5 Discussion

The introduction of this design into a dynamic clamp application allows for greater scalability and for more complex models to be connected to the in-vitro neurons than current options allow and without the added inherent architectural flaws of software-based approaches such as high latency and jitter.

The importance of scalability when designing neuromorphic hardware is rapidly increasing due to the recent advancements in neural model complexity. The programmable approach described in this thesis can be compared in terms of scalability against the previously popular LUT-based method utilized by the Georgia Tech group as mentioned on page 52.



(A) (B)

FIGURE 4.14: FPGA-based Hodgkin-Huxley model design scalability. (a) Scalability of memory usage for increasing model complexity. (b) Scalability of resource usage with increasing model accuracy.

As shown by Figure 4.14a using a programmable approach offers significant rewards when the model complexity grows, due to the quadratic rises in memory requirements of the LUTs in an approach similar to [128]. For each new type of gating variable introduced into the neural model the LUT-based approach requires a new block of memory to store the gating variable values. This increases the size of the channels simulator and therefore reduces the amount of times the simulator can be replicated across the whole FPGA, dramatically impacting upon the number of neuron model types that can be simulated. However, a programmable approach is capable of calculating the values dynamically and hence there is no growth in resource usages as the number of gating variables or neuron model types grows.

Although the number system selected for this design gave adequate precision, it is easily configurable to add extra resolution if required. For example, due to the large number of ion channels that can be simulated the design could be utilized for simulations of detailed models of multi-compartmental neurons [143], for which extra resolution could be beneficial. Figure 4.14b shows how the resource usage of the two contrasting approaches to neuron simulation scale with increasing resolution. In this figure all memory and DSP elements have been translated into slices. Using a pure LUT-based approach to reduce the error in the model by half the size of the LUTs must be doubled and this has a dramatic effect upon the overall number of slices. Whereas to reduce the error in a programmable approach the wordlength of the arithmetic calculations can be increased. Increasing the wordlength has less impact upon area as opposed to doubling the size of

the LUTs.

The design offered by [124] would scale in terms of area similarly to the approach offered here. However, the design in [124] used an iterative approach to calculate the result of the exponential function. This iterative approach requires an extra clock cycle for every additional bit of precision which is required. Therefore, increasing the resolution to achieve greater precision would have a detrimental impact upon the overall latency of the system.

4.5 Case Study 2. Izhikevich

Due to the complexity of the Hodgkin-Huxley equations some large-scale simulation projects have chosen to simplify the neural model, with the Izhikevich model the most popular variant [137]. This model has been implemented upon the SpiNNaker system [108], the BlueHive platform [130] and the 1 million neuron model illustrated in [106].

4.5.1 Previous Work

Jin et al. [108] described the process of translating the Izhikevich model from floating-point to fixed-point arithmetic. The fixed-point model was then implemented upon a low-power ARM processor. They looked to take advantage of this architecture by utilizing the available arithmetic operations, specifically a multiple and accumulate operation that could complete in a single cycle. They also introduced a common simplification that assumed that the time update period was 1ms, simplifying the integration of the equations. Specifically it removes a multiplication from the Euler integration.

Jin et al. [108] translated the equations in Figure 4.5 into the form of (4.11) and (4.12).

$$v = v(0.04v + 6) + 140 + I - u \tag{4.11}$$

$$u = -au + u + abv \tag{4.12}$$

If the membrane voltage v exceeds a threshold a spike is produced and the values v and u are reset according to (4.13) and (4.14).

$$v = c \tag{4.13}$$

$$u = u + d \tag{4.14}$$

The developed fixed-point model was extended by Fox et al. [130] onto an FPGA and a similar design was utilized by Imam et al. [121] in their asynchronous CMOS implementation.

Fox et al. [130] suggested using a large granularity with the processor operating in real biological time. The neuron parameters were stored in off-chip memory [130].

Imam et al. [121] suggested that storing parameters off-chip introduced significant memory bandwidth limitations, and as such, used local memory. They however, considered using a granularity of only 1 and only operating the design in quicker than real-time mode. They therefore did not consider the impact of storing additional neuron parameters.

Cassidy et al. [122] did consider storing of neuron parameters on-chip, but their development of an optimal granularity considers mainly FPGA-based design and considers only area consumption.

4.5.2 Design

The Cassidy model is extended here to include consideration of energy consumption, in order to provide a detailed comparison of a dedicated digital Izhikevich model, with equivalent microprocessor and analogue implementations.

For the optimal implementation two further simplifications are added to the equations developed by Jin et al. [108]. Firstly, the area overhead can be limited by reducing the number of arithmetic operation types. For instance, by calculating the value of negative u in (4.12) the subtraction operation can be removed from (4.11). This forms the following alternate equations:

$$v = v(0.04v + 6) + 140 + I + (-u) \tag{4.15}$$

$$(-u) = (-u)(-a + 1) + (-ab)v \tag{4.16}$$

The values of $-a$ and $-ab$ are neuron parameters and can be stored in memory and $(-u)$ is always used in its negative form, this negates the need for a subtraction unit.

Secondly, previous implementations have stored all of the required parameters as 16-bit values within the parameter memory [130]. The parameters a, b, c, and d (defined in Figure 4.5) alter the function of the neuron model, for instance, from a regular spiking neuron to a thalamo-cortical neuron. Therefore, for the 4 parameters and the 2 variables a total of 96-bits of memory is required per neuron. In fact, Ambroise et al. [158] store 9 parameters each at 18-bit resolution. However, as illustrated in Figure 4.5 each parameter may take up to 4 possible values. Despite this limit all known spiking types can be illustrated.

Therefore, only a 2-bit pointer is required to define which value each should take. This 2-bit pointer can then be used to index a small LUT containing the 4 correct 16-bit values for each parameter. This method reduces the required memory per neuron from 96-bits to only 40-bits. This is illustrated by Figure 4.15 whereby the total memory required for 8 neurons is reduced from 768-bits to 576-bits. This improvement will increase with a greater number of neurons.

Similarly to the previous case study, for large-scale system modelling minimizing the area consumed by the arithmetic datapath required to complete the neuronal equations is a primary objective. Also, as mentioned previously, there are two primary options for implementing the datapath, minimum and maximum resources.

Interestingly two recent approaches to designing digital Izhikevich neuronal datapaths have chosen different approaches. Cassidy et al. [122] utilized a maximum resource system whereas Imam et al. [121] used only single arithmetic units.

A design estimation comparing the two approaches is illustrated in Table 4.6. For this case study a VLSI system is targeted and as such the area delay costs are updated from what was previously shown in Table 4.2. The base unit for comparison now becomes a

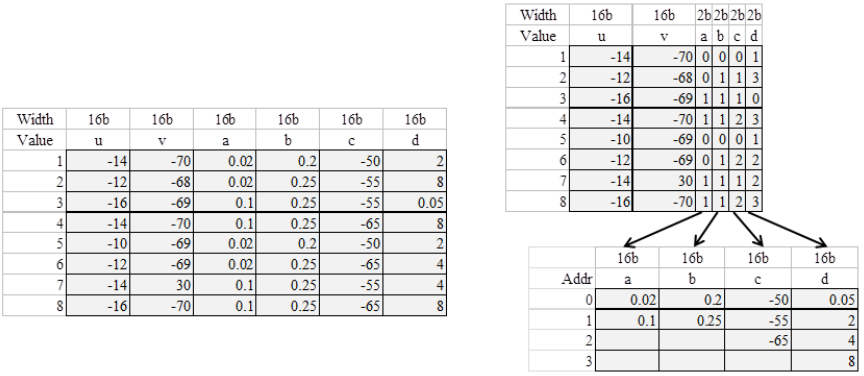


FIGURE 4.15: Simplification of Izhikevich neuron memory structure. On the left, the traditional method involving 96-bits per neuron. On the right, the simplification suggested here involving only 40-bits per neuron.

TABLE 4.6: Area/Throughput estimation comparison between maximum resource and minimum resource approach for Izhikevich neuron arithmetic datapath.

		Maximum Resource		Minimum Resource	
Design Unit	Cost per Unit	Number	Total Cost	Number	Total Cost
16-bit Reg.	1	17	17	8	8
32-bit Reg.	2	14	28	4	8
Mult (16-bit)	18	4	72	1	18
Addition (16-bit)	4	6	24	1	4
Total Cost			141		38
Throughput (Relative)			1.00		0.10
Equivalent Throughput (Relative)			1.00		0.37
Latency (Clock cycles)			5		10

16-bit data register. Using the *Synopsys Design Vision* synthesis tools it is estimated that a 16-bit multiplication unit is 18 times larger than a 16-bit data register, and a 32-bit addition unit is 4 times larger.

As Table 4.6 shows, the maximum resource approach is approximately 3.7x larger than the minimum resource approach. However, its throughput is 10x greater. If both designs are normalized to utilize the same area resources, a maximum resource approach has an equivalent throughput 2.7x greater.

For efficient large-scale simulation or low-power neuroprosthetic implementation another primary objective is minimizing the energy consumption for the design. To implement the same number of neurons the minimum resource approach requires approximately

2.7x more transistors than the maximum resource approach. Therefore, its static and dynamic power will be approximately 2.7x larger[2]. Alternatively, the minimum resource approach could be clocked 10x faster to achieve the same throughput as the maximum resource approach. This will have the effect of increasing the dynamic power by 10 but reducing the static power consumption by 3.7. To verify the area model and provide comparative analysis of the energy consumption for each approach the design of both options is detailed in the following section.

Both of the following designs were implemented using VHDL and the Xilinx ISE software package. All variables are stored using 16-bits. The multiplication units received 16-bit inputs and produced a 32-bit result, whilst the addition units used 32-bit inputs and gave a 32-bit result.

4.5.2.1 Maximum Resource Approach

In a maximum resource approach as many arithmetic units are used as required. The objective is to minimize the critical path in order to reduce the latency of the computations. The design of a maximum resource Izhikevich neuron is illustrated in Figure 4.16. Each arithmetic operation has a latency of 1 clock cycle, therefore the critical path is 4 clock cycles. Added to this is a single clock cycle at the end of the critical path to determine whether the membrane voltage requires resetting after a spike has been produced.

As well as making significant use of arithmetic units the maximum resource approach requires a significant number of data registers. This is due to the requirement to stall input data until it is required by the datapath. For instance, two sequential single cycle 16-bit registers are required to stall the value of parameter d before it can be used in a calculation.

The operations are scheduled to reduce the overall number of registers used. For example, the addition of I and $-u$ is perfomed as soon as possible to remove the requirement to store these variables in internal registers.

Due to the pipeline nature of the maximum resource approach a low clock frequency can be utilized. For example, for a granularity of 8, 5 cycles are required to calculate

[2]Assuming that the switching loads remains similar in both designs.

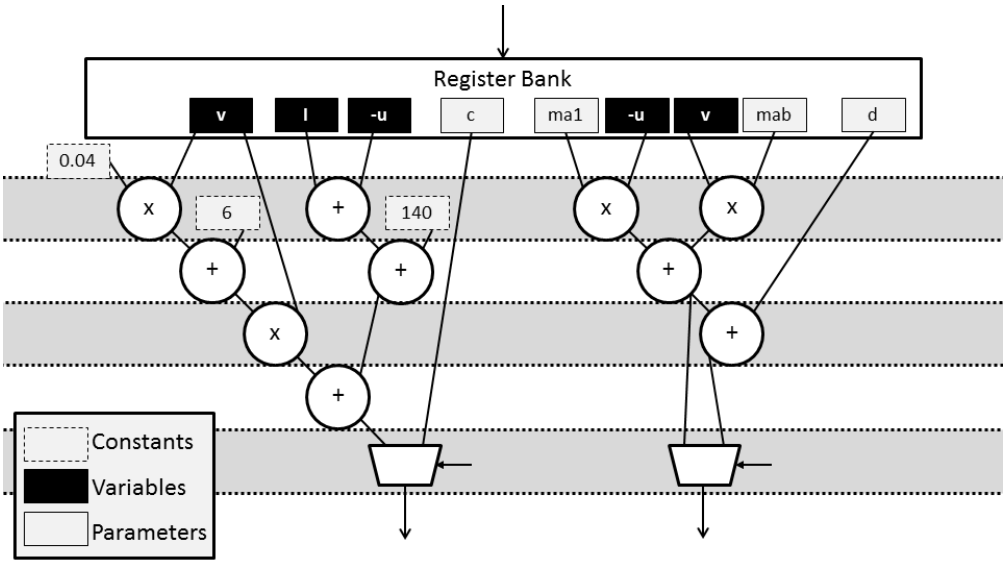


FIGURE 4.16: Maximum resource based approach of Izhikevich neuron model.

the first neuron, and then on each successive cycle another neuron is updated. So only 12 cycles are required in total.

The datapath may be combined with a dual-port memory to allow for continuous operation. In this mode a neuron's parameters are read at the same time as a previous neuron's parameters that have just been updated are wrote into the memory. Alternatively, a single-port memory can be used with an interleaving of read and write operations. This may reduce the throughput of the datapath by 2.

4.5.2.2 Minimum Resource Approach

For the minimum resource approach a single addition and multiplication unit is used. Each neuron is allocated a set time frame for its computation. The total latency of the design is a multiple of this time frame. For example, if the time frame is 10 cycles long and there are 10 neurons then a total of 100 cycles are required. Therefore, to reduce the latency and provide the highest throughput possible the time frame duration must be minimized. To achieve this it is important to allocate and schedule the operations upon the arithmetic units in the optimal manner.

For this optimization a list scheduling algorithm was used with a data dependency based priority scheme. A left-edge algorithm was used to reduce the number of required

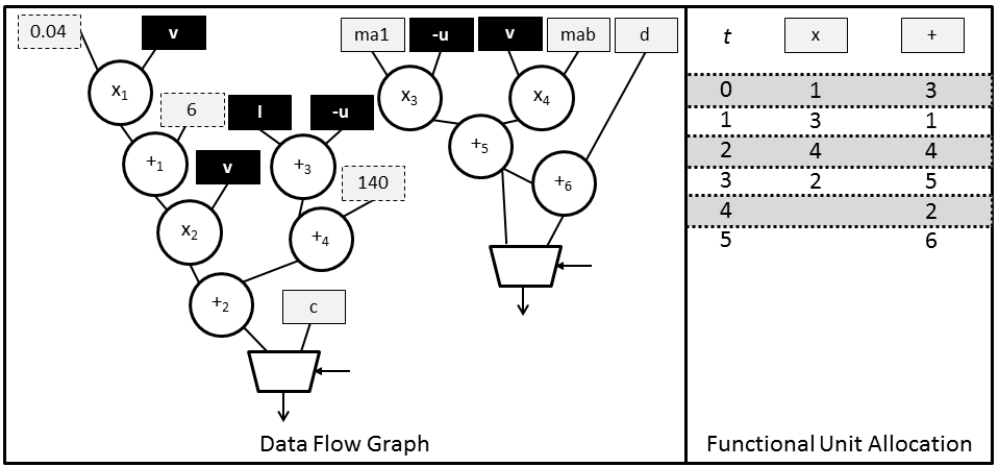


FIGURE 4.17: Data flow graph and functional unit allocation for the Izhikevich neuron model. The indices in the data flow graph relate to the value in the functional unit allocation.

registers. The allocation of operations to functional units is illustrated by Figure 4.17. Within this implementation each arithmetic operation requires a single cycle to complete.

4.5.2.3 Datapath Controller

Each of the datapaths, both maximum- and minimum- resource approaches described above, require an interface unit to provide the required data at the appropriate time. This interface is known as the datapath controller. The operation of this unit is similar for both approaches.

The datapath controller is responsible for feeding the data into the datapath and retrieving the result. It must interface with the memory which stores the neuron's parameters. The interface connections are illustrated in Figure 4.18.

The datapath controller has an internal clock and 2 counters incremented on each clock cycle. These counters are used as pointers to locations within the memory. At each memory location the associated parameters of a single neuron are stored. These parameters are retrieved in the correct clock period, before they are passed to the datapath. Once the datapath has completed the calculation upon these parameters the updated values are placed back into memory at the original location.

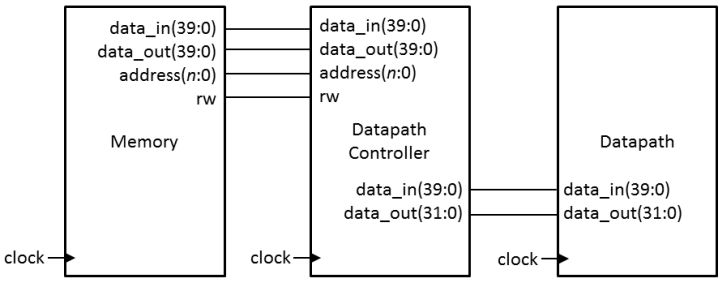


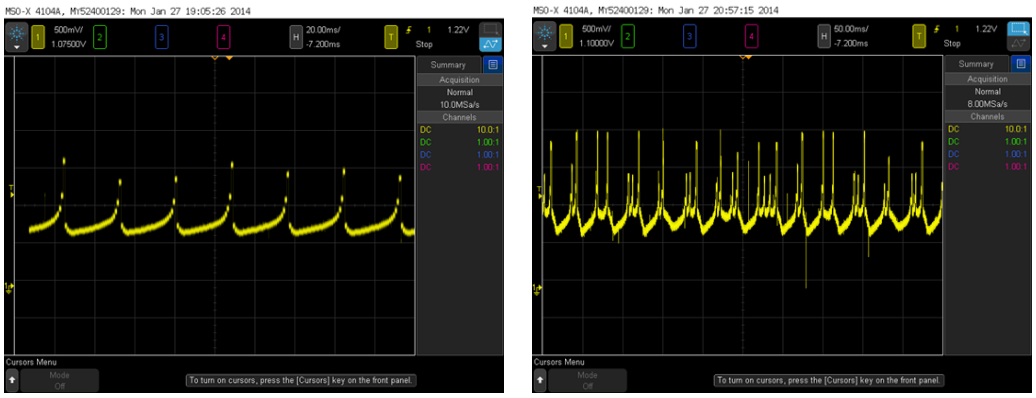
FIGURE 4.18: Interface connections between modules. The datapath controller fetches instructions and parameters from memory and feeds the datapath at the correct time.

The memory contains all the parameters for all the neuron that are implemented upon the processing core; i.e. each processing core has its own local memory and no global memory accesses are required. Within each 1ms time period the value of each neuron is updated by the datapath. Any neurons that have produced an output greater than a set threshold will produce an action potential.

4.5.3 Results

Both designs were fully implemented and tested on a Xilinx Virtex-7 VC707 Evaluation Board containing a Virtex-7 xc7vc485t FPGA. To illustrate the neuronal dynamics the FPGA was connected to a digital-to-analogue (D/A) converter, which in turn was connected to an Agilent Oscilloscope. The D/A converter received an 8-bit digital value from the FPGA through a parallel bus connection. The D/A converted this to a scaled approximation of the membrane voltage of a neuron. The spiking patterns from two different neuron types are illustrated in Figure 4.19.

To verify the performance of a fixed-point design a floating-point implementation of the same model is compared. A varying level of stimulus current was injected into a regular spiking neuron and the number of spikes over 1sec was recorded. As can be seen in Table 4.7, the spike count remains the same with a Matlab double-precision floating-point implementation and a 16-bit fixed-point model. This demonstrates that a fixed-point model can demonstrate a similar level of accuracy to a floating-point implementation at a functional level, despite the differences in actual calculated values of the membrane voltage.



(A) (B)

FIGURE 4.19: Membrane voltages from an Izhikevich neuron model implemented upon an FPGA. (a) Regular spiking. (b) Chattering

TABLE 4.7: Comparison of number of spikes in 1 second with a varying input current for a floating-point and a fixed-point implementation

Input Current (mA)	Floating-Point	16-bit Fixed Point
5	12	12
10	26	26
15	39	39
20	52	52
25	65	65

TABLE 4.8: FPGA Resource Utilization for the two different design approaches. Implemented upon an Xilinx Virtex-7 Device.

	Maximum Resource	Minimum Resource
Slices	29	217
DSP48E1s	5	2
Clock	506MHz	306 MHz

4.5.3.1 FPGA Implementation

Although the primary objective is in an Izhikevich model for ASIC integration an FPGA implementation can be used to compare with previous designs. In Table 4.8 the resource usage of the two different datapath designs are compared. As expected, the minimum resource design utilizes fewer DSP components. However, significantly more configurable logic blocks [3] are used. This increase is caused by the multiplexing required when using a limited number of arithmetic units.

[3]Each configurable logic block contains 2 slices, each of which contains 4 LUTs and 8 flip-flops [160]

TABLE 4.9: Comparison between FPGA implementations of Izhikevich neuron model.

Approach	FPGA	Virtual Neurons/Processing Core, n	Slices	DSP	RAM (kb)	Clock Freq. (MHz)	Design details
Max Resource	Virtex-7	128	61	5	5.1	506	Fixed-point
Cassidy et al. [122]	Spartan-3	8	500	1	5.6	80	Fixed-point
Thomas et al. [159]	Virtex-5	1024	2859	16	-	307	Floating-point
Ambroise et al. [158]	Virtex-4	117	473	1	-	84	Fixed-point

In fact, Xilinx state that pipelined designs are the most efficient option because of the considerable number of flip-flops available per device [160]. These flip-flops can be easily connected into data registers suitable for a pipelined datapath. Multiplexing resources and thereby breaking the pipeline is less efficient for an FPGA-based design.

In Table 4.9 the complete FPGA neural model is compared with previous projects. The design presented here compares favourably with previous implementations although consideration has to be made for the advancements in FPGA-technology between each implementation. There is also a trade-off to be made between using configurable logic or arithmetic resources. Cassidy et al. [122] were limited in their device to only 32 arithmetic units, so some arithmetic operations would have to have been completed using the configurable logic resources, and hence their slice utilization is higher. Whereas modern FPGAs provide thousands of arithmetic units, primarily to cater for the digital signal processing market.

Thomas et al. [159] have been the only group that have argued for floating-point implementation of the Izhikevich model upon FPGAs. Clearly their choice has had a significant effect upon the resource utilization.

Unlike the previous case study, an Izhikevich model is not limited by the DSP resources available. If the neural model was run in real-time mode each processing core could conceivably implement 500,000 virtual neurons whilst running at clock frequency of 500MHz. However, these neurons would require 20Mb of memory. Therefore, it is only possible to implement 3 processing cores per FPGA before the maximum on-chip memory available is exceeded. These 3 processing cores would require only 15 of the available 3600 DSP units. This is also of course neglecting any consideration for the resources required by the neural interconnectivity. The granularity investigation by

Cassidy et al. proposed 8,000 virtual neurons per processing core, however they consider delay as a performance parameter, whereas for a neuroprosthesis real biological time is essential.

In terms of granularity of processing cores, when implemented upon an FPGA it is desirable to consider making the most appropriate use of the resources available. Modern FPGAs are supplied with a large amount of dedicated resources as well as configurable logic blocks. These dedicated resources are more efficient in terms of area, and thereby provide a greater density of memory capacity. When implementing a processing core the design should target the dedicated block RAM available. These RAMs are defined as 36kb on a Xilinx Virtex-7, allowing for 900 virtual neurons per neuron processing core.

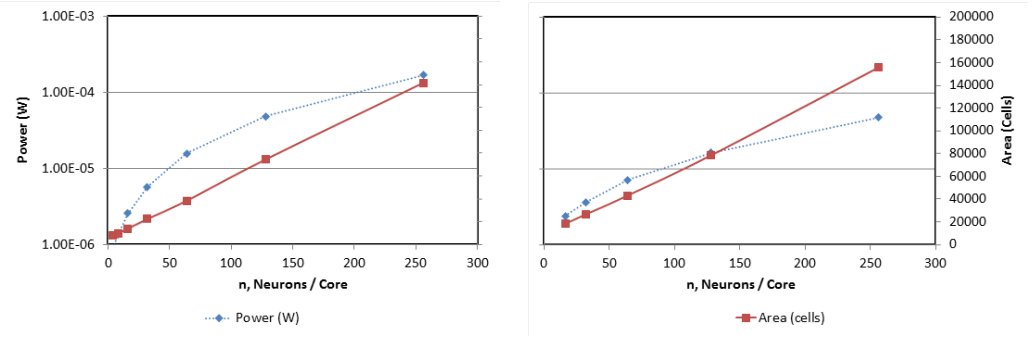
4.5.3.2 VLSI Implementation

Both designs have been synthesized using the Synopsys Design Vision software suite to compare area and the affect of a differing granularity. Also, an estimate of power consumption is provided by combining the synthesized design with a behavioural level simulation using Synopsys PrimeTime. Similar to Imam et al. [121] a 65nm CMOS technology operating at 1V is targeted. Although the design hasn't been translated to a full layout the synthesis results do provide sufficient levels of accuracy to compare implementations.

Each design is analysed at a varying level of granularity, from 4 virtual neurons per processing core up to 256 processing cores. Real-time operation is targeted, requiring each neuron to be updated once per ms. This leads to operating frequencies in the region between 13kHz and 3.3MHz depending upon the design option and the granularity.

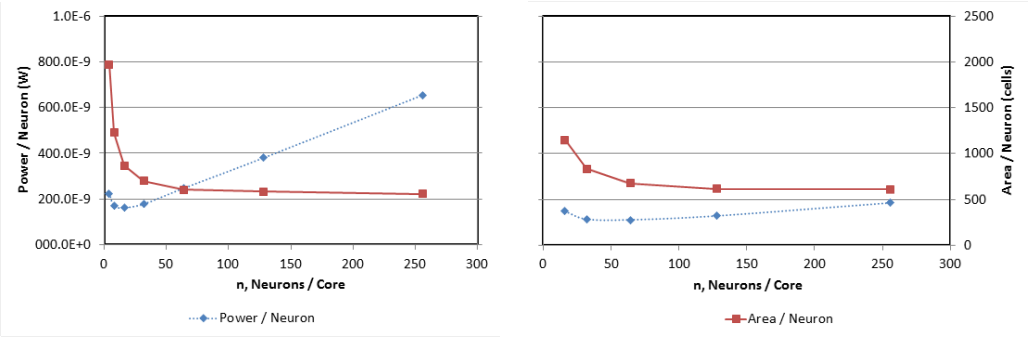
In Figure 4.20, a comparison of the area and power consumption of the two design options is illustrated. The minimum resource approach is shown to be between 1.1x and 3.5x more energy intensive. In the design section on page 69 the minimum resource approach was predicted to be 2.7x more energy intensive.

As expected the area of both approaches increases linearly with increasing granularity. This is due to the requirement to store extra neuron parameters in memory. The offset in the area is caused by the area consumed by the datapath itself. With a high granularity the area of the system is clearly dominated by the memory. This is a fact neglected by



(A) Minimum resource (B) Maximum resource

FIGURE 4.20: Area and power usage with a varying granularity for digital Izhikevich neuron models synthesized using 65nm CMOS.



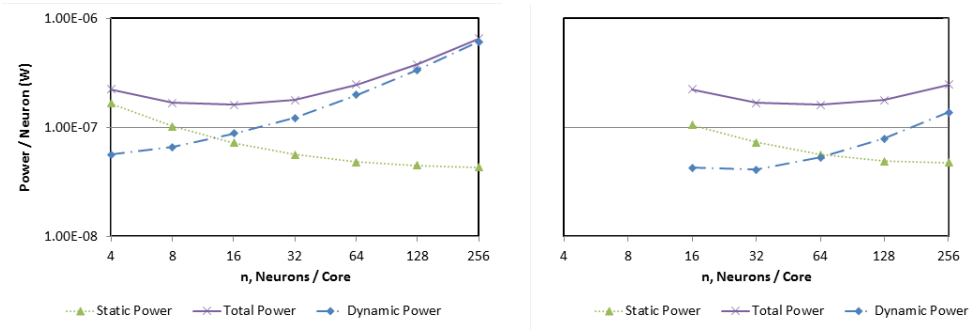
(A) Minimum resource (B) Maximum resource

FIGURE 4.21: Area and power usage per neuron.

Imam et al. [121] who give area results when considering only a single virtual neuron. In Figure 4.21 the area per neuron for each approach is shown. Imam et al. state that their area per neuron is equivalent to 30,000 cells [121], whereas by multiplexing multiple virtual neurons upon a single processing core the area per neuron can be reduced to between 500 and 1000 cells.

Figure 4.21 also shows that a minimum resource approach does offer a better area per neuron ratio than a maximum resource approach as predicted. However, its energy consumption per neuron is far greater at coarse granularities. This is caused by the much greater operating frequency required by a minimum resource approach. For example, a processing core with 256 virtual neurons will require an operating frequency of 3.3MHz for the minimum approach, as opposed to only 260kHz for the maximum approach.

For both approaches there is an optimal granularity to reduce the energy consumption per neuron. This is between 16 and 32 virtual neurons per neuron processing core. This minimal point is caused by the static power per neuron reducing with increasing



(A) Minimum resource (B) Maximum resource

FIGURE 4.22: Components of power consumption.

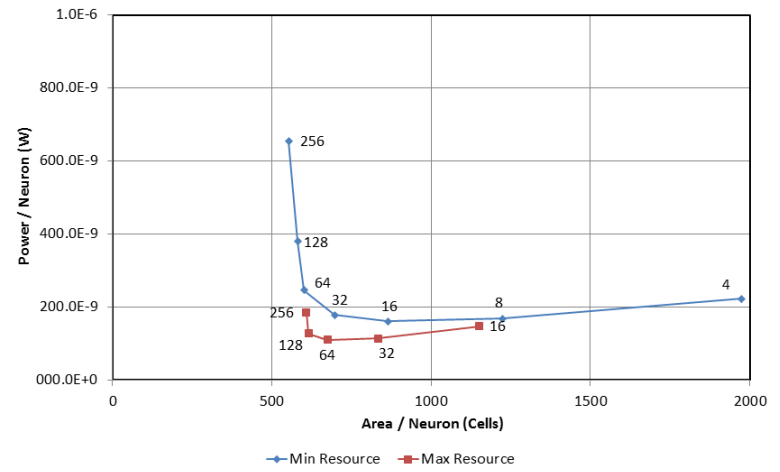


FIGURE 4.23: Power versus area consumption for a differing neural granularity.

granularity, whilst the dynamic power increases with increasing granularity, as illustrated by Figure 4.22.

In Figure 4.23 the area energy product of the two different design approaches is shown. For both designs there is a clear optimal granularity for a processing core. At a finer granularity the datapath area resources are not efficiently shared between the virtual neurons. Whereas at a coarse granularity the energy consumption increases without any benefit in terms of area gains. This is because increasing the number of virtual neurons per processing core requires an increase in operating frequency, but the area per neuron can only reduce asymptotically to the area consumed by a neuron's parameters. A coarser granularity has a greater detrimental impact upon the minimum resource approach due to the higher increases in operating frequency required.

4.5.4 Discussion

Extracting from Figure 4.23 it is reasonable to predict that the digital silicon Izhikevich neuron should consume approximately $100nW$ per neuron and occupy approximately 600 cells, which is equivalent to $600\mu m^2$.

When the area consumed by the memory components is removed the maximum resource approach datapath consumed approximately 11000 μm^2, whereas the minimum resource approach consumed only $4900\mu m^2$, a 2.25x difference. In Table 4.6 a difference of 3.7x was predicted. The difference between the estimation and the implementation is likely to be caused by the estimated value neglecting to consider the area of the multiplexors required for the arithmetic units.

The datapath illustrated by [121] consumed $30{,}000\mu m^2$. However, they have neglected to compress the equations as suggested by [108] and [122] and they use an asynchronous design. Their energy per neuron update is similar to what is estimated here.

The energy per neuron of $100nW$ compares favourably with the $100\mu W$ calculated by Sharp et al. [109] for implementing a neural model upon the ARM-core based SpiNNaker system. The equivalent area per neuron for SpiNNaker is $1000\mu m^2$, although this does not consider the area consumed by storing the neuron parameters and they choose to use a coarse granularity of over 1000 neurons per core to improve this area per neuron ratio.

The ARM-core utilized occupies an area of approximately $1mm^2$, which is much larger than the processing core described here. As such, their static power consumption will be greater; this perhaps can be offset by the coarse granularity they have used. They also want to implement as many neurons as possible on one processor to reduce the overall size of the implemented super computer - thereby reducing the associated energy overheads.

In comparison an analogue neuromorphic style neuron occupies approximately $4000\mu m^2$ per neuron [112] and consumes only 10nW of power [126]. Although this energy consumption is a factor of 10 smaller than what is achieved here it is dependent upon the number of spikes produced by the neuron. Also, to reduce circuit size analogue neurons tend to run in accelerated time. A real-time neuromorphic neuron is likely to be much

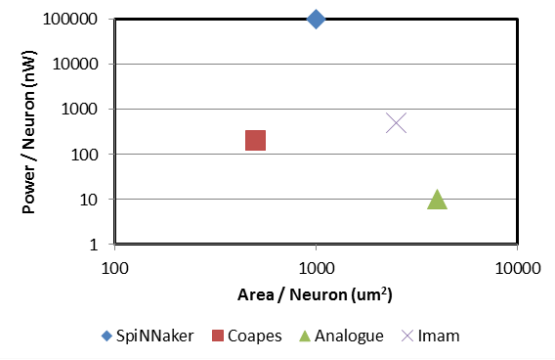


FIGURE 4.24: Comparison of design approaches to single neuron processing.

larger (due to the increase in the size of capacitors required [117]) and consume more energy [161].

A comparison of the different design approaches is available in Figure 4.24.

Currently, the design utilizes D-type flip-flop (DFF) style memory. Alternatively, static random access memory (SRAM) could be used. SRAM offers a higher density of memory and may offer a reduced power consumption in comparison with DFF. It may be of benefit beyond the granularity that what is currently considered - up to 256 virtual neurons per processing core. However, it has been shown that as the number of virtual neurons increases the dynamic power increases linearly due to the higher operating frequency required. It is unlikely that this increase will be overcome through the use of SRAM.

At the optimal granularity illustrated in Figure 4.22 the dynamic and static power consumption is approximately equal. Due to the low-operating frequency of the datapath at this granularity power-gating could be considered [162]. This will involve switching the circuits at a higher frequency, then when the operations are completed disabling the power supply. Using this technique, the dynamic energy will stay approximately the same but the static power will be reduced. However, at the optimal granularity 75% of the transistors of the neuron processing core are utilized for the neuron parameter memory - this memory cannot be disabled as the memory state is volatile. As such, the static power could be reduced by only up to 25% and the overall power consumption by 12.5%.

Operating circuits at lower voltages allows for reduced energy consumption, but increases the time it takes to charge gates and therefore the delay of the circuit. Since the neuron

processing core design is a low throughput application with a low operating frequency delay is non-critical and therefore it is a suitable candidate for either dynamic-voltage scaling (DVS) or sub-threshold implementation. By operating the processing core at a half of the usual supply voltage of 1V the dynamic power consumption will decrease by 4x due to the quadratic relationship between dynamic power and voltage. Also, the popular static power model generated by Butts et al. [163] suggests that static power reduces linearly with voltage supply. Therefore, by operating the processing core at 0.5V the energy per neuron can be reduced to only 37.5nJ, only 3-4x greater than an analogue neuromorphic implementation.

As previously mentioned, with an optimal granularity each neuron will consume approximately $700\mu m^2$. When scaling this up to meet the specification defined in Table 2.1 of 10k-100k neurons the equivalent silicon area consumed equals between 7-70mm^2. Also, when consuming 100nJ per neuron the chip power consumption will be between 1mW and 10mW. If the potential energy savings from power-gating and dynamic-voltage scaling are both considered this power consumption could be reduced to between 0.3mW and 3mW.

4.6 Chapter Discussion

The implementation of neuron models using digital techniques has been shown to be feasible, area and power efficient, scalable with technological advancement and comparable with alternative options, particularly the popular analogue neuromorphic design methodologies.

Further, previous digital design implementations have primarily considered leaky integrate and fire systems due to their simplicity in terms of hardware resources. The neuron model designs illustrated here, of both Hodgkin-Huxley and Izhikevich variants, are able to produce more biologically realistic spiking patterns, providing a greater degree of accuracy in any large-scale network simulations. It has been shown in this chapter that both Hodgkin-Huxley and Izhikevich neurons can be implemented efficiently if judicious design choices are made, therefore, they should be preferable over the traditionally leaky integrate and fire design.

With the increasing number of transistors available upon single devices a large number of neuron models may be implemented. For instance, on a state-of-the-art FPGA it has been shown how 100,000 individual Hodgkin-Huxley neurons may be implemented. Izhikevich [136] described how a simplified model illustrating the same behaviour can be 100x more computationally efficient than a Hodgkin-Huxley. However, in this work it has been found that due to memory constraints only a 15x performance improvement can be made, with 1.5 million Izhikevich neurons per FPGA.

Choosing the correct granularity of a neuron processing core has been shown to have a significant effect upon the design response, in terms of area, energy and performance, both for an FPGA and for ASIC devices. Selecting the wrong granularity may render a design proposal infeasible in the early stages of development if energy and area is constrained.

The SpiNNaker [4] project does utilize a differing granularity to what has been proposed within section 4.5.3.2, but their design objectives differ. For instance, they wish to model large-scale brain regions using a supercomputer approach, as opposed to the single chip implementation considered for a neuroprosthesis. Also, due to the scale of the project in terms of costs and resources they desire the system to be applicable to multiple domains beyond simulating brain regions, such as robotics [164]. For this they require a flexible design platform, which a dedicated Izhikevich or Hodgkin-Huxley datapath would not provide.

In this section, a design for an Izhikevich neuron model with the same performance as similar analogue implementations and with similar area and energy requirements has been described and demonstrated. Energy efficiency has long been the justification behind the development of analogue designs but, the comparative energy performance that has been illustrated alongside factors including: extended design-time, lack of programmability, the requirement for digital communication and model variability; negate this justification.

As well as demonstrating the optimal design of a digital silicon neuron this section has provided a model to determine the optimal granularity of processing element for spiking neural networks. This model is used in the following chapters whereby the neural-network-on-chip is considered, as described in section 5.6.

Chapter 5

Network Methodology

"You know my method. It is founded upon the observation of trifles"

The Bascombe Valley Mystery, Sherlock Holmes

Sir Arthur Conan Doyle

The previous chapter has introduced optimal designs for silicon neuron models. The computation within the brain is encoded within both the generation of spikes and the transmission of spikes between these neurons. Therefore, an efficient silicon communication infrastructure must be developed in order to correctly replicate the transmission of spikes.

This design concept is illustrated by Figure 5.1. Multiple processing cores, each capable of modelling multiple neurons are implemented, but a common communication infrastructure must be provided to allow for information about spike events to be passed between the cores. The focus of this chapter is to study this communication.

The requirements for the communication platform are first of all described, before comparisons between the most common design approaches are provided. Next, a detailed introduction to a neural-network-on-chip system is given for which a methodology is developed to study the energy and area design parameters for any neural network topology. In the following chapters, this methodology is used to detail two different designs for neuroprosthetic systems.

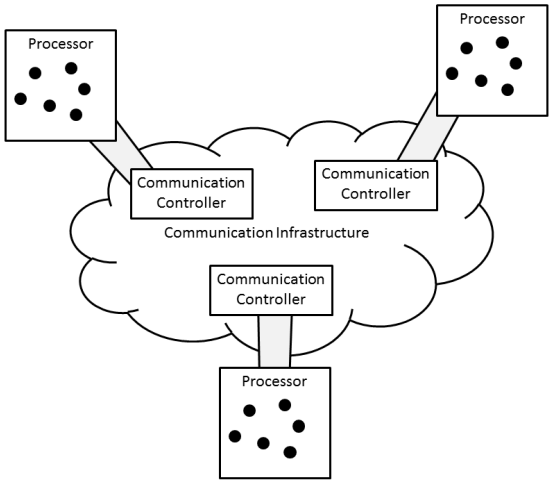


FIGURE 5.1: Communication concept requirement. Multiple processing cores, each capable of modelling multiple neurons are implemented. A communication infrastructure must be provided to allow for information to be transmitted between each of these processing cores.

5.1 Requirements

The silicon network is responsible for communicating all spikes from source neurons to the destination neurons. To provide a system with maximum flexibility any neuron should be able to be connected to any other neuron and the connectivity should be flexible, both at initialization and at run-time, to allow for learning and plasticity.

Typically, for biological networks there may be over 1000 destination neurons [75]. It is infeasible to directly connect all of these neurons using point-to-point wiring. A typical logic gate would only have 4 output connections before extra buffering is required [75]. Also, even if the technology allowed point-to-point connectivity reconfigurability of the network would be a challenge.

However, silicon systems operate at much higher speeds than their biological equivalents. This allows for multiple biological axons to be multiplexed onto a subset of silicon wires. A controlling system is required to manage the multiplexing operation. The controlling system must allow for:

- Fair arbitration - equal access to the shared resources must be provided. Unequal access could result in the system having parasitic effects upon the communication of spikes, which in turn will alter the result of the computation of the neural network.

- Efficient routing - spike events must be efficiently directed along the silicon wires to the correct destinations.

To simplify the communication process we can assume that the spike produced by a neuron is an all or nothing digital pulse. The source neuron in effect broadcasts this pulse along its axon through which the destination neurons are listening through synapses and their dendritic tree. The source neuron knows nothing of the state of the pulse once it has been broadcast. The destination neurons must be aware of the source of the pulse in order to update synaptic connectivity as required.

The typical delay of a spike from source to destination, known as the axonal delay is in the order of milliseconds [165]. The controlling system must transmit the pulses within this timeframe and not introduce variation within the delay of transmission so as to not affect the computation as it would be in biology. In effect, the silicon systems should provide a transparent platform for the spiking neural network.

5.2 Design Options

5.2.1 Previous Work

Communication between neurons is the bottleneck in most neural network simulations implemented in software, on GPUs or upon supercomputers. To overcome this many research groups have investigated developing specialist platforms to optimize the process.

For example, the previously mentioned SpiNNaker project [4] uses a packet switching network. Each spike pulse is represented as a packet containing the identifier of the source neuron. This packet is routed between processing cores using a multicast technique. The 18 cores upon each chip are connected using a NoC and each chip is connected with the 6 nearest neighbours in a torus topology.

In contrast to SpiNNaker's multicast approach, EMBRACE [132] use an alternative unicast packet routing methodology. For each source-destination connection pair a separate packet is generated. Unfortunately, they don't appear to consider the high level of connectivity between neurons and the effect of this upon the traffic bandwidth requirements. Unicasting is feasible if each neuron is only connected to several other neurons,

but with 1000 post-synaptic connections the traffic will increase significantly. However, they do propose a packet compression technique to reduce the traffic, although this is only available when a group of neurons all target the same destination neuron [132].

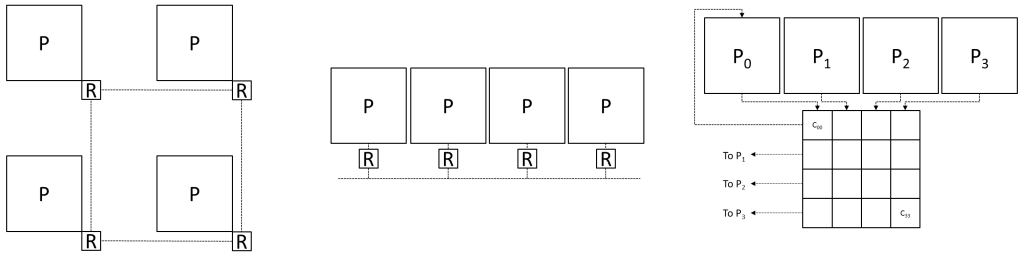
Similarly, the Bluehive project [125][130] passes messages containing source and destination information between multiple FPGAs in their FPGA-board level platform.

Alternatively, the Neurogrid project [166] uses a broadcast routing strategy, whereby all events are passed along a 1-D chain of processing chips. These broadcast packets can be filtered as desired. They suggest that when connections are dense that broadcasting is more effective than sending multiple copies of packets.

Emery et al. [119] combined the packet switched approaches of other groups with a circuit switching design. They suggest that many neuron-to-neuron connections are local and only a few connections travel long-distances. As such, in their system local connections are connected using a circuit switching approach whereby an individual wire is configured to connect a "source" neuron with a "sink"' neuron. Long-range connections are implemented using a packet switching based-approach. Emery et al. consider using only a granularity of 1 virtual neuron per neuron processing core, which has been shown in Chapter 4 to be non-optimal. Adding more virtual neurons will require the local circuit switching protocols to become increasingly more complex and non-tractable.

Similar to the circuit switching of Emery et al. the Neurocore chip described in [120] uses a crossbar circuit switching technology. Within this design a matrix of connections is generated using SRAM cells, whereby a 0 output from the cell represents no connectivity, and a 1 represents a connection. For full-connectivity this requries a quadratic number of cells and as such they are only able to demonstrate a 256 neuron array upon a single chip. The HiCANN project [117] uses a similar matrix array for synaptic connections but with a packet-switched approach between multiple matrix arrays.

Ambroise et al. [158] propose a design with a similar objective to ours, brain-machine interfaces. They use an SRAM model for neuron-to-neuron connectivity. Each row in the SRAM contains a list of destination for a particular source neuron. As such, their design is memory bound and they are limited to fewer than 117 neurons per FPGA with only limited network connectivity.



(A) Mesh NoC (B) Bus (C) Matrix

FIGURE 5.2: Different neuron-to-neuron connectivity options.

In nearly all the systems mentioned above, timing within the neural network is modelled relative to the system time. For example, a spike produced by a neuron is judged to have arrived at its destination neuron when the message has transmitted through the silicon system. This creates an asynchronous platform similar to biology and negates the need for timing to be tracked.

A summary of previous work is provided in Table 5.1.

5.2.2 Comparison of Options

To compare the interconnect methodologies previously studied the most popular techniques, which are illustrated in Figure 5.2, are evaluated assuming the neural network model consists of m processors each implementing n neurons, giving a total neural network size of s. Following [169] we can estimate the scalability of the area and power consumption for the three different methodologies.

Network-on-Chip A simple NoC system comprises m processors each connected to a routing element to form m modules. The modules are arranged in a $\sqrt{(m)} * \sqrt{(m)}$ mesh[2] whereby each module is connected to its four nearest neighbours, see Figure 5.2a. When a spike is produced a packet originates at the source module and is directed through the mesh to the correct destinations by the routing elements. The links between each module are of a constant size. To add more capacity to the system extra modules are attached to the mesh.

[2]Other arrangements are available

TABLE 5.1: Summary of neuron-to-neuron communication protocols.

Project	Project Aims	Neuron Model	Granularity[1]	Hardware Platform	Neuron-to-Neuron Communication Choice	Neuron Core Connection Topology	No. of Neurons	Learning	Reference
SpiNNaker	Simulate networks of a billion neurons in real-time	Digital, LIF, Izhikevich	1000	ARM-based supercomputer with specialized communication architecture	Multicast packet routing - AER source address based	Low-level: NoC, High-level: Torus	10^9	Yes	[4]
EMBRACE	Spiking neural networks to be used in embedded neural network applications including ...	Analog LIF	10	65nm CMOS	Packet-based containg source and destination information	Hierarchical NoC	10^4	Not in NoC literature	[167] [131]
Bluehive	Simulate networks of neurons in real-time (an fpga-based comparison to SpiNNaker)	Digital Izhikevich	16000	Multi-FPGA	Novel synaptic pointer passing algorithm	2-D grid	10^5	No	[130]
Neurogrid	Simulating multiple cortical areas in real-time	Analog	1000s	250nm CMOS	AER Broadcast	1-D Grid	10^4	On a separate system	[166]
Emery	Hardware platform for spiking neural netwokrs	Digital LIF	1	130nm CMOS	Hybrid between packet and circuit switching	2-D Grid	-	Yes	[119]
Neurocore	Brain-like cognitive computers	Digital	1	45nm CMOS	Crossbar	Matrix crossbar	10^2	Yes	[120]
HICANN/FACETS	Platform for studying large-brain regions in faster than real-time	Digital	512	180nm CMOS - wafer scale	Hybrid multi-level	Multi-level	10^4	-	[168]
Ambroise	Brain-machine interface	Digital Izh	117	FPGA, Virtex-4	SRAM	-	10^2	No	[158]

[1] Defined as the number of neurons connected to a single communication controller

Bus Within the bus methodology, each processor is attached to the same interconnect, see Figure 5.2b. When a spike is produced the processor requests control of the bus and transmits the message to all other processors. The other processors are required to listen to the incoming message and filter out irrelevant messages as required. The bus may be arranged in a different topology to reduce area and energy overheads, such as within [133]. Alternatively, as in [169] the bus may be segmented. In this scenario messages are broadcast along a single segment, before transmission to the next segment in the chain. A segmented bus is in effect a 1-dimensional NoC.

Matrix Finally, in the matrix architecture, when a spike is produced by a processor, a corresponding row within the matrix is selected, see Figure 5.2c. This row contains details of the connectivity. Each connected processor then receives a synaptic update. This is the methodology utilized by [120] and [158]. The matrix architecture can also be considered to be a cross-bar switch.

Circuit switching techniques are not considered as these are deemed to be infeasible for the scale of biological neural networks that are of interest. The introduction of concentrators/expanders and intelligent circuit switches requires the use of defined look-up routing tables which are shown to be hugely dominant in the consumption of resources within a neural network system. Therefore, such an approach is likely to provide very similar results as that shown for a NoC.

Area Estimations The area of processing within each methodology will be constant, however the area associated with the communication methodology will vary. The area of each methodology can be estimated as the area required by the interconnect topology. For a NoC, as the number of modules grows the number of links between modules grows at the same rate. Similarly, for the bus technique doubling the number of processors requires doubling the length of the bus. However, for a matrix style approach the area grows quadratically as for each new processor a new row and a new column is required.

Power Estimations The power can be estimated by assuming that it is a product of the area of the interconnect and the operating frequency of the interconnect [169]. For the NoC, the operating frequency of each link will remain constant as the link size is constant. Therefore the power will only grow with the area increase. For the bus, the

TABLE 5.2: Cost functions for three different neuron-to-neuron connectivity options.

	Area	Power
NoC	$O(s)$	$O(s)$
Bus	$O(s)$	$O(s^2)$
Matrix	$O(s^2)$	$O(s^3)$

operating frequency will be required to increase to overcome increased traffic pressure upon the bus. This will result in quadratic increases in power consumption. Similarly, for the matrix approach the operating frequency will have to increase adding to the power consumption. These costs associated with the different methodologies are summarized in Table 5.2.

Summary A network-on-chip based approach is the most suitable to implement the neuron-to-neuron connectivity due to its highly parallel and concurrent nature. In fact, all previous implementations have utilized a network with packet-switching techniques at some-level. Only in small-scale systems have alternative approaches been deemed to be viable [120][158].

In this study the primary interest is in the mesh and torus NoC topologies. Previously, hierarchical topologies, such as binary tree, have been shown to be more inefficient [133] when traffic patterns are not suited. With the reconfigurable nature of neural networks, both at initialization and at run-time, it is difficult to guarantee optimal traffic patterns to enable efficient use of hierarchical structures.

5.3 Network-on-Chip

A NoC consists of multiple processing cores and a communication infrastructure to support passing of information between these cores. In a neural-NoC this information mainly consists of the spikes produced by the neurons modelled within the processors.

Within each processor core multiple neurons are modelled. When a neuron's voltage membrane exceeds a threshold a circuit is activated. This circuit then produces a packet representing the spike which is to be passed along the virtual axon to all of the required destinations. This virtual axon consists of a series of interconnected routing elements.

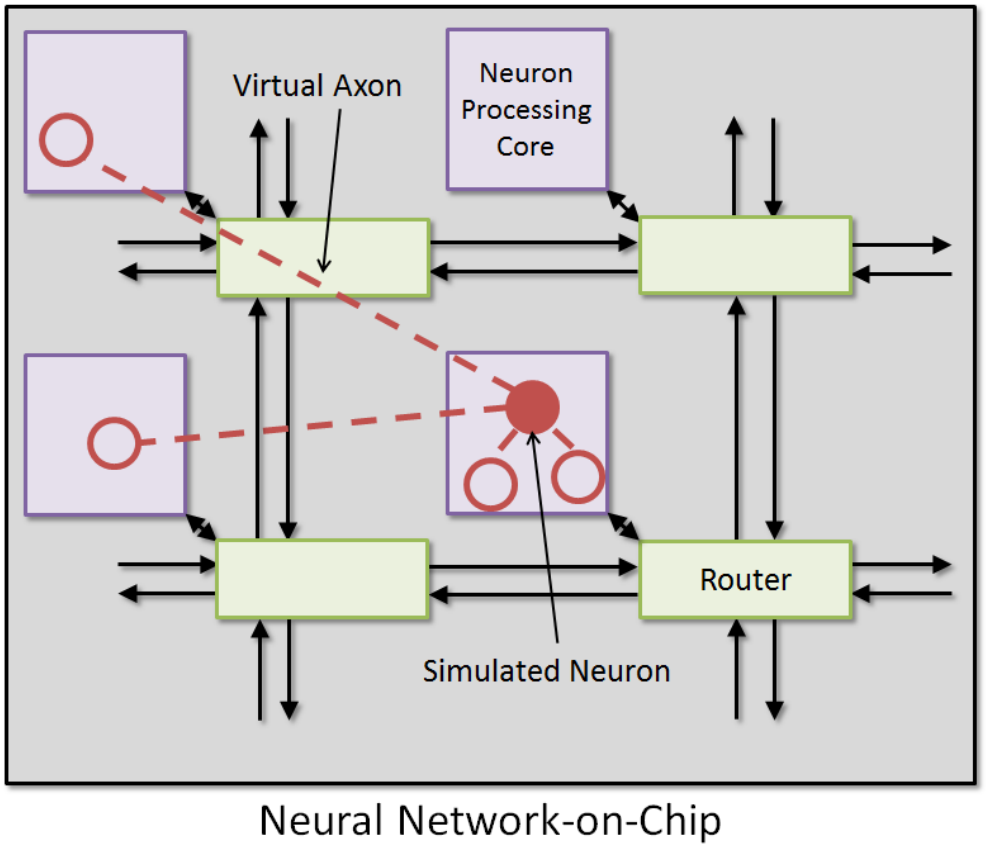


FIGURE 5.3: Neural-Network-on-Chip Structure. A neural network, containing s neurons is simulated upon a network-on-chip containing m processors arranged in a k by k 2-dimensional grid. On each processing core n neurons are implemented. These neurons may communicate with their connected neurons upon other processing cores through the network-on-chip infrastructure and transmission of packets.

When a router receives a packet it must decide whether to transmit the packet to its own processor, and/or forward the packet to its neighbouring routing elements.

In the topology investigated within this thesis, most routers are connected to four neighbouring routers. Routers located at the edge of the mesh are only connected to routers within the mesh, i.e. there is no wraparound connections to make the system a torus.

This topology is shown within Figure 5.3.

There are many different strategies to determine how to route the packets through the network and the choice of routing strategy has a significant affect upon the performance of the system as well as the area and energy overheads. In the following sections the three most popular techniques and their system requirements are described.

To provide an initial prediction of the performance of each routing strategy an estimate of the latency, throughput, power and area requirements can be defined by extending

the analysis techniques of Bolotin et al [169]. In this thesis, these analysis techniques are extended by fully considering different routing strategies and also by assessing the affect of a varying granularity.

To accomplish this analysis some assumptions/definitions are required:

- A neural network model contains s neurons implemented across m processing cores arranged in a 2-dimensional square mesh,with each processing core containing n neurons. There are k rows/columns within a mesh of m cores
- The neurons are randomly allocated to the neuron processor cores. Although, this is not strictly true in many situations it does create an upper bound in the performance estimations. It also gives rise to the relationship listed in (5.1), which is a measure of the mean Manhattan[3] distance between two neurons in a mesh [133], where m is the number of processing cores.

$$c = \frac{2}{3}\sqrt{m} = \frac{2}{3}k \tag{5.1}$$

- Secondly, that on average each source neuron has $\bar{N_d}$ destination neurons.
- Each neuron produces a spike at a rate of $\bar{\delta_n}$ spikes per second.
- A mesh contains L links

$$L = 2k(k-1) \tag{5.2}$$

- Finally, the area of each processing core is directly proportional to the number of virtual neurons. See (5.3), where A_{core} is the area of a processing core and a is the mean area per neuron. This area per core can be used to determine the mean length of NoC physical wires, or links, between routers, as in (5.4)

$$A_{core} = an = \frac{as}{m} = \frac{as}{k^2} \tag{5.3}$$

$$\bar{l} = \sqrt{an} = \sqrt{\frac{as}{k^2}} \tag{5.4}$$

[3]The Manhattan distance, or the taxicab distance, is the distance between two points assuming you can only travel in a straight line along either the x or y dimension [170]

These assumptions allow an estimate of the power, area, throughput and latency to be determined:

- A contributing factor towards power in a NoC is within the switching of the links between routers in order to transmit packets. This power can be estimated using (5.5), whereby f_n is the mean frequency of the link and U_{arch} is the mean utilization of each link as defined by (5.6). The utilization of each link is determined by the chosen routing strategy, as such, in the following sections an estimate is derived for each strategy. However, it is shown in section 5.5 and section 6.2 how this commonly used derivation of power in a NoC neglects some key components of a neural-NoC.

$$P = CV^2 f_n U_{arch} \tag{5.5}$$

$$U_{arch} = \frac{TotalBandwidth}{No.ofLinks} = \frac{BW}{L} \tag{5.6}$$

- To determine the power consumption a relationship must be formed for the overall capacitance of the NoC. Capacitance is related to the capacitance per unit length, C_0, and the total length of all the links in the network, l_{mesh}. Equation (5.7), which shows the total length of all the links, is derived by multiplying the number of links by the width ($\bar{w}$) and length of each link. This gives rise to the total capacitance of mesh neural-NoC (5.8).

$$l_{mesh} = 2k(k-1)\bar{w}\sqrt{\frac{as}{k^2}} = 2\sqrt{as}\bar{w}(k-1) \tag{5.7}$$

$$C_{mesh} = C_0 l_{mesh} = C_0 2\sqrt{as}\bar{w}(k-1) \tag{5.8}$$

- Area cost of the NoC is defined as the pitch of each wire, W_p, which is constant for each technology node, multiplied by the total length of all links. The area of the NoC interconnect is not dependent upon the chosen routing strategy. It is fully described in (5.9). As can be seen the area grows linearly with the size of the mesh and with the square root of the number of neurons in the network. Again, in

section 5.5 and section 6.2 it is shown how this popular estimation for area within a NoC is not a good representation for a neural-NoC.

$$A_{mesh} = 2W_p\bar{w}\sqrt{a}(k-1)\sqrt{s} \tag{5.9}$$

- The delay of each link is also constant and not dependent upon the routing strategy. The link delay is the time it takes for a signal to propagate across the link. This is determined by (5.10) where $R_\square$ and C_O are technology dependent. The corresponding link delay for a neural-NoC with mesh topology is given in (5.11). The link delay increases linearly with the number of neurons but has an inverse relationship with the number of processors.

$$T_{cycle} = R_\square C_O \bar{l}^2 \tag{5.10}$$

$$T_{mesh} = \frac{R_\square C_O as}{k^2} = \frac{R_\square C_O as}{m} \tag{5.11}$$

- The total latency of a topology is the link delay multiplied by the *zero-load hops* [171]. The zero-load hops is the total Manhatten distance travelled by each packet to reach its destination assuming no contention within any of the routers along the packet's path. The zero-load hops is represented by H_0 and the latency by T_0. The zero-load hops is dependent upon the chosen routing strategy.

- The throughput, represented by $f_{s_{max}}$, is determined as the bandwidth that will saturate the bottleneck channel within the NoC [171]. As the bandwidth is related to the chosen routing strategy the maximum throughput that can be achieved also changes with routing strategy.

5.3.1 Unicast

Unicast routing involves the direct transmission of an individual packet from the source neuron to each destination neuron. The packet can contain both the source identifier and the destination identifier. The routers are able to use the destination identifier to determine how to direct the packet through the network - if the identifiers are representative of a neuron's location. This concept is illustrated in Figure 5.4, whereby a source neuron is connected to three other neurons. Therefore three unique packets are

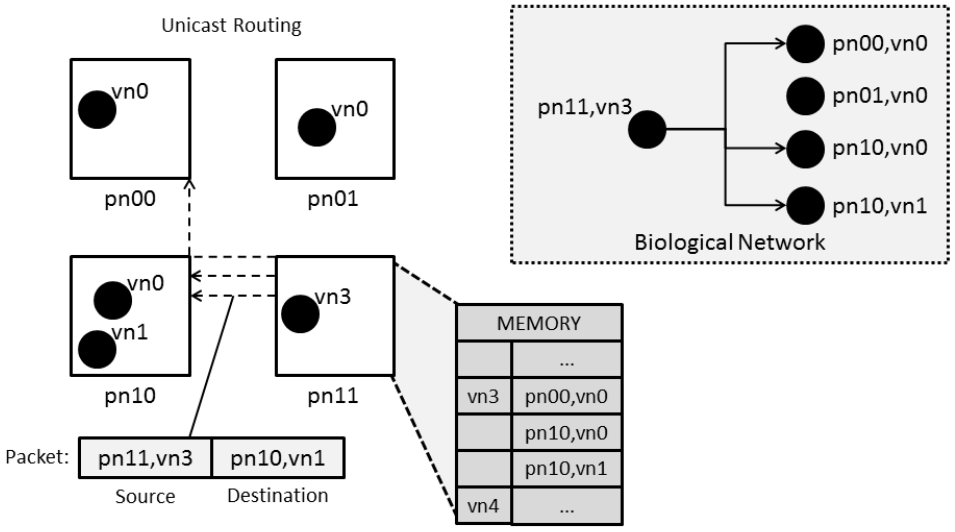


FIGURE 5.4: Unicast routing strategy. The neuron $pn11, vn3$ is connected to three other neurons, as defined in the memory block of $pn11$. When it produces a spike an individual unique packet is produced for each source-destination pair. These packets are routed through the mesh to the correct destinations.

transmitted and routed to the correct processing core. When the packets arrive at the core they are added to the target virtual neuron's synaptic input.

As shown in Figure 5.4, the connectivity of the neural network is stored at the source processing core. Each neuron must store a pointer to a location in memory containing a list of connected destination neurons. This pointer could be stored alongside the neuron's parameters[4] allowing for efficient access. Each processing core is required to store a list of lists for all of its virtual neurons. The length of this list is dependent upon the granularity of the neuron processing core and the mean number of connections, $\bar{N}_d$. However, the total length of all the lists in the network should remain constant regardless of the granularity.

As the routing information is contained within the packets themselves the routers can be implemented fairly easily and efficiently. Therefore, unicast is beneficial for its simplicity in many scenarios, but large-scale neural networks can become inefficient because of the significant number of packets generated due for each spike event. For example, if $\bar{N}_d$ is equal to 1000, then 1000 packets unique packets are generated for each spike. This leads to a high bandwidth and bursty traffic.

[4]For example, the v, u, a, b, c and d parameters of an Izhikevich neuron as discussed in section 4.5

5.3.1.1 Theory

To determine the estimated power consumption for a unicast mesh neural-NoC equations (5.5) and (5.8) can be used alongside (5.14), which describes the utilization of the interconnect links using a unicast routing mechanism. The utilization is the total number of packets generated, T_p, multiplied by the average distance each packet travels, D_{ave}, divided by the number of links in the system.

$$T_{p,uni} = s\bar{\delta}_n\bar{N}_d \tag{5.12}$$

$$D_{ave} = c = \frac{2}{3}k \tag{5.13}$$

$$U_{uni} = \frac{T_{p,uni}D_{ave}}{L} = \frac{s\bar{\delta}_n\bar{N}_d\frac{2}{3}k}{2k(k-1)} = \frac{s\bar{\delta}_n\bar{N}_d}{3(k-1)} \tag{5.14}$$

The power consumption is shown in (5.15). It can be seen that theoretically the power consumption does not depend upon the size of mesh that is used, as the factor k is removed. However, this is assuming the worst case scenario of random placement of neurons. It is shown in section 6.2.1 how optimal placement of neurons may reduce the power consumption and how the optimal placement is dependent upon the size of the mesh.

$$P_{uni} = CV^2f_nU_{uni} = C_02\sqrt{a}s\bar{w}(k-1)V^2f_n\frac{sf_s\bar{N}_d}{3(k-1)} = C_0\frac{2}{3}\sqrt{a}\bar{w}V^2f_ns\sqrt{s}f_s\bar{N}_d \tag{5.15}$$

The maximum achievable throughput is the bandwidth that will saturate the bottleneck channel. With unicast routing this is equivalent to the bisection bandwidth [171]. This bandwidth is the amount of packets generated that cross from one half of the network to the other half of the network. By assuming that the neurons are randomly allocated then 50% of packets will cross the bisection and the bisection consists of $(k-1)$ possible routes. This leads to the following relationship:

$$BW_{bisection,uni} = \frac{T_{p,uni}}{2}\frac{1}{k-1} = \frac{s\bar{\delta}_n\bar{N}_d}{2(k-1)} \tag{5.16}$$

To avoid saturating the bottleneck channel this relationship must not exceed one packet per clock cycle, leading to:

$$1 \geq \frac{BW_{bisection,uni}}{f_n} \tag{5.17}$$

From which, in turn it is possible to derive the maximum theoretical firing frequency for a unicast neural-NoC strategy (5.18). As shown the maximum rate of firing is inversely proportional to the number of neurons and the degree of connectivity, but proportional to the size of the mesh.

$$f_{s_{max,uni}} = \frac{2f_n(k-1)}{s\bar{N}_d} \tag{5.18}$$

The latency of the routing strategy is dependent upon the zero-load hops and the link delay as defined previously. The zero-load hops is the number of cycles it takes to transmit all of the packets produced by a single spike. If it is assumed that packets are transmitted sequentially from the source neuron at one packet per cycle then it is $\bar{N}_d$ cycles before the final packet can begin transmission. With randomized placement it will then take $\frac{2}{3}k$ cycles to transmit this packet to its destination. This leads to a zero-load hop of (5.19) and a latency of (5.20).

$$H_{0,uni} = \bar{N}_d + \frac{2}{3}k \tag{5.19}$$

$$T_{0,uni} = H_{0,uni}T_{mesh} = (\bar{N}_d + \frac{2}{3}k)R_{\square}C_O\frac{as}{k^2} = R_{\square}C_O as\bar{N}_d(\frac{1}{k^2} + \frac{2}{3k}) \tag{5.20}$$

5.3.1.2 Empirical Model

An empirical model has been developed to investigate the efficiency of unicast routing with a pre-defined neural network connectivity. The model determines:

- the bandwidth requirement - the number of packets that need to be transmitted and the distance of each packet assuming a defined mapping of neurons to processing cores. The mapping process is described in section 5.4.
- the memory overhead - an estimation of the total number of entries required in the memory of each processing element as illustrated in Figure 5.4.

This information is then combined with the area and energy models described in section 5.5 using the process flow outlined in Figure 5.16. From this an accurate estimate of the efficiency and constraints of a unicast approach for different network connectivities can be determined.

5.3.2 Multicast

To reduce bandwidth overheads, which is the main constraint in unicast, multicast routines have become more popular. With multicast a single packet originates at the source and is duplicated and directed as required by the routing elements within the network.

The packet therefore only contains the identifier of the source neuron. The routing elements are required to have some intelligence to decide in what direction to route each packet. Typically this intelligence is based around LUTs. The LUTs contain a list of identifiers. When a router receives a packet it checks its LUTs for the received packets identifier to determine what operation to complete. For each identifier the LUT stores a list of the operations required for that identifier alongside the identifier.

To reduce the number of identifiers in each list only a select few are stored. If no match is found in the list then the packet should pass through the router using a pre-defined default channel. [172]. Only if the packet should be passed to a non-default channel on its way to its destination should an identifier be present within the list. This process is illustrated in Figure 5.5.

The proposed multicast routing strategy has the following features:

- deterministic - the route of each packet through the network is known in advance as each router contains a pre-defined lookup table. A copy of a packet will travel

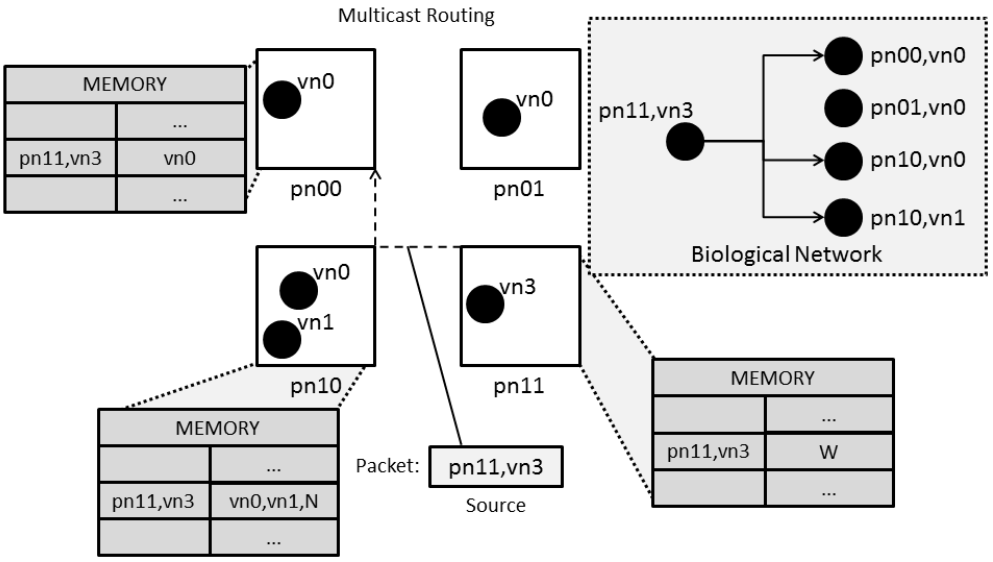


FIGURE 5.5: Multicast routing strategy. The neuron $pn11, vn3$ is connected to three other neurons. When it produces a spike a packet is passed to West to $pn10$. The memory of $pn10$ specifies that any packets received from $pn11, vn3$ should be passed to two of its own virtual neurons as well as along its North link towards $pn00$.

along an identical path. Deterministic routing in a 2-D mesh is known to be deadlock free.

- dimension-ordered - packets traverse through the mesh using pre-defined default channels. Only when a router contains an entry in its lookup table will the packet switch to a non-default channel. Again, dimension-order routing within a 2-D mesh guarantees that communication will be deadlock free [171]

- distributed - decisions are apportioned to each router. As the structure of the network is homogeneous each router is able to use the same routing algorithm.

The number of routing entries in memory can be reduced further by using the compression techniques proposed by Wu et al. [172] . Using this technique the packets are routed purely based upon their source processing core. This may take advantage of multiple virtual neurons upon the same processing core having similar connectivity and therefore a similar communication pattern.

5.3.2.1 Theoretical

Multicasting works most efficiently when traffic is clustered. For example, if a source neuron is connected to a group of other neurons all implemented upon a single neuron

processing core only a single packet is required to be sent. However, for the theoretical estimation we are assuming worst-case conditions, were no locality exists and therefore there is no traffic clustering. Also, in most large-scale neural networks $\bar{N}_d$ is likely to be much greater than the number of cores $m = k^2$. Therefore, with random placement it is likely that a single source neuron is connected to destination neurons upon every core within the network. In effect, with these assumptions multicasting degrades into a broadcasting strategy. As such, the worst-case theoretical performance of multicasting is described in section 5.3.3.1.

5.3.2.2 Empirical Model

By optimizing the placement of neurons upon processing cores the power and area requirements can be reduced. The optimization techniques are described in section 5.4. To evaluate the improvement of optimizing the placement of neurons an empirical model was developed. The model determined the bandwidth and memory requirements for a multicasting routing strategy given a defined neural network connectivity and a placement of the neurons upon the processing cores. The number of processing cores is also variable to investigate the effect of granularity. The model incorporated options to include the core compression techniques described above and the logical compression described in section 5.3.5.

The model returns a memory table for each router within the network along with the total distance travelled by all the packets for a given firing rate. For the example provided in Figure 5.5 the empirical model would return that the average memory table was 0.75 words deep and that the total distance travelled by all packets was 20 if $f_s = 10Hz$.

5.3.3 Broadcast

In broadcasting a single packet originates from the source neuron. The packet contains purely the source identifier. The packet is transmitted to all the processing cores within the system. These cores are required to filter through received packets and only pass on the correct packets to their virtual neurons.

Broadcasting is very simple to implement but in some situations may result in an excess amount of traffic as packets are transmitted to locations where they are not required.

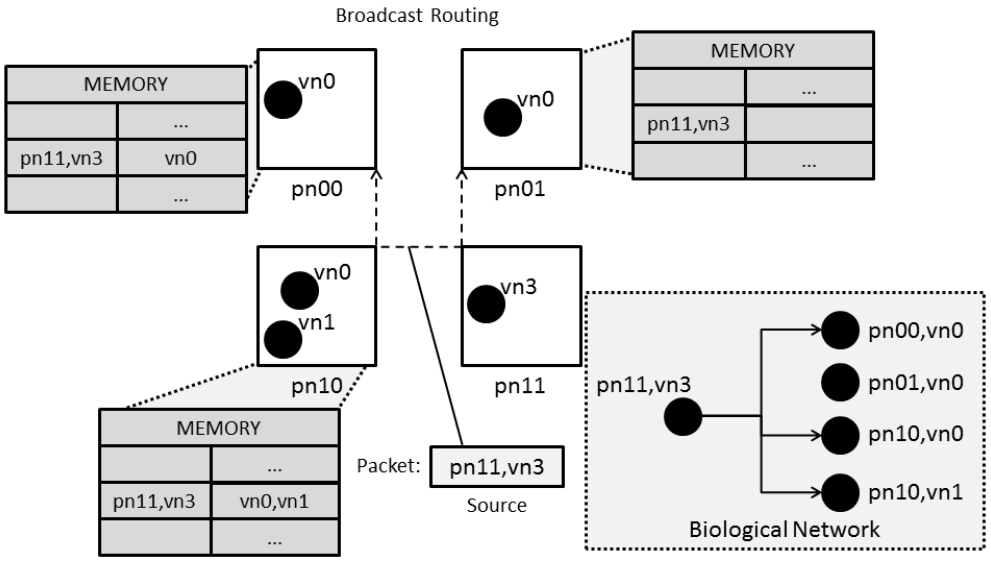


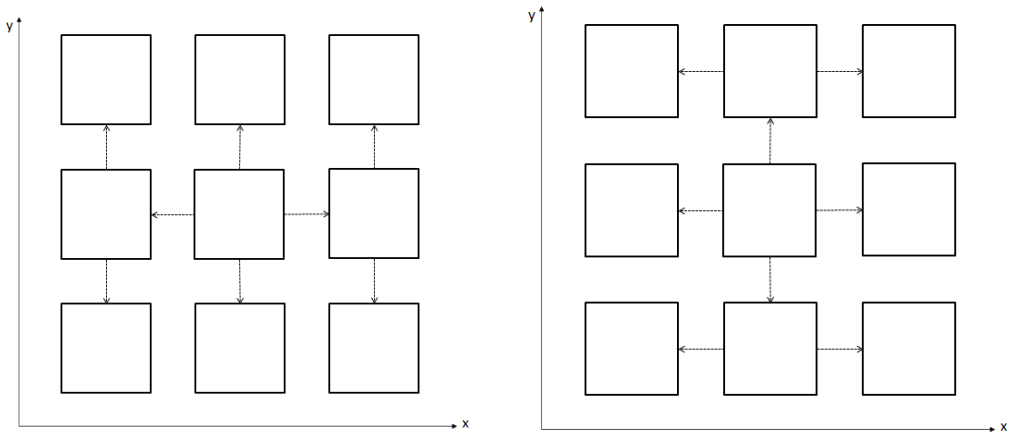
FIGURE 5.6: Broadcast routing strategy. The neuron $pn11, vn3$ is connected to three other neurons. When it produces a spike a packet is passed to all other processing elements. Each processing element must filter received packets.

The choice to use broadcasting is a trade-off between this excess traffic and the gains made by using a simple routing protocol. As shown in Figure 5.6, the routers now only contain the local connectivity, and packets are blindly passed through the network.

Broadcasting in a 2-d mesh uses dimension-ordered routing, where packets are delivered along one dimension or axis first, such as the X-Axis, before they are switched along the perpendicular dimension, such as the Y-Axis, [171][173] in order to reach all destinations within the network. If all the packets originating from a processing core are delivered first of all along the same dimension then this will result within a bottleneck at a low packet injection frequency. This can be avoided by sending 50% of the packets first of all along the X-Axis and the remaining 50% of packets first of all along the Y-Axis. This effectively doubles the theoretical maximum throughput as the load upon the bottleneck channel within the network is halved. This process is illustrated in Figure 5.7.

As well as the reduced size of the routing tables broadcasting offers other benefits. For example, if the connectivity of the neural network is updated fewer routing tables need updating as there are significantly less routing entries. Whereas, in multicasting new paths will need to be determined and all the routing tables updated accordingly.

Updating of routing tables is challenging as it involves inserting additional traffic into the NoC in order to configure tables at specific routers. For broadcasting, fewer routing



(A) (B)

FIGURE 5.7: Alternative broadcasting routing options to increase maximum throughput. Packets are routed using a dimension-ordered scheme, where they first of all travel along one axis, before switching to the next axis. To increase throughput by reducing the load upon the bottleneck channel packets can be routed either X-Axis first (a) or Y-Axis first (b).

tables will need updating so this problem is reduced by a proportionate amount. The issue of updating or configuring routing tables is explored section 7.3.10

Also, broadcasting packets allows for all processing cores to listen to the state of the system. A module could be attached to listen to network traffic and pass on the information outside of the NoC. An example of this scenario is provided in section 7.3.9.

5.3.3.1 Theoretical

As defined previously the power consumption of the network is related to the link utilization. For broadcasting this is defined in (5.23). The mean link utilization is the amount of packets generated, T_p, multiplied by the distance that each packet is required to travel, D_ave, divided by the total number of links within the network.

$$T_{p,broad} = s\bar{\delta_n} \tag{5.21}$$

$$D_{ave,broad} = k^2 - 1 \tag{5.22}$$

$$U_{broad} = \frac{T_{p,broad} D_{ave,broad}}{L} = \frac{s\bar{\delta_n}(k^2-1)}{2k(k-1)} = \frac{s\bar{\delta_n}(k+1)}{2k} \tag{5.23}$$

This can be combined with the (5.5) and (5.8) to determine the estimated power consumption with broadcasting:

$$P_{broad} = V^2 f_n (C_O 2\sqrt{as}\bar{w}(k-1))(\frac{s\bar{\delta_n}(k+1)}{2k}) = V^2 f_n C_O \sqrt{a}\bar{w} f_s s\sqrt{s}\frac{(k+1)(k-1)}{k} \tag{5.24}$$

The maximum throughput is the bandwidth required to saturate the bottleneck channel. Since all packets are transmitted to each processing element this bandwidth is not equal to the bisection bandwidth. Instead, the throughput is limited by the bandwidth required to saturate the channels towards the edges of the mesh. This is caused by each processing element receiving all packets, but the elements within the corner of the mesh having only 2 channels upon which these packets can be received, as opposed to 4. In the following chapter of this thesis this concept is graphically illustrated and verified in Figure 6.17.

The bandwidth through the bottleneck channel with broadcasting is equal to the traffic generated by each processing element in each direction mechanism, T_m (Figure 5.7) (5.25) multiplied by the number of processing elements sending traffic through the bottleneck channel (5.26). From this it is possible to determine the maximum achievable firing rate for a broadcast mesh neural-NoC (5.27). As shown by this equation the maximum throughput will decrease asymptotically as the size of the NoC increases.

Equation 5.26 is best derived graphically, as in Figure 5.8.

$$T_m = \frac{s\bar{\delta_n}}{2k^2} \tag{5.25}$$

$$(k^2 - k) + (k-1) => k^2 - 1 \tag{5.26}$$

$$f_{s_{max,broad}} = \frac{2f_n}{s}\frac{k}{k+1}\frac{k}{k-1} \tag{5.27}$$

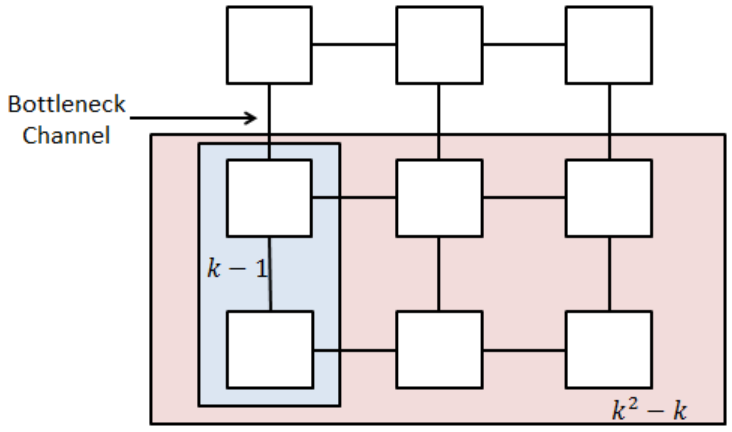


FIGURE 5.8: Graphical derivation of Equation 5.26. Each processing element produces T_m packets in each direction when broadcasting. The bottleneck channel will receive $k^2-k+k-1$ multiplied by T_m packets.

To calculate the latency of a broadcast mesh topology combine (5.11) with (5.28), which is the zero-delay hops. This value is the mean number of hops for each packet to arrive at all destinations within the network. The latency is shown in (5.29).

$$H_{0,broad} = \frac{3}{2}k - 1 \tag{5.28}$$

$$T_{0,broad} = R_{\square} C_O a \frac{3s}{2k}(1 - \frac{1}{k}) \tag{5.29}$$

5.3.3.2 Empirical Model

A similar model to that described in section 5.3.2.2 was developed. The model received as an input a neural network connectivity list, a mean firing rate and the network topology parameters. It returned the estimated energy and area requirements for a broadcast strategy. The model contained an option to consider the logical compression techniques described in section 5.3.5.

5.3.4 Comparison

As noted previously, with the assumptions required to make theoretical predictions multicast routing degrades to the same performance level as broadcasting. Using the empirical models the legitimacy of these assumptions are investigated in section 6.2 and

TABLE 5.3: Theoretical comparisons of different routing strategies and topologies for a neural-NoC. s is the number of neurons, k is the number of rows/columns within the NoC 2-dimensional mesh, and $\bar{N}_d$ the mean degree of connections for each neuron. Common factors such as operating voltage and frequency have been removed.

	Unicast	Broadcast Mesh	Broadcast Torus
Power	$s\sqrt{s}\bar{N}_d$	$s\sqrt{s}\frac{(k+1)(k-1)}{k}$	$2s\sqrt{s}\frac{(k+1)(k-1)}{k}$
Area	$\sqrt{s}(k-1)$	$\sqrt{s}(k-1)$	$\sqrt{s}2k$
Throughput	$\frac{k-1}{s\bar{N}_d}$	$\frac{1}{s}\frac{k}{k+1}\frac{k}{k-1}$	$\frac{2}{s}\frac{k}{k+1}$
Latency	$\frac{s\bar{N}_d}{k}(\frac{1}{k}+\frac{2}{3})$	$\frac{3}{2}\frac{s}{k}(1-\frac{1}{k})$	$\frac{4s}{k}$

section 7.2. However, it is possible to compare the theoretical scalability of broadcasting and unicasting. This is accomplished in Table 5.3. As can be seen, broadcasting is not influenced by the degree of neuron connectivity unlike unicasting. Therefore, for typical large-scale biologically-inspired networks with a large N_d broadcasting is expected to perform better. However, with a small N_d unicasting should be the preferred option.

Also, in Table 5.3 broadcasting within a mesh and a torus topology is compared. In a torus topology the interconnection links are on average twice the length as mesh interconnects [171]. Therefore, although a torus topology does offer twice the maximum throughput tham mesh, its area, power and latency are negatively affected.

5.3.5 Logical Compression

The routing tables used in broadcasting and multicasting include an index and a value. When a packet arrives its ID is matched with the list of indices stored in the routing tables. If a match is found then the value for that index is retrieved in order to complete the operation on the packet. As suggested by Khan et al. [174] logical compression techniques can be used to reduce the number of indices required within the routing tables. This process is illustrated in Figure 5.9 where x represents a "don't care" value.

Within the empirical models the optimized Espresso software package [175] is used to compute the logical minimization.

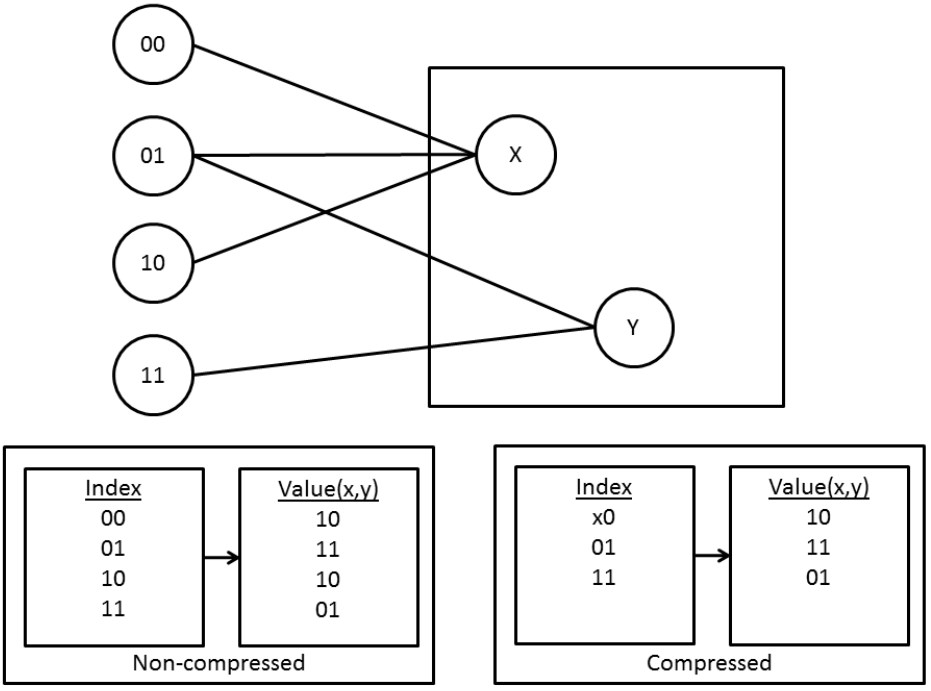
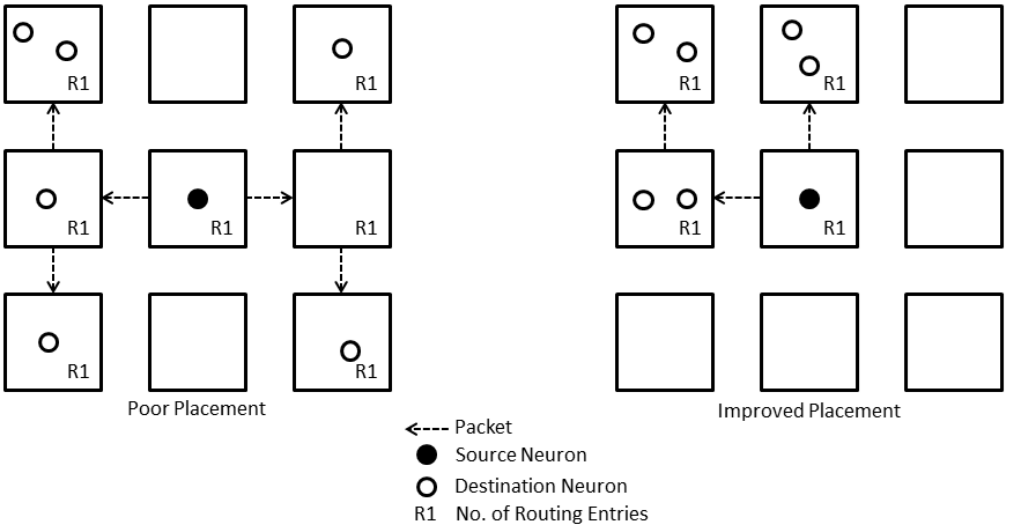


FIGURE 5.9: Comparison between non-compressed and logically compressed routing tables.

5.4 Partitioning

As mentioned previously, unicast and multicasting rely heavily upon the optimized placement of neurons upon cores. This can reduce bandwidth and routing overheads as packets are required to travel shorter distances and hence the overall energy consumption is reduced [176]. For multicasting it will also require fewer packets to be generated and fewer routing tables. This is illustrated by Figure 5.10, whereby in the left figure a total of 6 packets are transmitted and 6 routing entries are required, as opposed to the figure on the right, where the destination neurons are clustered around the source neuron and only 3 packets and 3 routing entries are required.

To optimize the placement of neurons upon cores the neural network must first be partitioned into sub networks. Each of these sub networks may be allocated to an individual processing element. Many algorithms exist to partition networks. Often these algorithms aim to reduce the number of edges from within a sub-network to other external sub-networks, such as the Kernighan-Lin algorithm [177]. For the developed empirical model the popular open-source software METIS [178] was utilized. The developers



		Unicasting	Multicasting
Energy, E_n	Poor Placement	11	7
	Improved Placement	8	4

FIGURE 5.10: Comparison of simulated annealing energies. By improving the placement of neurons upon cores, the energy of the state has decreased for both unicasting and multicasting.

claim that this software provides high quality partitioning in much quicker time than comparative software [179].

Once the network has been partitioned the sub-networks must be located upon the optimal processing elements to reduce the NoC overheads. Although METIS can provide an initial estimation of these locations, the process is extended by including a simulated annealing algorithm.

Simulated annealing is a common heuristic algorithm used to find the optimum solution to a particular problem, such as the placement of components within a circuit or the well known travelling salesman[180].

An initial state, s_n of the system is defined to have a set energy, E_n. The objective of the algorithm is to find the state with the lowest energy. A random change to the initial state produces a new energy value, E_{n+1}. If this energy is lower then the new state is accepted as an improved state and is used in further iterations. Alternatively, using a stochastic process the new state may also be accepted if it has a higher energy.

The likelihood of accepting a transition to a state with a higher energy is dependent upon a parameter known as the temperature, T_t. Initially the temperature is set high and therefore unfavourable transitions are more likely to be accepted. Further, transitions with a greater increase in energy are less likely to be accepted. This relationship is shown below, where $R(0,1)$ is a random number between 0 and 1:

$$P(E_n, E_{n+1}, T_t) > R(0,1) \tag{5.30}$$

As the algorithm progresses the temperature "cools". The lower temperature reduces the probability of a negative transition being accepted.

This stochastic process prevents the state of the system becoming stuck within local optimum points. This process continues until the algorithm is halted, usually dependent upon when an acceptable solution has been found or after a defined simulation time.

In the developed simulated annealing algorithm the state is represented as the location of sub-networks of neurons which are allocated to individual processing cores. The energy of the state is associated with the cost of routing packets between processing cores for all of the connected neurons. As the cost of routing packets differs for each routing algorithm, a different energy calculation is used, as described below. Within broadcasting all packets travel to all destinations so there is no benefit in optimizing the placement of neurons, therefore it is not considered for simulated annealing.

Unicasting - Energy Calculation Within unicasting the objective is to minimize the distance that packets between neurons are required to travel. Therefore, the energy of a neuron, E_s, is defined as the summation of the weighted edges, e_i, originating from the processing core to all other processing cores where the neuron has a connection. The weight corresponds to the Manhattan distance upon the 2-dimensional mesh between the two processing cores multiplied by the number of connections. The energy of a processing core, E_p, is then defined as the summation of all energies of neurons upon that core. The total energy of the state is the total of all the energies of all the processing cores. This is illustrated by Figure 5.11.

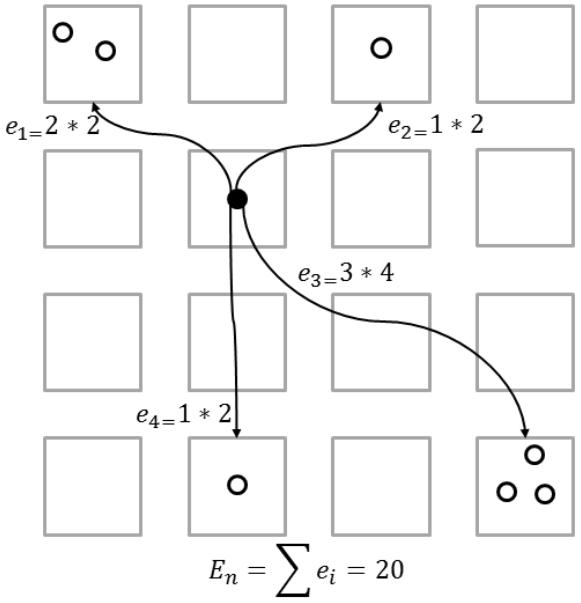


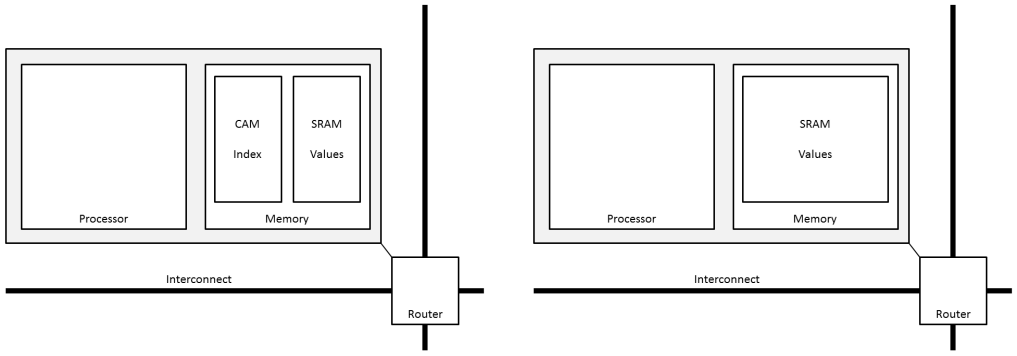
FIGURE 5.11: Calculation of the energy associated with a particular state for the unicast routing strategy.

Multicasting - Energy Calculation Within multicasting the objective is to minimize the number of routing entries that are required. Therefore, the total energy of the state is defined as the summation of all the routing entries that are needed within the network-on-chip.

Figure 5.10 illustrates an example of the energy associated with a unicasting and a multicasting routing scheme, along with how this energy is reduced by improved placement of neurons upon processing cores.

During each step within the simulated annealing algorithm the state is changed, this is accomplished by randomly swapping the location of two individual processing cores within the 2-dimensional mesh. Randomly swapping allows for exploration of the available problem space and is used by Kirkpatrick et al. [180] when they first illustrated the use of simulated annealing for improving the layout of digital circuits

In addition, at each step the temperature is cooled at a linear rate. After some time the temperature reaches a point whereby swaps which increase the energy are no longer accepted. Once this point is reached the algorithm continues until the local minima is found. The length that the algorithm runs for is dependent upon the size of the problem to be solved. For example, a 2x2 rectangular mesh with 4 processing cores has only 24 different states to explore, whereas a 4x4 rectangular mesh with 16 processing cores has



(A) (B)

FIGURE 5.12: The unit cell of a homogeneous neural-NoC. Each cell consists of a processor, memory for the connectivity, and a router to direct traffic between the cells. (a) For broadcast or multicasting. (b) For unicasting.

$16! = 2e13$ states to explore. The simulated annealing algorithm attempts to the find an optimum state by exploring only a subset of the total number of potential states.

The results of partitioning and the simulated annealing algorithm are demonstrated in section 6.2.1. This section uses the algorithms to map a typical neural cortical column onto a defined 2-dimensional mesh network-on-chip.

5.5 Unit Cell

A neural-NoC is a homogeneous system of many interconnected unit cells. To fully understand and evaluate the NoC this unit cell and its components must be fully studied. Unit cells, as shown in Figure 5.12, consist of the following main elements:

- Processing - responsible for calculating synaptic and neuron updates. A sample dVLSI neuron was described in the previous chapter. The developed empirical network model utilizes the sample dVLSI neuron as a baseline component. As such, no further details are provided in this chapter for the processing.
- Communication - transmitting packets from the source processing core to the target core requires a network infrastructure. This primarily consists of routing elements arranged in a mesh with interconnect wires between the routers. A router may have a local memory block available to assist in routing decisions.

- Memory - in a neural-NoC the routers are heavily dependent upon the memory required for the routing tables. These tables store a list of the neural network connectivity and for multicasting the routing information. For unicast routing the memory purely consists of RAM, whereas multicasting and broadcasting rely upon a combination of RAM and content-addressable memory (CAM). CAM memory provides a single cycle lookup function. When a packet arrives the CAM is searched for the identifier, and if a match is found then an index/address is passed to the RAM which returns the value. This process is described further in section 5.5.3.

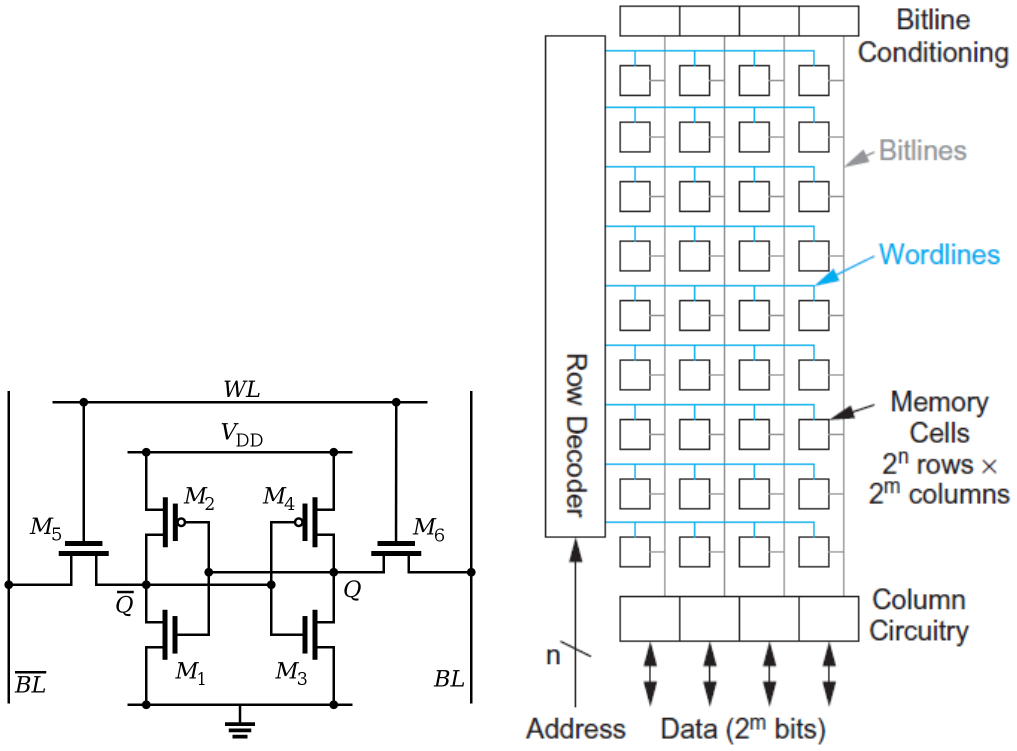
For each of the components of the unit cell, a model representing the energy and area of that component under differnt parametric conditions is described below.

5.5.1 Router

The routers are responsible for managing packetization of spike events from the processor, transmitting of the packets to the correct destinations, and depacketization of received packets back into spike events for inclusion in the neural models. When required buffers within the routers may temporarily hold packets until they are able to progress.

The routers within a broadcast scheme are simple and efficient - any packets arriving on a channel will be forwarded to other applicable channels unconditionally. For multicast, the packet contents will be checked against a lookup table for an instruction, if none is found then the packet will be passed along a default route. For unicast, a router will inspect the contents of the packet and determine which way to route the packet correspondingly.

[181] states that the power consumed by a NoC's communication is primarily through the energy dissipated in the switching of the interconnect wires, indeed [169] use this metric alone to compare power in different communication topologies. Therefore, in this thesis only the area and energy of the interconnects are considered in terms of the communication infrastructure. The size and power consumption of the routers is believed to be negligible in comparison to other components, such as the interconnects.



(A) (B)

FIGURE 5.13: SRAM Architecture. (a) A 6 transistor memory cell [182]. (b) Memory array layout. Adapted from [139].

5.5.2 RAM

The RAM is used to store the values held within the routing tables. SRAM/DRAM is preferred over standard flip-flop memory cells because they are smaller in size. An SRAM cell requires only 6 transistors, as illustrated in Figure 5.13a, and a DRAM cell requires only a single transistor and capacitor. This reduction in memory cell size can result in higher density of memory and a reduced power consumption [139].

Although DRAM provides a higher density of memory SRAM is faster and easier to access. DRAM also requires regular refreshing to prevent the memory state from being lost. Within the empirical models both DRAM and SRAM are compared to determine the most suitable choice for the application.

The RAM cells are arranged in a matrix topology with row and column circuitry to read and write to specific cells as in Figure 5.13b. When the cell value is written the complementary bit lines (BL) are set as required and the write line (WL) is asserted. This sets the state of the cross-coupled inverter. To read the value the bit lines are left

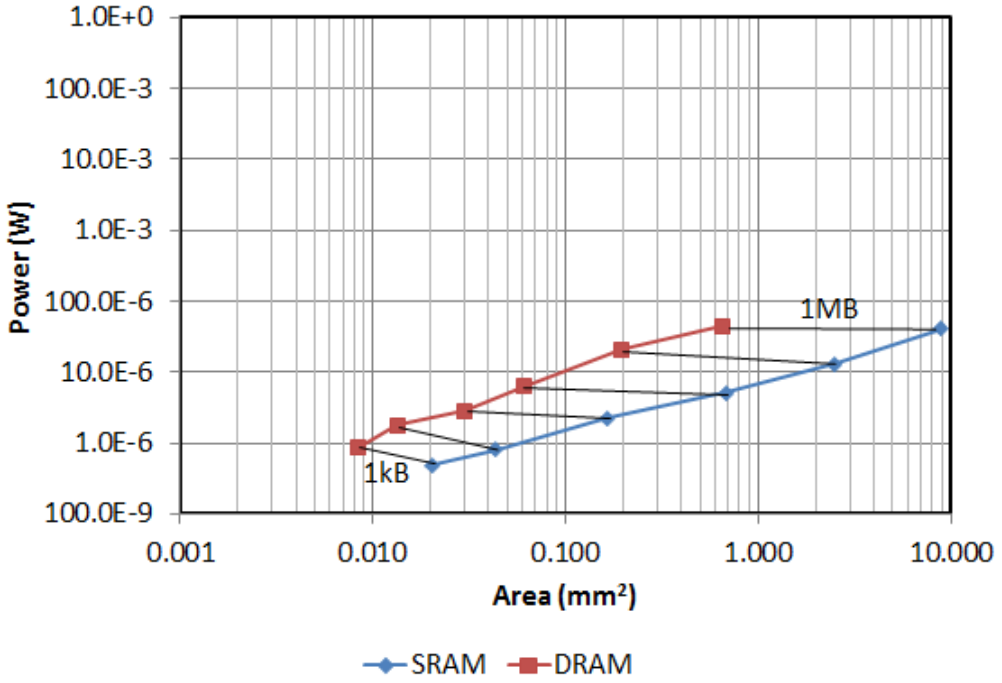


FIGURE 5.14: Area and power relationships for a varying memory size, assuming a 65nm technology node and 100,000 memory accesses per second.

floating whilst the write line is selected [139]. The values of the cell can then be detected by the sense circuits in the column circuitry

Rough estimation of the area of a RAM memory block can be achieved by simply determining the number of cells required [139]. However, energy in a RAM is consumed in many different components and therefore energy modelling is not a simple task [183][184]. In fact it is a topic for a PhD thesis in itself [185]. Energy can be consumed in the memory cells, within the decoding circuitry and within the bit and write lines. To complicate matters further the energy consumed in each of these components is dependent upon the size, structure and layout of the whole memory system [184].

CACTI [186][187][188] is a memory modelling tool provided by HP Labs. It allows for a user to accurately estimate the area, energy and delay of different memory models. Although primarily focused upon modelling caches for high speed microprocessors a RAM interface is also provided for estimating SRAM and DRAM parameters. In Figure 5.14 the area and power parameters for SRAM and DRAM are explored. As shown a 1kB RAM will occupy twice as much area in SRAM technology but consume 50% of the power of DRAM. As the memory size grows DRAM performs better. Presumably this

gain is caused by the shorter interconnect (and therefore lower dynamic power) as each memory cell is smaller.

The CACTI platform provides a baseline for performance. Increased performance can and has been achieved through the use of techniques including sub-threshold operation [189] and reduced memory cell transistor count [190].

The empirical model described in this chapter, and used in Chapter 6 and Chapter 7, determines the depth and width required for the routing tables. This information is then fed into the CACTI tool to determine the optimal memory arrangement and to provide area and energy information. For example, if the empirical model suggests that each routing table is required to be 1024 words deep with a 32-bit word width and with 100k read accesses per second then the CACTI tool suggests that approximately 500nW of power and $0.02mm^2$ of area is required. This suggests that each SRAM memory cell is $0.6\mu m^2$, which is in the region previously demonstrated for 65nm technology by multiple groups [191][192].

5.5.3 CAM

A CAM is very similar to an SRAM but it can also perform single-cycle search operations. If an input key is provided to the CAM then any row which matches the key will respond with a match signal. An address encoder may then take this signal to provide an index into a dedicated RAM cell, such as is shown in Figure 5.15. Alternatively, the match signal could be used directly as the wordline in the RAM cell, if the CAM/RAM blocks were closely integrated. There is a design trade-off to be considered between choosing an address encoding or the direct wordline access, which is dependent upon the size of the CAM/RAM.

CAMs are used within the NoC infrastructure to provide a quick search functionality. When a router receives a packet, the contents of the packet may be checked against the contents of the CAM in a single clock cycle. The CAM may then return a result indicating what action the router should apply to the packet.

A ternary-CAM (TCAM) extends the standard CAM by accepting "don't-care" values, whereby multiple rows may match the input key if a subset of the input key matches.

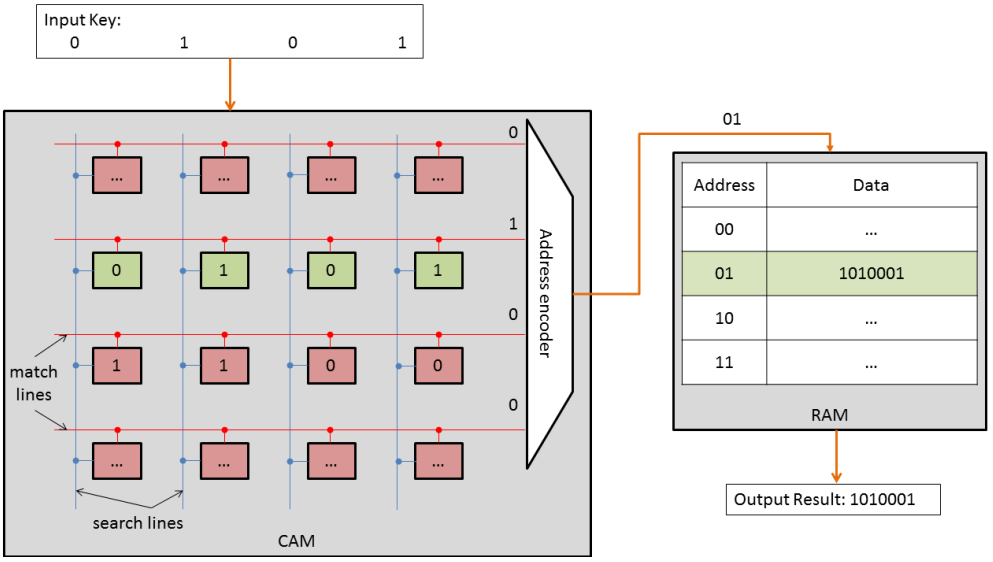


FIGURE 5.15: Arrangement between content-addressable memory (CAM) and random access memory (RAM). The CAM receives an input key, if a match is found it produces an ouput. This output can be used to index a RAM to then provide a result.

TABLE 5.4: Summary of the state-of-the-art in CAM developments.

Authors	Institution	Year	Size (kbits)	Width (bits)	Search Speed (ns)	Fab Size (nm)	Voltage (V)	Cell Area μm^2	Energy (fJ/bit/search)
Pagiamtzis, Sheikholeslami	Toronto, Ca	2004	37	144	7	180	1.8	-	2.89
Cho, Sohn Yoo	KAIST, Kor	2005	144	144	2.2	100	1.2	22.4	0.7
Noda et al.	Renesas, Ja	2005	4500	144	7	130	1.5	3.59	-
Yang et al.	Chungbuk, Kor	2011	18	144	2.5	180	1	-	2.82
Huang and Hwang]	Taiwan	2011	37	144	2.5	65	1	7	0.165
Hayashi, Noda et al.	Renesas, Ja	2013	4500	72	4	65	1	1.69	1.98
Arsovski et al.	IBM	2013	1300	640	1	32	0.95	-	0.58

A typical CAM cell extends the standard SRAM cell by adding 4 extra transistors, a match-line and a search-line [139]. Figure 5.15 illustrates how these lines connect to the CAM cells. Initially, the match-lines are all pulled high, then the input key is placed onto the search-lines. If there is a mismatch between a CAM cell and the input key then the match-line is pulled low. If all the cells within a word are equal to the input key then the match line remains high.

CAMs typically consume a large-amount of power. This is due to each cell being searched when an input key is being received. Most of the energy is consumed in the match-lines [193]. Every match-line attached to each row in the CAM is pulled high and if no match is found then it is pulled low again. This results in a very high activity rate.

CAMs are often used in routing internet traffic. Hence, there is great commercial and research interest in developing low-power systems. A summary of the latest developments in CAM technology is provided in Table 5.4. As can be seen the state-of-the-art technology in terms of low-power is consuming under 1fJ/bit/search.

Agrawal et al. [193] provide an extension to the CACTI tool for modelling power and delay for ternary-CAMs which the empirical model described in this chapter utilizes.

Similarly to the RAM system, the empirical model calculates the size of each routing table and how often it is searched. Both of these parameters are dependent upon the routing strategy. From these parameters it is possible to determine the overall energy consumption using the model provided by Agrawal et al and the optimized statistics from Table 5.4. For the area estimation of the CAM cells the original CACTI tool with a modification to take into account the enlarged cell area used by CAMs can be utilized.

For example, for a CAM cell involving 1024 words of 32-bits each, with 100k search accesses per second and a search cost of 1fJ/bit/search the CAM will consume an estimated $3.28\mu W$ of power. With a CAM cell approximately twice the size of an SRAM cell this CAM arrangement will occupy $0.04mm^2$.

5.5.4 Interconnect

Spike events are translated into packets that are required to be transmitted around the network. Transmitting of these packets involves significant energy consumption [133][194]. Equation (5.31) illustrates how to calculate this dynamic power:

$$P = CV^2fU \tag{5.31}$$

To determine the energy consumed in a neural-NoC interconnect the empirical model must calculate the total capacitance, C, be provided with the mean voltage swing, V, and calculate the switching rate of each interconnect link, fU.

The capacitance of a global interconnect is well studied since it has a significant affect upon the power and delay of a digital circuit. The problem is exacerbated with current and future technologies as global wires start to dominate a digital circuit's performance more than the gate design [194][195][196]. This is due to global wires increasing in length and the capacitance per unit length of an interconnect only reducing slowly with each fabrication node [197].

A typical interconnect wire has a capacitance per unit length of between 0.1-0.2pF/mm [139][198][171][199][200][197]. Therefore, the energy consumed to transmit a single bit

over an interconnect with a typical 1V swing is between 0.1-0.2$pJ/bit/mm$. This value can be reduced by using low-swing techniques [198][199] or intelligent encoding techniques to reduce the number of transitions required [194][201]. However, as a baseline, the empirical model assumes a value of 0.1$pJ/bit/mm$.

The mean interconnect link length can be determined by taking the square root of the total area of the unit cell. The area of the unit cell can be calculated by summing the area consumed by the processing core and the memory elements described in the previous two sections. The empirical model combines this mean interconnect length with the calculated bandwidth and the interconnect energy rate to determine the estimated power consumption.

For example, if we assume that the unit cell consists of a processor with a size of 0.04mm^2, a RAM with a size of 0.02mm^2 and a CAM of 0.04mm^2 then the total unit cell is 0.1mm^2. Therefore each interconnect link is 0.32mm long. If 100,000 packets/sec of 32-bits each traverse across this link with 0.1$pJ/bit/mm$ then the total power consumption is approximately 100nW.

5.5.5 Unit Cell Summary

In this section the model of each sub-component of the unit-cell has been described with a focus upon the energy and area consumption. The total energy and area consumed can be estimated by summing all of the sub-components. The resource utilization of each sub-component varies with the granularity of the unit cell. As such, to determine the optimal unit cell parameters the neural-NoC must be considered with different granularities. This has not previously been investigated fully.

5.6 Process Flow

The information provided within this chapter forms the basis of a methodology to determine optimum design parameters for a neural network-on-chip, with a focus on a 2-dimensional rectangular mesh topology.

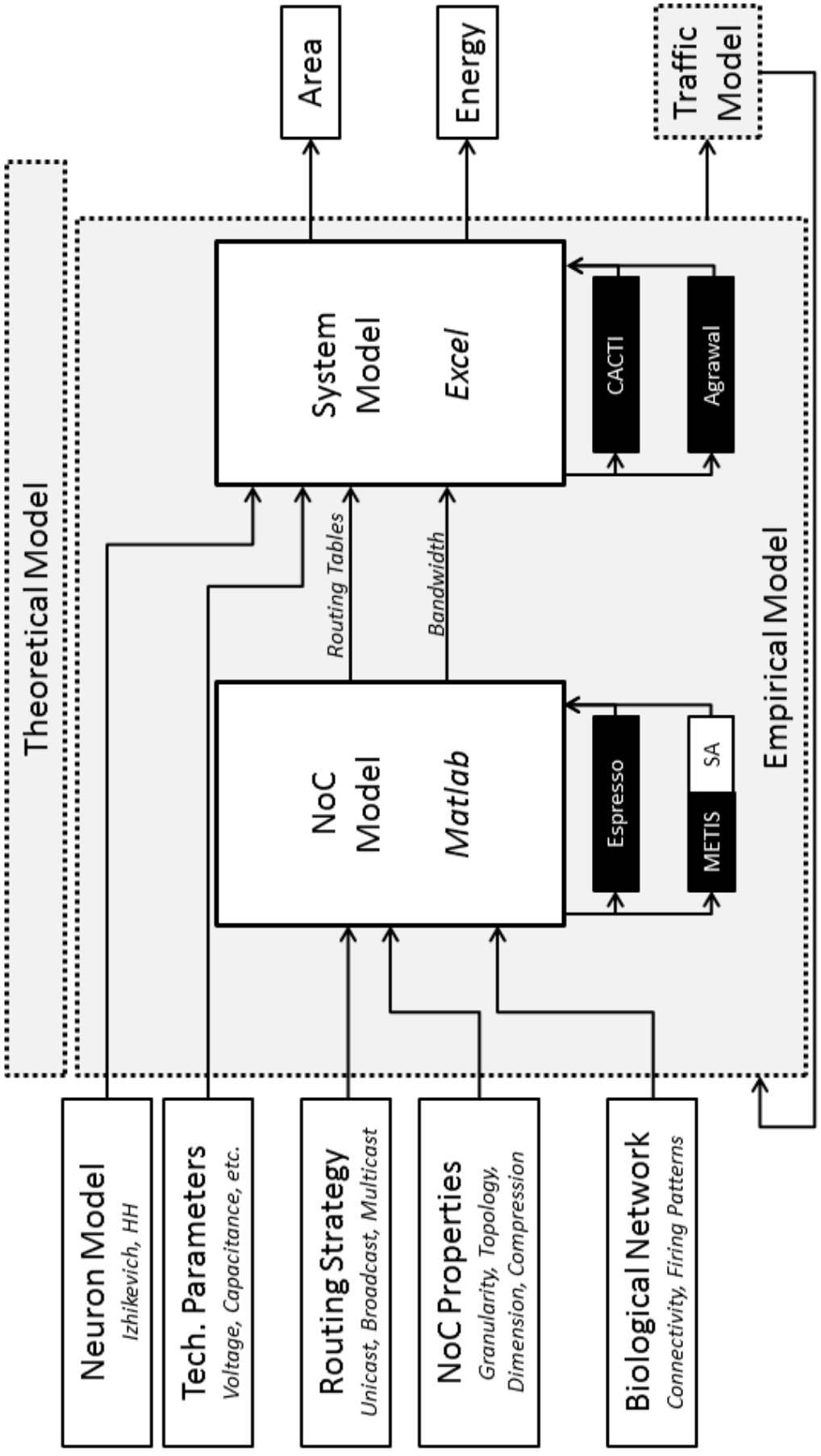


FIGURE 5.16: The process flow for evaluating a neural-NoC system. The empirical model receives details about the system including the network to be implemented, the NoC properties and the routing strategy. This information is first of all fed into a matlab model of a NoC system which calculates the required routing tables and the overall bandwidth. The complete system model combines these results with the technology dependent circuit parameters and the neuron model developed in section 4.5 to determine the overall energy and area consumption of the system. The model allows the optimal parameters to be determined prior to any design development. The black boxes are sub-models used by the empirical model that have been developed by other research groups. The traffic model in the bottom right hand corner can be used to verify the correct function of a proposed system. The implementation of the traffic model is described in section 6.3.

A neural network containing s neurons could be implemented upon a network-on-chip using $m = k^2$ processing cores. But, the number of processing cores has a significant impact upon the maximum achievable performance of the design, upon the energy consumed, and upon the silicon area required. This is shown in Table 5.3, where the theoretical constraints of power, area, throughput and latency of a network-on-chip are described.

A neural-network-on-chip relies upon the communication of packets between neurons on different processing cores, which represent the transmission of action potentials. The theoretical constraints shown in Table 5.3 also illustrate that the methodology for routing of packets between processing cores has a significant effect upon the end design.

Within section 5.3.1.2, section 5.3.2.2 and section 5.3.3.2 the empirical models for unicasting, multicasting and broadcasting routing schemes are defined. These empirical models, which are implemented in Matlab, take two primary parameters- 1) the connectivity of the neural network to be considered, 2) the size of the network-on-chip in terms of processing cores. From these parameters the empirical models calculate the size of the routing tables which are needed by the network-on-chip, as well as the number of transmitted packets that would be expected to transverse the network, otherwise known as the bandwidth. The models also rely upon the external tools, Espresso, Metis and the simulated annealing algorithm, to determine the optimum results.

The system model is used to calculate the amount of energy and the silicon area which is to be used by the proposed design. This system model requires three sets of parameters, 1) the size of routing tables, 2) the network bandwidth, and 3) the single neuron hardware design described in Chapter 4. The system model calculates the energy and area results using the information provided in section 5.5.

By varying the initial parameters: neural network connectivity, network-on-chip size, routing strategy; the complete model allows for the optimum parameters to be determined in order to reduce energy and/or area. This model is illustrated in Figure 5.16 and is used in the succeeding chapters.

For instance, in the next chapter a typical neural cortical column involving up to 65,000 neurons is investigated. Figure 6.11c shows the forecast area/energy consumption for the cortical neural network, which is dependent upon the routing strategy and the size

of the network-on-chip. It is shown that for this specific cortical network, energy/area can be reduced by up to 5x with judicious selection of network-on-chip size. Similarly, Figure 7.3 shows the energy/area relationship for an alternative neural network.

Further, the traffic simulation described in the next chapter can be used to verify correct operation of the proposed network-on-chip when considering the design parameters produced by the empirical model.

5.7 Summary

In this chapter different network protocols previously described in the literature have been introduced and compared. From this study a network-on-chip approach was deemed to be the most suitable for implementation to meet the defined specification. For the NoC approach, the available design options have been theoretically explored and a model developed to determine the expected energy and area requirements of a completed implementation. In the following chapters this empirical model is used to propose two different NoC designs for two alternative neural network challenges.

This model provides an analysis only for a 2-dimensional rectangular mesh NoC as it is the most common and the simplest to implement within a single silicon chip. The model allows for the investigation into various trade-offs within the design, primarily the granularity of the processing core. These trade-offs have an impact in terms of power, area and performance, as shown within the following two sections. The model could be extended to study alternative trade-offs, including investigating alternative topologies, such as torus, or potentially multi-dimensional systems- perhaps including 3D integrated circuits.

Chapter 6

Neural Network Case Study 1 - Cortical Column on a Chip

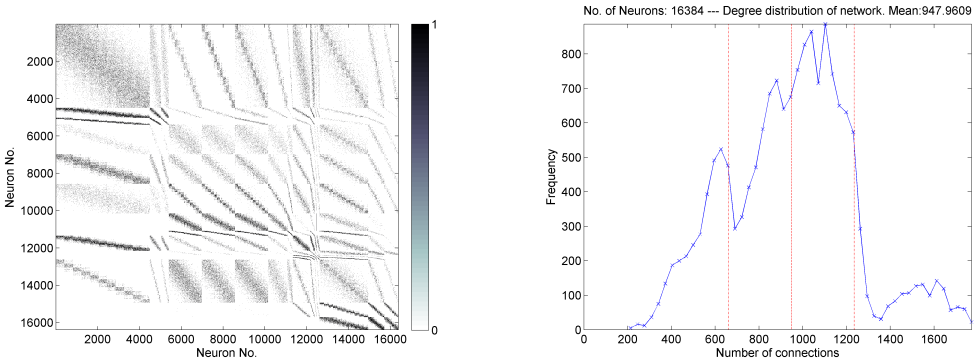
"There is nothing like first-hand evidence"

A Study in Scarlet, Sherlock Holmes

Sir Arthur Conan Doyle

In this chapter the previously described methodology is used to study the optimal design implementation of a standard biological neural network. The focus of the design is a single chip NoC implementing a typical neural cortical column.

In the first section, the cortical column neural network under consideration is introduced, before the results of putting this network through the NoC empirical model are shown in order to determine the optimal design parameters, such as granularity and routing strategy. The capabilities of the proposed neural-NoC design are tested by a software traffic simulator that studies characteristics including maximum bandwidth. The achievable maximum bandwidth is shown to closely correlate with the theoretical predictions from Table 5.3. The final section discusses the implications of the results alongside providing potential improvements, both to the design and to the empirical model.



(A) (B)

FIGURE 6.1: Connectivity statistics for a typical neural cortical column as described by Binzegger et al. [205]. This neural network has become a benchmark for the performance of modelling systems and has been used by groups including SpiNNaker [109] and SyNAPSE [104]. This network is used throughout this chapter to study the optimal design of a neural-NoC. (A) Connectivity matrix of neurons within the network, (B) The degree distribution of the neurons. On average a neuron is shown to have approximately 1000 connections.

6.1 Neural Network

A cortical column is a microcircuit of interconnected neurons thought to be a fundamental building block of processing within the cortex [202]. They are thought to be repeated structures each of which performs the same computation but on a different subset of the inputs and producing a different subset of the output. Due to the amount of investigation into columns there is a significant amount of anatomical data available, making these microcircuits an attractive choice for implementation in silicon. In fact, their implementation is still the first major milestone objective of many silicon neural network implementations [2]; this is despite the speculation about the existence and function of columns [203][204].

In 2004, Binzegger et al. [205] provided a detailed analysis of the visual cortex of a cat. The circuit model that they developed has been used by the majority of the silicon neural network developers whom are interested in studying biological systems as the fundamental column of the cortex [106][104][206][109]. As such, this circuit can be considered a suitable benchmark for a single-chip implementation for a neural prosthesis and it can be used comparatively with other previous silicon implementations.

The neural network model[1] combined anatomical connection statistics [205] with axonal

[1]The author utilized a model developed by Richard Tomsett, based around the Binzegger results.

spread measurements [207][208]. This model allows for neural networks between 10,000 and 100,000 to be generated with realistic biological properties.

For the purposes of neural-NoC implementation, the network model can be simplified to a simple binary matrix representation removing superfluous biological details. A sample connectivity matrix is illustrated within Figure 6.1a. Within this matrix, a connection between two neurons is represented as 1, and no connection represented as a 0. As shown in Figure 6.1b, each neuron has on average nearly 1,000 connections.

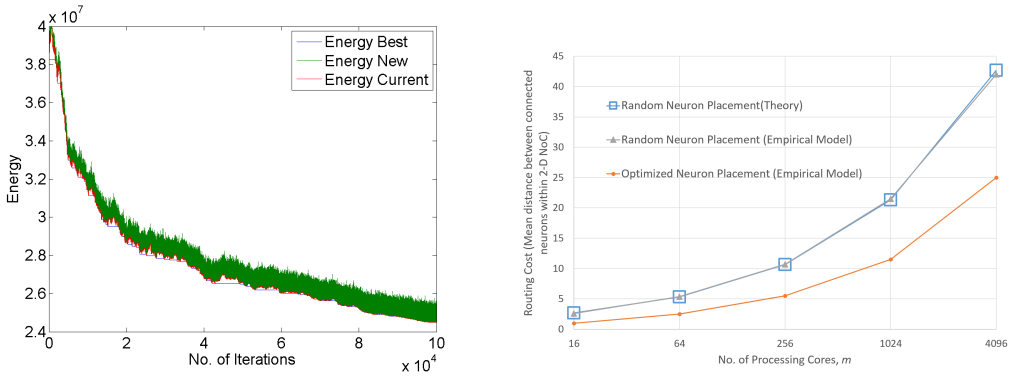
6.2 NoC Analysis

6.2.1 Neuron Placement

The empirical model described in section 5.6 provides an estimation of the number of routing entries and the bandwidth for each routing strategy. From these figures an estimation of the performance, the energy consumption, and the silicon area of a neural-NoC can be determined. Placement of neurons upon specific processing cores has an impact upon the routing strategy, as it affects the distance that packets have to travel to communicate between two connected neurons. Therefore, to develop an estimation of the performance of a routing strategy the best possible placement must be provided. For instance, a multicasting routing strategy scheme with randomly placed connected neurons degrades into a broadcasting strategy, as each neuron is likely to have to send a packet to every processing core due to the random placement. Hence, optimal placement is important to achieve the best performance and area/energy rewards.

Many research groups [132][119] have stated that due to the clustering of neurons within a neural network, communication overheads can be reduced by forming localized groups of connected neurons within the silicon architecture. As such, section 5.4 described a process for partitioning a neural network and mapping it to a 2-dimensional grid. This partitioning process means each packet has to travel a shorter distance through the NoC, reducing the bandwidth and routing table overheads.

In Figure 6.2a the result of the simulated annealing algorithm described in section 5.4 is shown. This figure highlights the reduction in routing overheads that is achieved by optimizing the placement of neurons upon processing cores using simulated annealing.



(A) (B)

FIGURE 6.2: Partitioning and placement of neurons onto cores. (A) Results of simulated annealing algorithm to find the lowest routing cost. (B) Reduction in routing cost for different size NoCs by using the optimized neuron placement algorithm described in section 5.4. The empirical model is used to calculate the routing cost, as can be seen the routing cost calculated with random placement closely correlates with the theoretical prediction provided by Equation 5.1

This optimization is further illustrated in Figure 6.3. This figure shows three sample neurons from the network described in section 6.1 implemented upon two different network-on-chip grid sizes, using random and optimized placement. The neurons are located on a particular core as labelled. The strength of connection between the neuron and its connected destination processing cores is quantified through the colour. Figure 6.3a shows a small-scale example to provide further explanation.

The degree label indicates the number of neurons that the source is neuron is connected, which as shown in Figure 6.1b is approximately 1000 neurons on average. The energy label represents the energy calculation used during the simulated annealing algorithm, as described in section 5.4.

As can be seen the optimization process does create clusters around the processing core of the source neuron. However, some neurons are still connected to a considerable proportion of the cores within the network-on-chip. In the following sections assume all of the neural networks have gone through this partitioning and placement process.

6.2.2 Traffic Bandwidth

Figure 6.4 shows the traffic bandwidth information for a neural network with 65,000 neurons as described in section 6.1 for three different routing strategies implemented upon a rectangular mesh 2-dimensional network-on-chip. The NoC contains multiple

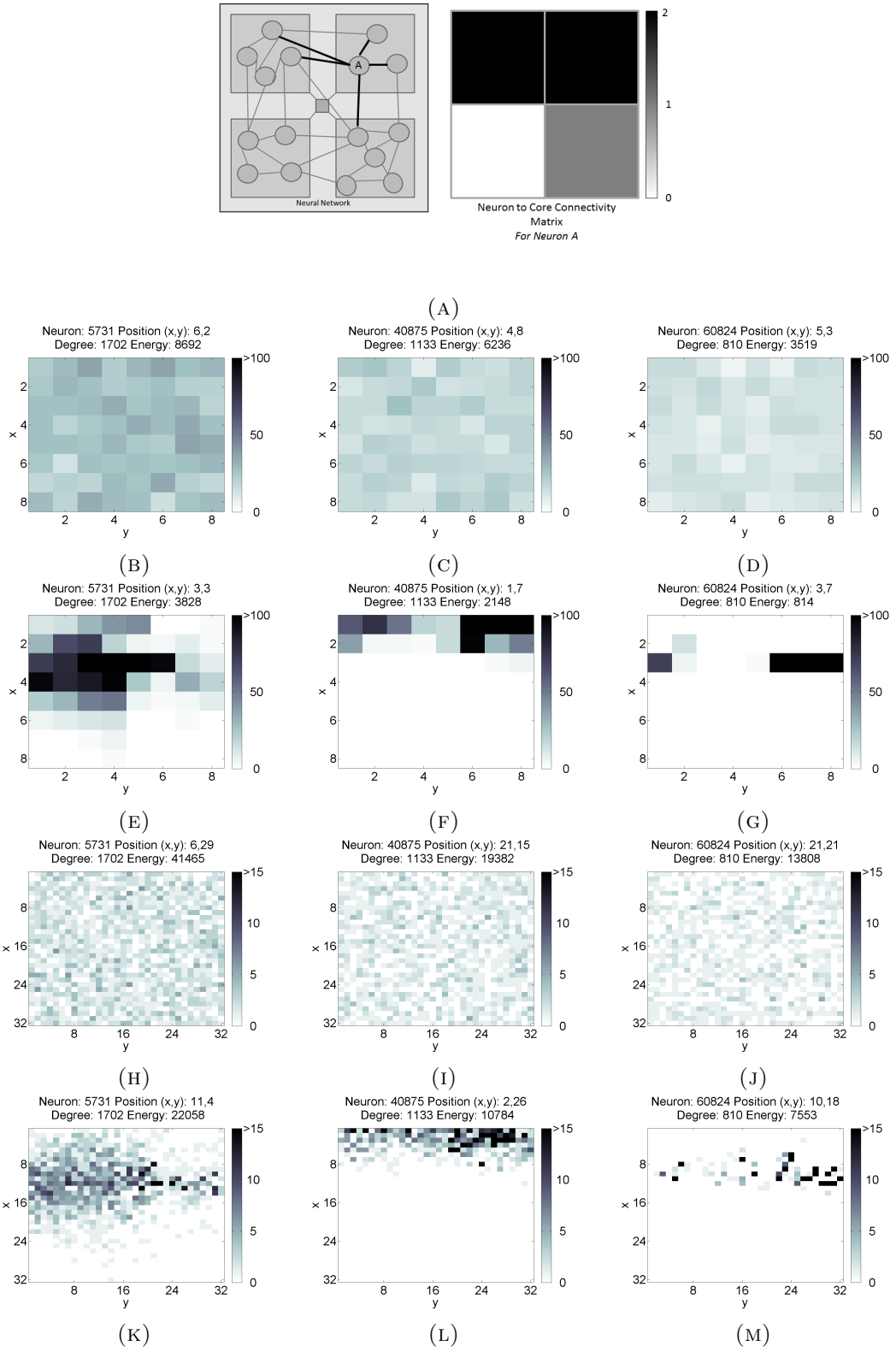
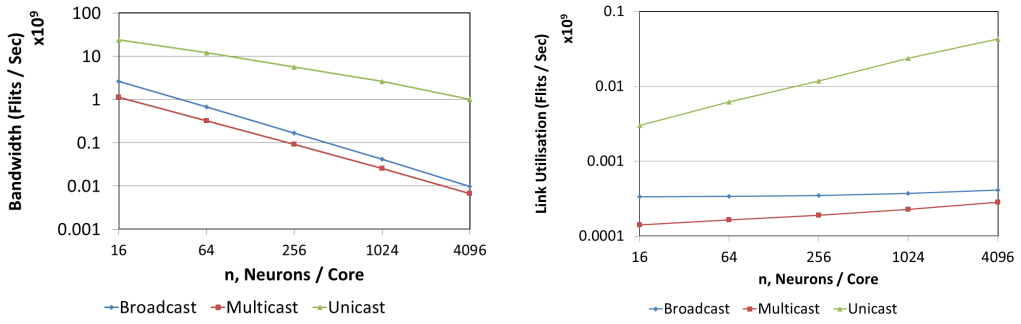


FIGURE 6.3: Optimizing placement of neurons. Degree represents the number of neurons that the source neuron is connected to. Energy represents the routing cost for a source neuron as described in section 5.4. (A) Method to interpret the succeeding figures. A neuron A is connected to 5 other neurons. Two of which are upon its own core, two upon the core to its immediate left, and one on the core below. This information is represented in the neuron-to-core connectivity matrix. (B)-(D) The connectivity of three neurons randomly placed within a 64-core network. (E)-(G) The same three neurons but with optimization of the placement. (H)-(M) The same information but for a 1024-core network.



(A) (B)

FIGURE 6.4: Traffic results for a neural network containing 65,000 neurons for three different routing strategies implemented upon a network-on-chip with a varying number of processing cores. (A) The overall system bandwidth. A *flit* is the communication of one packet from one processing element to a neighbouring processing element. (B) The mean utilisation of each link within the network. With more processing elements, there is more interconnections links, each of which can operate at a lower frequency.

.

processing cores which can each model a variable number of neurons. If each processing core can model more neurons then fewer cores are required.

As expected, when the number of neurons upon a processing core increases, and therefore there are fewer cores, the overall system bandwidth decreases. However, as shown by Figure 6.4b the mean frequency that each interconnection link within the system operates at increases.

The unicast routing scheme is many orders of magnitude more inefficient than the other two routing schemes. This is to be expected due to the high degree of connectivity within the neural network as shown in Figure 6.1b, as for each connection a separate packet is transmitted. Multicasting does not perform significantly better than broadcasting, even with the targeted routing of packets. This is due to each neuron being connected to a wide spread of processing cores throughout the NoC, despite the partitioning and placement routines described in the previous section.

In section 5.3.2 core-based routing was described. This involves every packet originating from a processing element being routed in the same direction, as opposed to a neuron-based routing, where the packet is routed according to the source neuron. Figure 6.5 demonstrates the effects of core-based routing for the proposed single-chip system. As can be seen with core-based routing the bandwidth degrades to the same performance as that of broadcasting. This is due to the inter-core connectivity, whereby due to the

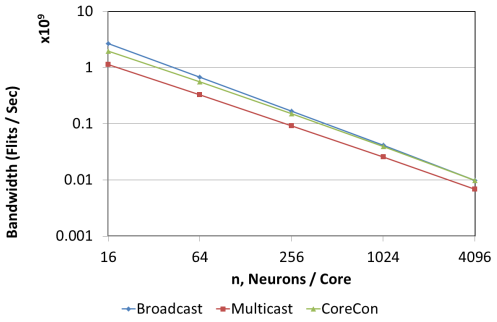


FIGURE 6.5: Traffic bandwidth comparison with core-based routing suggested by [172].

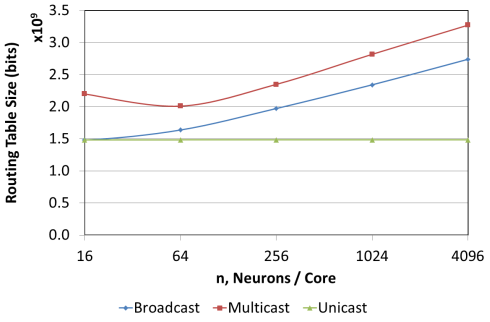


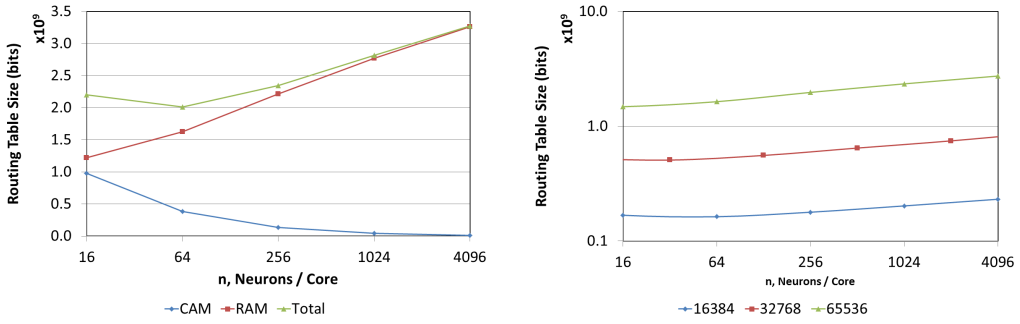
FIGURE 6.6: Total size of routing tables versus number of processing elements for a 65,000 neuron network.

large degree of connectivity each processing core is connected to almost every other core within the NoC.

6.2.3 Routing Entries

The empirical model also provides information about the number of routing entries required for the NoC. In Figure 6.6 the total size of all the routing tables for the three different routing mechanisms is illustrated. This figure studies a neural-NoC with 65,000 neurons implemented upon a varying granularity of processing core. The size of routing tables for unicasting remains static for each granularity, however, for both broadcasting and multicasting the size of routing tables is dependent upon granularity.

Interestingly, there is an optimal granularity to reduce the routing table size. This is caused by the total number of routing entries being a combination of two different memory elements, RAM and CAM. At coarse granularities, the memory is dominated by RAM, whereas at finer granularities the RAM requirements reduce whereas the CAM requirements increase. This is illustrated by Figure 6.7a, which shows the RAM/CAM



(A) (B)

FIGURE 6.7: Finding the optimal granularity of processor to reduce size of routing tables. (A) Contribution of CAM and RAM to total memory consumption for multicasting. (B) The optimal point varies with the size of the neural network that is to be implemented.

combination for multicasting. Figure 6.7b shows the total routing table size for broadcasting when implementing three different size neural networks. As can be seen, the optimal granularity changes depending upon the size of neural network that is to be implemented.

6.2.4 RAM

Section 5.5.2 describes the methodology for studying the area and power consumption for the RAM component of the unit cells. In Figure 6.6 the total memory within all the routing tables varies with the granularity of the processing cores. This is dissected in Figure 6.8a to show the RAM required per unit cell. It is for these size cells that the area and energy consumption is illustrated in Figure 6.8b and Figure 6.8c. The model indicates that for the size of memories required DRAM should be the preferred option.

Clearly, as the number of neurons increases per processing core greater area and energy resources for each core are required. However, the total resource usage for a given neural network size must be considered. If a processing core models fewer virtual neurons more processing cores are used. The total area and energy resources consumed by RAM is detailed in Figure 6.8d. As shown, to minimize RAM overheads many fine grained processing cores should be used as opposed to a small number of large processing cores. With the optimal parameters the power and area consumed by the RAM can be reduced by 7x to only 13mW and $1cm^2$.

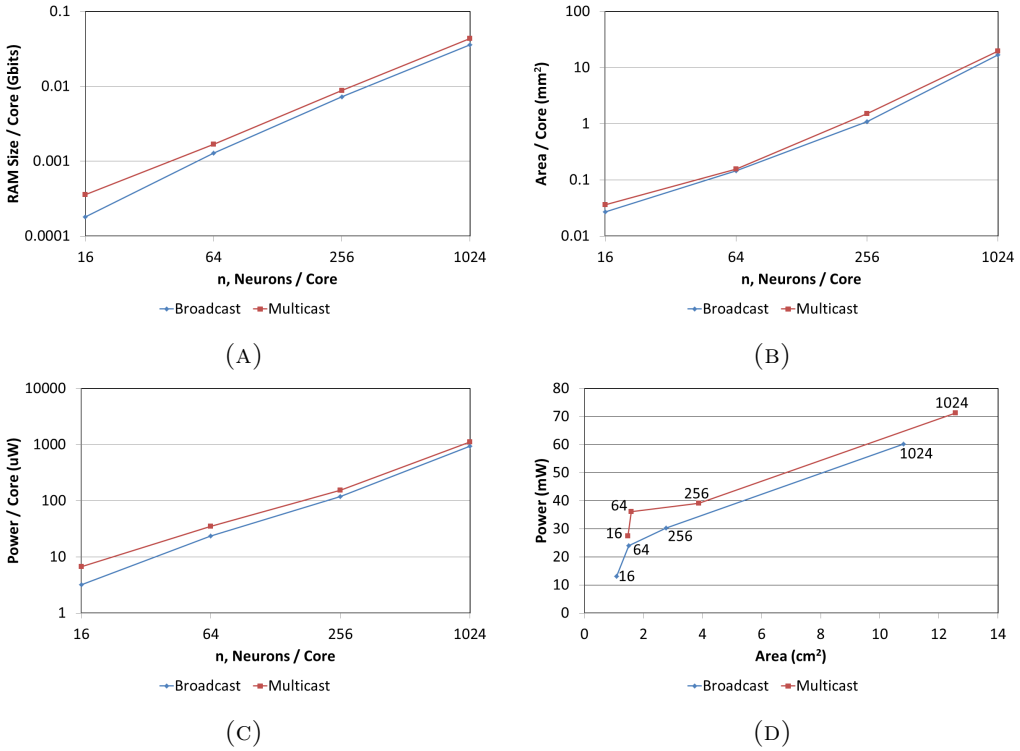
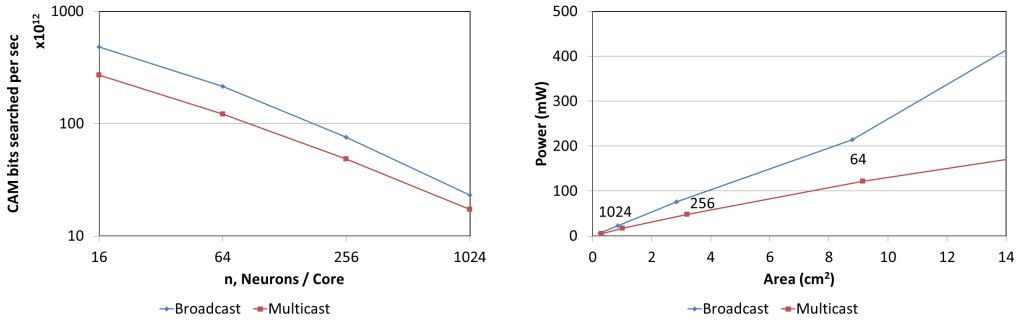


FIGURE 6.8: The area and energy resources required by the RAM component of the neural-NoC. (A) The size of the RAM required by each unit cell increases as the number of neurons increases per unit cell. (B) The area consumed by each RAM cell. (C) The power consumed by each RAM cell. (D) The power and area relationship for a neural network of 65,000 neurons implemented upon a NoC with varying granularity. The labels represent how many neurons per processing core.

Also, as expected Figure 6.8 shows that in terms of RAM resources broadcasting is more efficient than multicasting. This is caused by the non-requirement to store extra routing information. Unicasting is not considered here as the bandwidth is too large as shown previously and as such it is not expected to be the optimal strategy.

6.2.5 CAM

The content-addressable memory is also expected to consume a significant proportion of the available resources. A CAMs energy consumption is strongly correlated with the number of search operations it is required to carry out. Figure 6.9a illustrates how the number of search operations varies with the processing core granularity. As shown, with a fine granularity many more search operations are required. This is due to the increase in the number of processing cores, and therefore CAMs.



(A) (B)

FIGURE 6.9: The area and energy resources for the CAM component of the unit-cells. (A) The total number of search operations required varies with the granularity of the processing core. (B) The area/power relationship of the CAM component for a varying granularity. To reduce resource usage in the CAMs a coarse granularity should be used.

For broadcasting, when a neuron produces a spike a packet is transmitted to every processing core and therefore every CAM is searched. By having more processing cores more searchers are undertaken, albeit each search is within a smaller CAM unit.

As can be seen by Figure 6.9b, the CAM area/power relationship scales in the opposite direction to that of the RAM. To obtain the minimum area and power relationship many neurons should be implemented upon each processing core. Also, by comparing Figure 6.9b with Figure 6.8d CAMs are expected to consume 10x the power of the RAM units, although a similar area is expected.

6.2.6 Interconnect

To determine the energy consumed by the interconnect fabric the mean length of each interconnect link must first be determined. This can be estimated by taking the square root of the area of a unit cell. Figure 6.10a illustrates how the area of a unit cell varies with processing core granularity. This area is a combination of the area of the RAM and CAM defined in the previous section alongside the area of the neuron model calculated in Chapter 4. It can be seen that area is predominantly consumed by memory.

This unit cell area leads to the mean link length illustrated by Figure 6.10b. This figure confirms the theoretical relationship described in Equation 5.7, whereby the mean link length grows with the square root of the number of neurons per processing core.

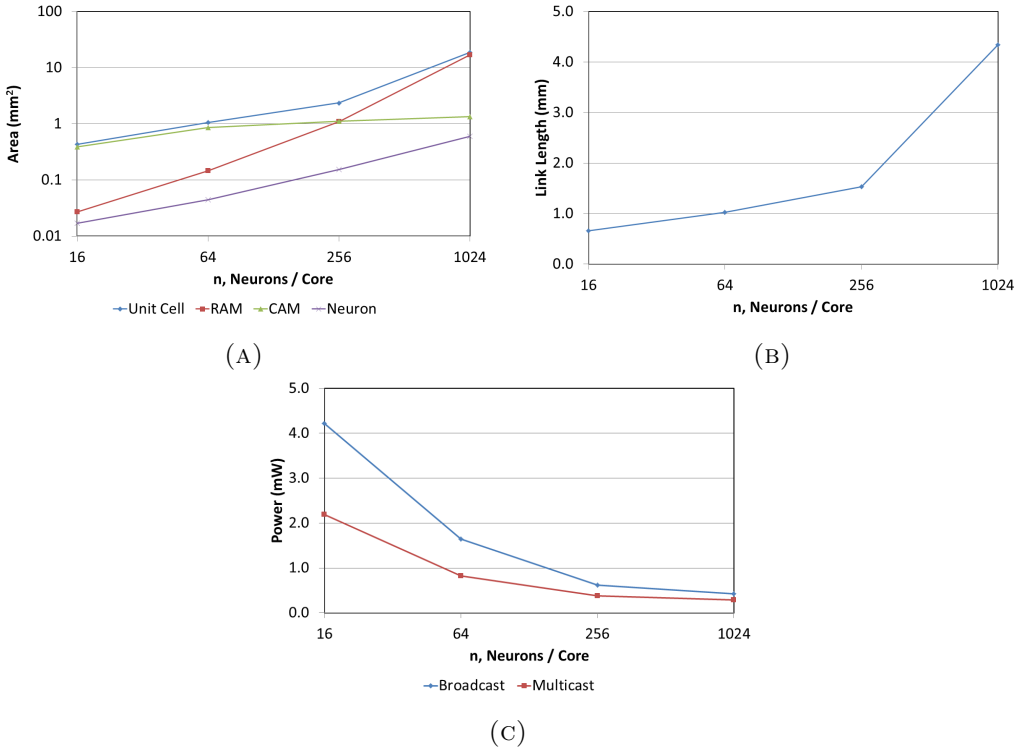


FIGURE 6.10: Energy consumed in the NoC interconnections. (A) Total area of unit cell and contributions of different components. (B) Mean link length of NoC interconnect. (C) Power consumed in interconnect links.

By combining the mean link length information with the bandwidth results provided in Figure 6.4 an estimate of the power consumed in the interconnect fabric can be formed. This is illustrated in Figure 6.10c. This figure shows that having a finer granularity, with only a few neurons per core, increases the energy consumed in the communication. However, the energy consumed in the communication is not of the same scale as that consumed in the memory. Vainbrand et al. [133] developed a theoretical model of the power in a neural-NoC and neglected to consider the energy consumed in the memory, which has been shown here to be dominant.

6.2.7 Combined

In order to fully evaluate the neural-NoC infrastructure the contributions of the RAM, the CAM, and the interconnect must be combined with the neural processing platform described in Chapter 4.

In Figure 6.11a the total area varies with differing granularity of processing core. As mentioned previously in section 6.2.3 there is an optimum granularity to reduce area

overheads. This is mainly caused by the intersection in the scaling of the memory components. ie. RAM is more efficient with a finer granularity, whereas CAM is more efficient with a coarse granularity.

There is a similar relationship with the power consumption as illustrated in Figure 6.11b. In fact, the optimal point is at the same level of granularity. Clearly, this leads to the conclusion that for this size neural network each processing core should implement 256 neurons, whether multicasting or broadcasting is used.

The area consumed by a broadcasting strategy is less than that of multicasting, as should be expected. However, multicasting is more energy efficient. In Figure 6.11b the increased energy efficiency is accentuated at fine granularities. This is caused by the dominance of the CAM units in the energy consumption at this granularity. In broadcasting, each packet is transmitted to every destination regardless of if a connection exists. This leads to at fine granularities many packets been transmitted to destinations where they are not required and the CAM tables still being searched, needlessly increasing the energy consumption.

The overall power/area relationship for a neural-NoC with the neural network described in section 6.1 is illustrated in Figure 6.11c. This graph clearly illustrates the detrimental effect that choosing the wrong granularity may have upon the final outcome of the system's performance, justifying the requirement to study the design parameters at an early stage.

6.3 Traffic Simulator

The previous section illustrated detailed analysis of the area and power performance of the designated neural-NoC. To clarify and investigate the design further a neural-NoC traffic simulator has been developed. This simulator has been developed with the intention of:

- confirming the theoretical performance parameters defined in section 5.3
- verifying correct performance of a system with the technological design parameters defined by the empirical model from Figure 5.16

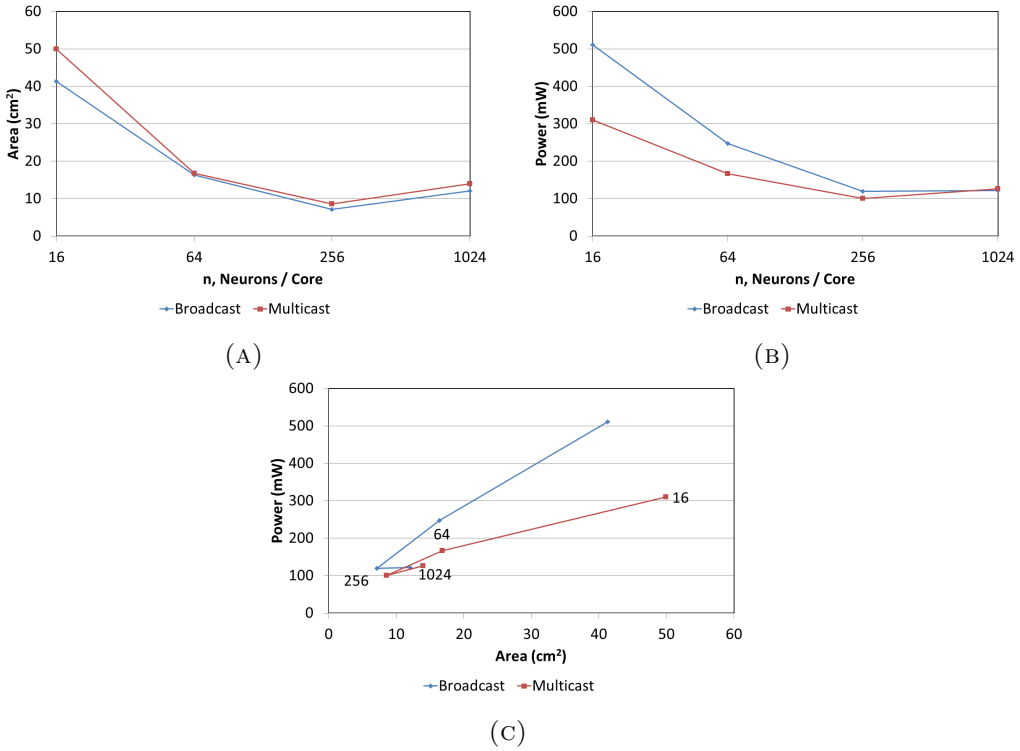


FIGURE 6.11: Overall area/power relationship for a neural-NoC. (A) Area varies with granularity of processing core. (B) Power varies with granularity of processing core. (C) The final area/power relationship with a varying granularity of system.

- evaluating the optimal design of sub-components of the neural-NoC
- studying the impact of the NoC infrastructure upon the computation of the neural network itself

Each of the above objectives are discussed further in section 6.3.3. Firstly however, the implementation of the software simulator is described in the following section.

6.3.1 Simulator Implementation

The simulator is developed in an object-oriented style using the Java programming language. It is a cycle-based simulator.

The simulator is packet based - meaning all transactions consider only the communication of a complete packet, as opposed to breaking packets into smaller units known as flits. Most NoC simulators do consider flit-based routing, as it provides performance benefits. However, these benefits occur mostly when considering very large variable

length packets. On the other hand neural spike packets are generally quite small and with a fixed size, as a packet only contains the identity of its source neuron.

Each generated packet must be routed from its source to its destination(s). It is shown in Figure 6.11, that for a neural-NoC broadcasting is comparable with a multicasting routing strategy. For simplicity, and because it is not commonly implemented in software simulators, it was decided to implement a broadcasting routing strategy. If correct performance can be verified with a broadcast strategy then a multicast strategy will perform even better.

The topology used by the simulator is a 2-dimensional flat mesh grid, with a variable size. This allows for a comparison of varying granularity to be undertaken. The simulator source code could be easily extended to include alternative topologies, such as a torus.

The simulator uses a simplified method of packet generation, as opposed to implementing a complete neuron model. Each processing core stochastically produces neural spike packets which are broadcast through the network. The rate of generation is controlled by a global parameter to the simulator. This allows for the steady-state performance of the network to be initially determined before characteristics of neural communication, such as burstiness, are considered. In order to study specific neural communication patterns a pre-defined list of spike events may be provided as a parameter to the neural-NoC simulatorl.

Within the software simulator each packet is represented as an object, references to these objects are passed between the various locations within the network, such as routers and buffers. Each packet object contains the time of its creation. When packets have been broadcast to all destinations their time is recorded, this information is used to calculate the latency of the communication. This time is the simulation time, which is related to the cycle of the simulator.

All generated packets are added to a global list. The throughput of the network can be calculated by searching this list to determine those packets that arrived at their destination within the allocated time period. The global packet list allows for the state of all the packets to be determined, including historic packets which may have completed traversing the network. If a global packet list was not kept object references to packets

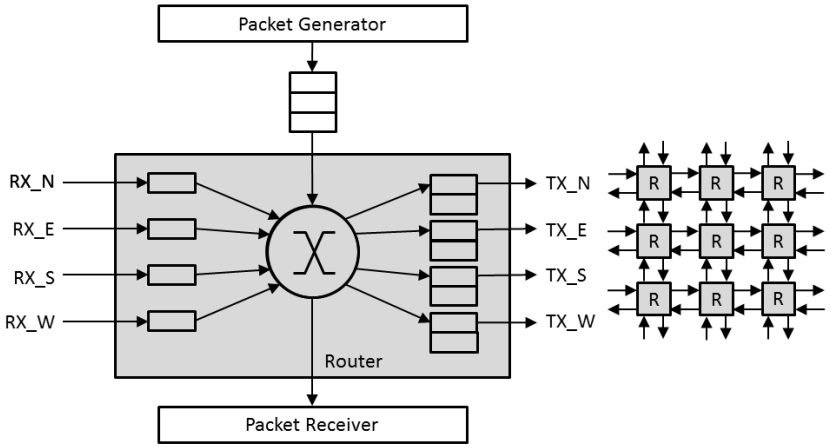


FIGURE 6.12: Design of router for the network traffic simulator. Each output port has a buffer with a variable depth. Each input port has a single word buffer. Packets are stochastically generated by the packet generator and held in a source queue until they are able to be transmitted. The switch randomly selects a packet from an input port to move into the correct output ports.

may be lost when they have finished traversing the network - making it difficult to determine the throughput and latency of the network over an extended period of time.

If packets are unable to progress through the network any further they may be dropped. The network implements a lifetime protocol, whereby if a packet becomes stuck and its age is beyond the allocated lifetime the packet is dropped. Dropping of packets prevents the network from being deadlocked, however, consideration must be made of how to implement dropping packets efficiently in hardware.

The main component of the network simulator is the router. This component manages the transmission of packets from their source to their destination. The router contains four input ports and four output ports, whereby an output port of one router is directly connected to an input port of a successive router. Each input port contains a single buffer. If an input buffer is full then the proceeding output port will be unable to transmit. The router's switch will attempt to move the contents of the input buffer to the required output buffers if there is available space. The size of the output buffer is variable and is set by a global parameter to the simulator.

The design of the router and how the routers are interconnected into the network is illustrated in Figure 6.12.

When a packet is generated, which happens stochastically, it is transmitted in a single dimension first, be it a row or a column. When the packet reaches each router along that

dimension it is duplicated and transmitted along the perpendicular direction, thereby reaching all routers within the network. Each packet contains an identifier for the router to determine whether to to switch the packet along the perpendicular direction. This is the process illustrated in Figure 5.7b. Packets are randomly selected to transmit first along the row or the column dimension of the network.

Each packet is assumed to be transmitted between two routers within a set time period which is a global parameter of the simulator. The arrangement of the packet transmission is not considered. For instance, a 32-bit packet could be transmitted in hardware in parallel across 32 wires at 1MHz, across 1 wire serially at 32MHz, or in series-parallel across 4 wires at 8MHz.

Each run of the simulation first of all proceeds through a warm-up state, whereby traffic levels and patterns reach steady-state performance. Next, the simulation enters the measurement state where the key characteristics of the network are measured before entering cool-down where packets generated in the measurement state are still considered whilst maintaining a background level of traffic. This protocol is specified by Dally et al. [171].

Each router and the state of each packet is updated with a fixed global clock. To simulate the nature of a network-on-chip the ordering of updates of routers and the switching with routers was randomized. This removes the reliance of the traffic results on the simulator implementation by preventing the system becoming deadlocked through a fixed-ordering update scheme and uneven access to shared resources. Implementing a random order of updates at each simulation cycle ensures a fair access and ensures no simulation artefacts are present within the output. Random ordering effectively replicates a completely fair arbitration scheme.

If the updating of routers was done in a fixed-order then the first routers to switch would gain a priority within the network, as they would move their packets to the destination first. The last routers to switch are therefore more likely to experience bottlenecks because a majority of the packets have already been transferred, which may have resulted in filling internal buffers. With a fixed-order update scheme, the last routers to switch will always be the last routers, and therefore they will deadlock first. By mixing the ordering of updates at each simulation cycle no router gains priority and the traffic distribution and access to resources becomes uniform. It is an important

```
Driver -f 10,100 -t 125 -k 16 -n 65536 -b 4 -l 50 -r 0.01 -s 0.0001 -e 0.0025
```

LISTING 6.1: Sample instruction to instantiate and run the network traffic simulator

```
------------Neural NoC Simulation---------------
Settings:
Grid Size:  16
Neurons:      65536
Neurons/Core: 256
Phit Period:  125ns
Firing Rate:  10Hz
Max Life:     50
Buffer:       4
Runtime:      0.01s
Steps:        80000
Time    Packets Meas.   Lat.    Dropped Thr.
4000    319     319     22.96   0       1.00
8000    644     325     23.10   0       1.00
12000   1003    359     23.05   0       1.00
16000   1358    355     23.05   0       1.00
20000   1680    322     22.97   0       1.00
24000   2024    344     23.27   0       1.00
28000   2371    347     22.92   0       1.00
32000   2690    319     23.02   0       1.00
36000   3072    382     23.22   0       1.00
40000   3397    325     23.22   0       1.00
44000   3723    326     23.06   0       1.00
48000   4080    357     23.21   0       1.00
52000   4386    306     23.13   0       1.00
56000   4739    353     23.01   0       1.00
60000   5055    316     23.07   0       1.00
64000   5382    327     23.02   0       1.00
68000   5680    298     23.17   0       1.00
72000   6004    324     23.17   0       1.00
76000   6307    303     23.15   0       1.00
No. of Packets: 6659.0
No. of Measurement Packets: 1624.0
Mean Latency: 23.06527093596059
Max Latency: 31.0
Dropped Packets: 0
Throughput: 1.0
```

LISTING 6.2: Sample traffic simulation output

design consideration when implementing a neural-network on chip to ensure that the routing strategy is fair, i.e. equal access to shared resources must be provided.

The software simulation can be instantiated from the command-line whereby it expects a list of parameters. Multiple parameters may be defined to run a test over a defined range of values. For instance, the command line instruction highlighted in Listing 6.1 will run the simulation twice, once with a mean firing rate of 10Hz, and once with a mean firing rate of 100Hz. A sample output of the traffic simulation is illustrated in Listing 6.2

TABLE 6.1: Theoretical limits for maximum mean firing frequency and minimum latency as illustrated on Figure 6.13. f_n represents the mean operating frequency of the network-on-chip and is set at 8MHz, s represents the number of neurons within the network and is set at 65536.

Grid Size	Maximum Mean Firing Freq. (Hz)	Minimum Latency (Clock Cycles)
k	$f_{s_{max,broad}}$	$H_{0,broad}$
	$\frac{2f_n}{s}\frac{k}{k+1}\frac{k}{k-1}$	$\frac{3}{2}k-1$
4	260	5
8	248	11
16	245	23

6.3.2 Simulator Results

The initial results of the simulator are shown in Figure 6.13 for three different NoC granularities. As can be seen, the output of the simulator corresponds with the theoretical values of latency and throughput calculated in section 5.3 and as expected the maximum throughput of the NoC decreases with an increasing number of processing cores.

The packet injection rate is defined as the ratio of packets injected to the highest theoretical throughput limit for a broadcast mesh NoC, as defined in (5.27).

By taking inspiration from calculating the effectiveness of solar cells using a "fill factor" [209] methodology it is possible to determine the performance of each network topology in relation to each other and also their theoretical limits. In photodiodes the fill factor is calculated as the ratio of the maximum obtainable power to the product of the open-circuit voltage and the short-circuit current [209]. A photodiode with a greater fill factor has fewer losses and therefore a greater performance.

The fill factor of a network can be determined in a similar way; as a ratio between the maximum product of achievable throughput and latency to the product of their theoretical limits. The calculated fill factor for the developed simulator decreases slightly with increasing grid size, suggesting that at smaller grid sizes the network is able to perform closer to the theoretical limits.

The initial results illustrated in Figure 6.13 show the network performance with no lifetime protocol. If a packet gets stuck it remains in its buffer until it can progress as no packets are dropped. This means the network may reach a point whereby no more packets are able to be transmitted as all buffers are full and the network becomes deadlocked.

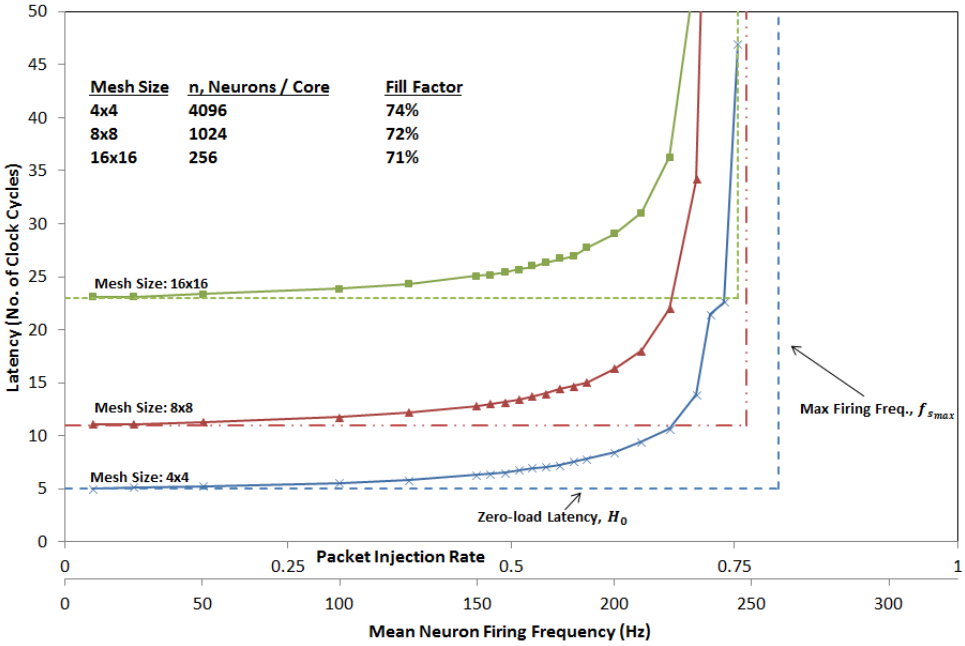
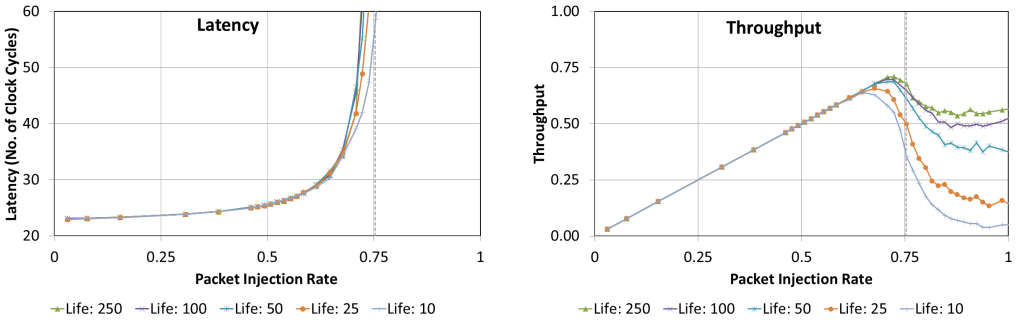


FIGURE 6.13: Latency versus packet injection rate for three different NoC granularities. The software simulator corresponds closely with the theoretical predictions provided by equations in Chapter 5. See Table 6.1 for derivation of theoretical limits.

6.3.2.1 Packet Lifetime

In order to overcome this scenario the simulator design is extended to allow for packets that have been deadlocked for a considerable length of time to be dropped, freeing up resources and allowing the network to progress. The effect of different lengths of time before packets being dropped can be seen in Figure 6.14. With a lifetime of 100 cycles, a packet will be dropped if it cannot progress and is older than 100 cycles. This allows for the network to partially recover, as illustrated in Figure 6.14b. It appears that a shorter lifetime will significantly diminish the throughput of the network, whilst a longer life cycle will allow for a more stable throughput.

Although the network is able to partially recover whilst dropping packets there will be significant distortion introduced into the computation of the neural network. As such, it will be preferential to avoid the network running under heavy loads continually. The implications of a lifetime protocol in terms of hardware are discussed in section 6.3.3.



(A) (B)

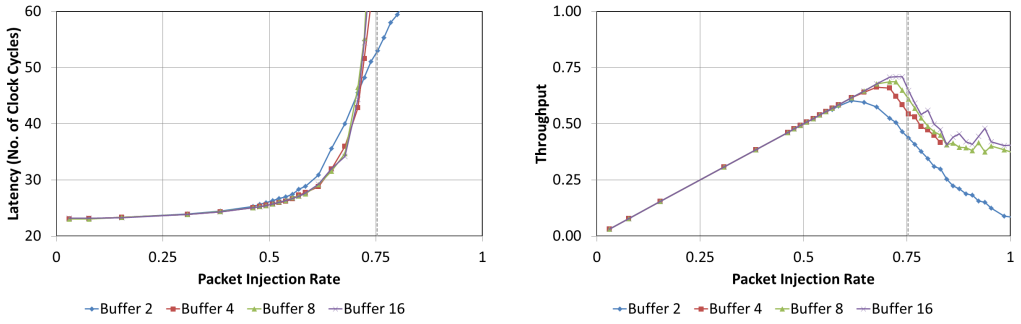
FIGURE 6.14: Investigating the impact of maximum packet lifetime upon the latency and throughput characteristics of neural-NoC. The "life" is how many clock cycles a packet may be stalled before it is dropped.
Neurons=65536, Grid Size=16x16, Buffer Depth=8

6.3.2.2 Buffer Depth

The latency and throughput characteristics are also reliant upon the depth of the buffers within the routers. A small buffer increases the probability that a packet may be delayed causing a backlog of traffic and increasing the number of packets that take longer to traverse through the network. However, buffers are expensive in terms of area resources [194] and large buffers should be avoided when possible.

The throughput and latency characteristics with a varying buffer size are shown in Figure 6.15. The mean latency does not increase as rapidly with a small buffer, however, more packets are dropped and therefore there is a weaker throughput. There is little deviation in the latency and throughput characteristics once the buffer depth increases above 4 suggesting this to be an optimal size.

Increasing the buffer size does not have a dramatic effect on the overall throughput and latency characteristics. A leaky bucket analogy can be used to describe this [171], where the buffer depth equates to the size of the bucket. Increasing the size of the bucket does not have any effect upon the rate that the bucket leaks. The minor increase in throughput with an increased buffer size can be linked with the bucket taking longer to fill, such that fewer packets are lost when the bucket overflows.



(A) (B)

FIGURE 6.15: Investigating the impact of buffer depth upon the latency and throughput characteristics of neural-NoC.
Neurons=65536, Grid Size=16x16, Life=50

6.3.2.3 Operating Frequency

Equation (5.27) derived that the maximum throughput grows linearly with the operating frequency of the network. This is confirmed in Figure 6.16 where the maximum achievable mean neuron firing rate for a varying operating frequency increases with operating frequency. By extracting this linear relationship it can be calculated that a neural-NoC will be able to sustain a mean firing rate of 8kHz with a router operating frequency of 1GHz. However, this mean firing rate is far beyond what is typically required and as such, this extra performance opens up the possibility of:

- running the network at a lower frequency to conserve power
- running the network at maximum frequency to model neural networks in faster than real-time. Currently 1 second of simulation/operation time would represent 1 second of biological time. However, increasing the frequency of operation could potentially simulate 1 second of biological time in significantly less than 1 second. This concept is used in the next chapter, where 1 second of biology is simulated in 25ms of real time.
- using serial links at high frequency between routers to reduce wiring area consumption.

6.3.2.4 Traffic Distribution

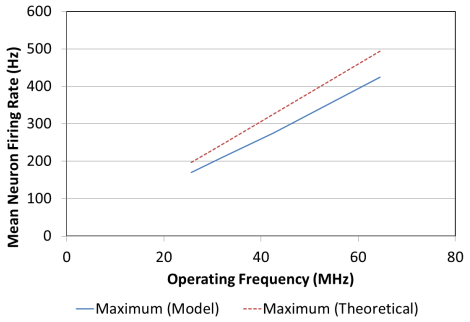


FIGURE 6.16: Comparing maximum mean neuron firing rate with network-on-chip operating frequency.

In section 5.3 it is stated that the throughput of a network is limited by the bandwidth that will saturate the bottleneck channel. In a broadcast mesh NoC this bottleneck channel is any channel transmitting data to the outer layer of the mesh. This is caused by in broadcasting every core receiving a copy of every packet transmitted. But, there is an uneven distribution of what channels each core receives packets upon. For instance, cores at the top of the mesh will receive the majority of their packets from the southern channels, whereas their northern channels will receive few, if any, packets. Therefore, these cores have some channels which are more heavily laden with traffic than other channels, and as such they are more likely to drop packets first when the traffic increases. This is why we see packets being dropped towards the edges of the mesh.

This hypothesis is confirmed analytically using the software traffic simulator, as illustrated in Figure 6.17. This figure shows a 16x16 core mesh. Each core is divided into four segments, with the colour of each segment representing the traffic level for a particular input channel to that core.

This uneven packet distribution suggests that a homogeneous router design is insufficient, and instead, the resources of a router should be skewed towards the links that are under the greatest burden. For instance, the channel buffers could be of different sizes inside each router or the router may give priority to certain channels.

Alternatively, the link width or the link frequency could be adapted such that links under a heavy load are wider/faster than those under a lighter load. This would allow for faster packet transmission and thereby reduce the likelihood of collisions and loss of packets in the outer layers of the mesh.

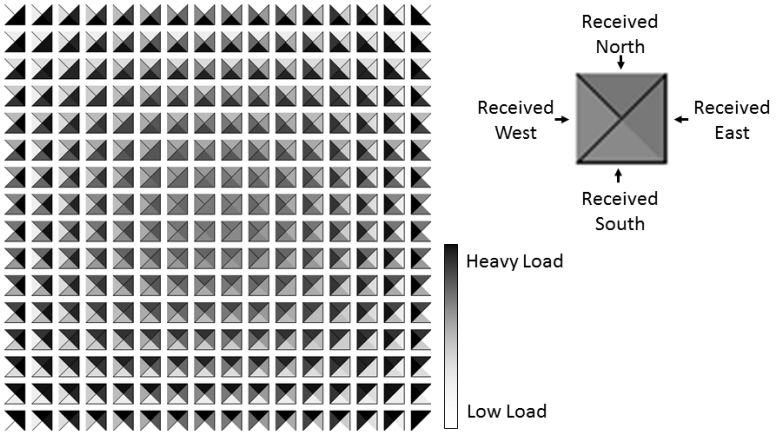


FIGURE 6.17: Channel load distribution through the mesh.

The uneven distribution of load throughput the network impacts upon the location that packets are dropped. Within Figure 6.18 the router locations within the mesh that packets are dropped are shown. By having a heterogeneous router design, whereby the outer routers are able to cope with higher traffic levels they will avoid becoming bottlenecked, and as such, the network will drop fewer packets. In order to accommodate greater traffic levels the routers must possess links with a higher bandwidth, either through a wider link or a link with a higher frequency.

6.3.2.5 Impact upon Computation

As mentioned in section 4.2, the foundation of a neural network's computation is based around the communication of the spikes between neurons. It is therefore imperative that a silicon neural network platform does not interfere with this communication pattern. Ideally the platform should deliver the spiking information in a deterministic way to avoid introducing unknown elements into the neural network's computation.

Within this section the developed traffic simulator is utilized to study the impact of the network-on-chip structure upon the neural model results. Three different encoding schemes are considered, rate-, temporal-, and correlation- based encoding.

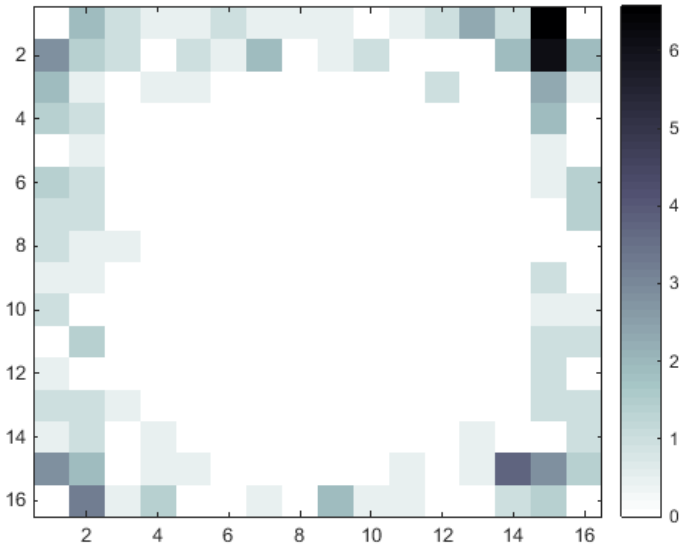


FIGURE 6.18: Location of packets dropped within a 16x16 mesh. Colour represents percentage of overall packets dropped.

Rate-based Encoding To study the performance of rate-based encoding upon the proposed neural-NoC the traffic simulator was provided with a pre-defined list of spike event packets. These packets were transmitted from their source to a destination node and the number of arriving packets within a defined time period were counted. The experiment was completed multiple times with a varying level of background traffic. The expected packet rate was then compared with the achieved packet rate using covariance correlation coefficient. The results can be seen in Figure 6.19. At low levels of background traffic the achieved packet rate is closely correlated with the expected packet rate and all packets are transmitted in under 100 clock cycles. However, at background traffic levels close to the threshold the correlation drops as some packets are lost and others are held in transmission for a considerable number of clock cycles. Figure 6.20 compares the effect of packet injection rate upon the correlation of the rate-based encoding for three different grid sizes in. Clearly, to maintain a high level of correlation the packet injection rate must remain below 0.5.

Temporal Encoding For latency based coding Figure 6.13 shows how below a packet injection rate of 0.5 the mean latency remains close to the theoretical minimum. However, this mean latency calculation may hide important characteristics of the traffic patterns. For instance, how does the latency of a neuron-to-neuron communication change

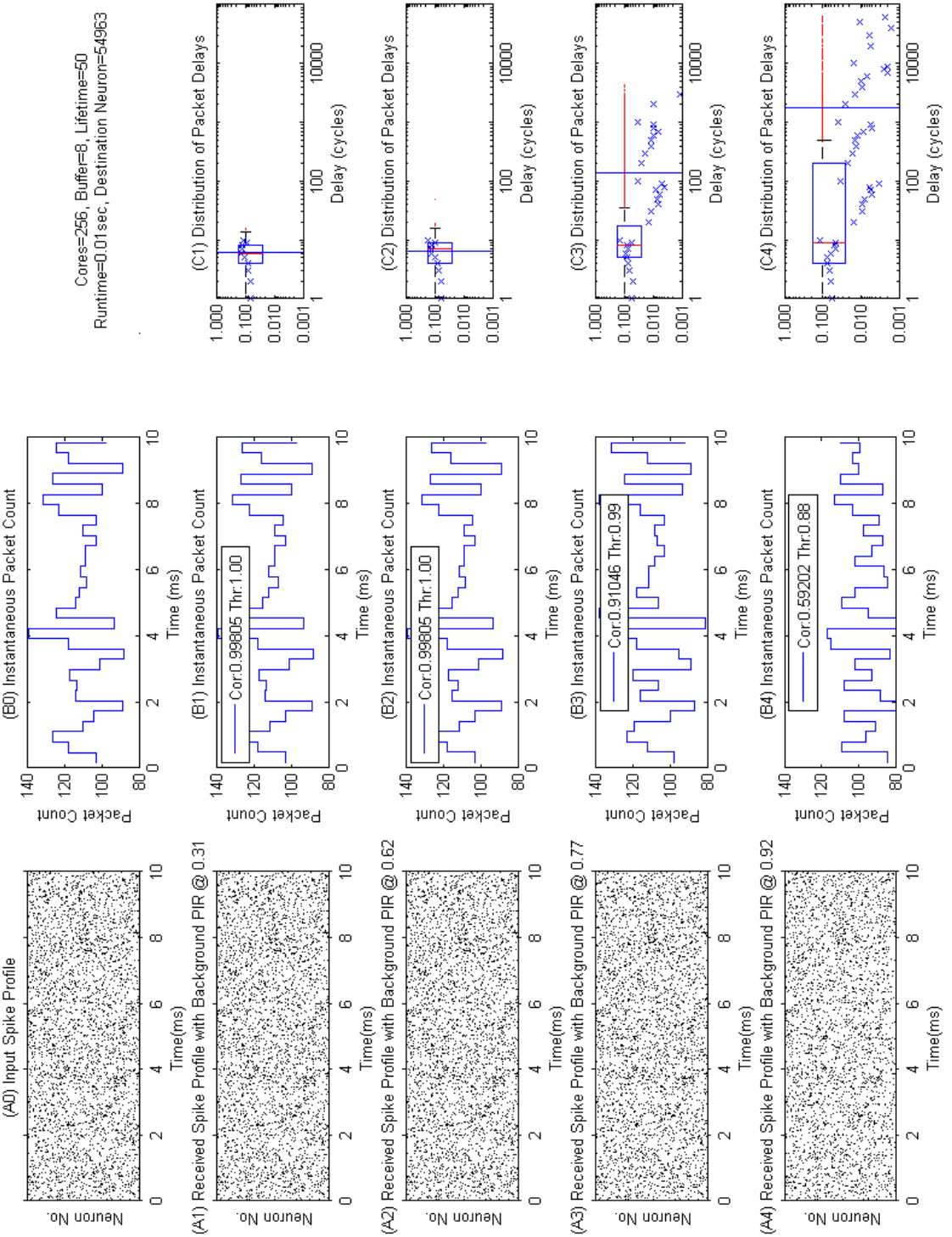


FIGURE 6.19: Studying the effect of background traffic upon rate-based encoding of neural spikes. (A0) The initial spike profile. (A1-A4) The received spike profiles for different levels of background traffic. (B0) The initial packet count within a defined time period. (B1-B4) The received packet count. (C1-C4) Distribution of the delays within the spike packets. The 'x' represent the probability density function, a box-plot is used to represent the median and quartile ranges of the distribution. The top-to-bottom line represents the mean.

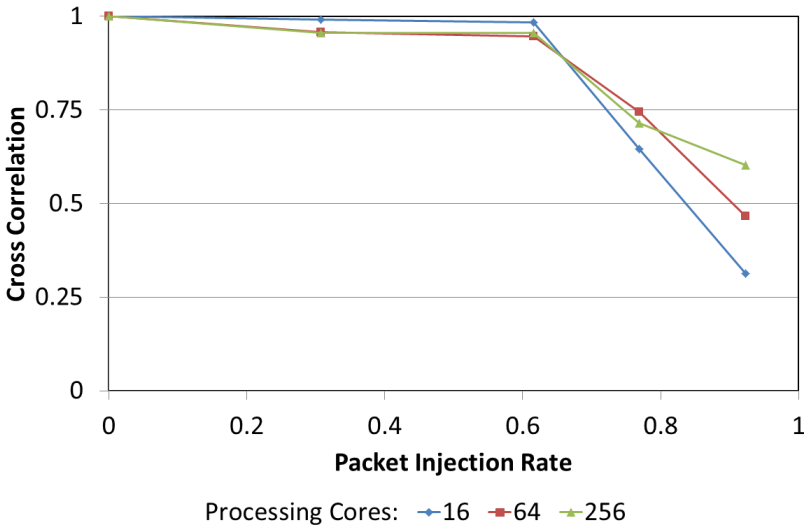


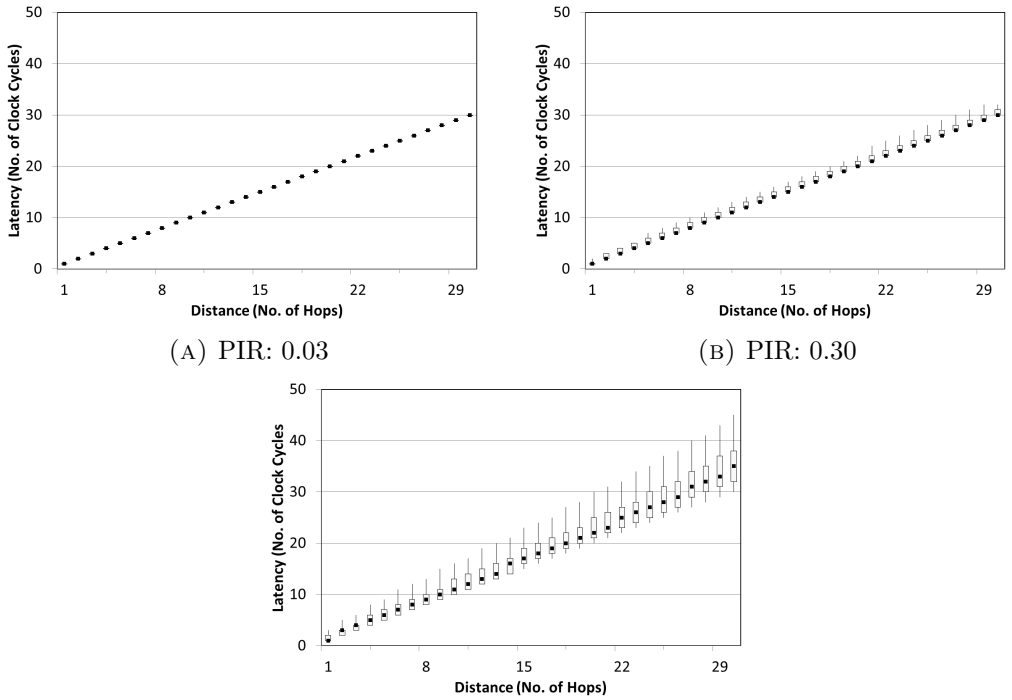
FIGURE 6.20: Correlation between received events and transmitted events for different background traffic levels for three different NoC sizes.

with the distance between neurons within the NoC. To study this the traffic model was executed with different levels of background traffic whilst tagging a proportion of the packets. Both how far these packets travelled and how long they took to reach their destination were measured and the results are shown in Figure 6.21.

For low traffic levels the latency is deterministic and is proportional to the distance between neurons. With increased traffic levels more variation is introduced into the latency as packets are held waiting in buffers. This suggests that if some neuron-to-neuron connections are temporally/latency related they should be located close to each other within the NoC and/or the NoC should be operated at low traffic levels to avoid variation in timings.

Correlation Encoding In correlation encoding information is related to the time difference between events. For instance, after the onset of a stimulus, two separate neurons may produce a spike that is transmitted to a third neuron. The third neuron may interpret the time difference between the arrival of the two spikes in some way. As such, there is a need to investigate the reliance of this time difference upon the infrastructure of the neural-NoC.

In a similar design to the previous experiment spike events were tagged with a time and a location and their progress through the network measured. The results of the correlation encoding experiment are in Figure 6.22. Within this figure, the relationship between the



(A) PIR: 0.03

(B) PIR: 0.30

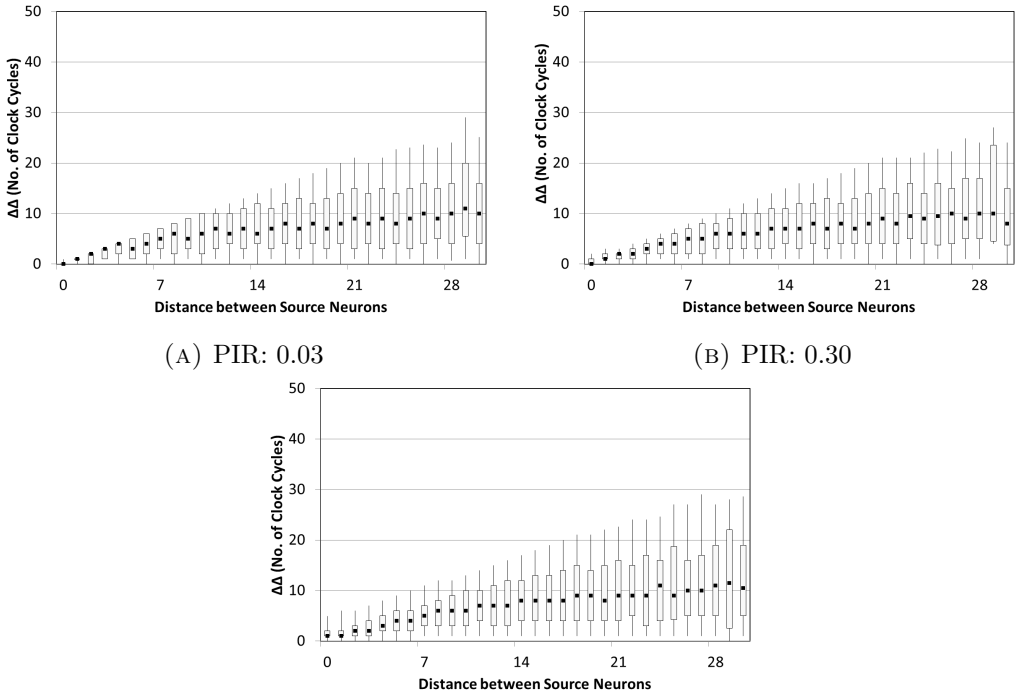
(C) PIR: 0.60

FIGURE 6.21: Relationship between latency of packet transmission and distance between two neurons for a varying level of background traffic

change in timing and the distance between the two source neurons for three different background activity rates is shown. The change in timing is defined as the difference between the difference in the spike arrival times and the spike generation times. As shown in this experiment there is less reliance upon the background activity rate and more dependence upon the distance between the neurons involved in the computation. Clearly, to minimize interference the neurons should be located close together or the network should be operated at a frequency that negates the error introduced by the delay in terms of clock cycles. For instance, a 10 clock cycle delay at an operating frequency of 1MHz equates to only 10us, which is unlikely to alter the computation of a neural network operating in the millisecond domain.

6.3.3 Simulator Discussion

In Figure 6.14b, it is illustrated how dropping of packets allows for the network to continue to operate under heavy traffic loads. However, in section 6.3.2.5 the negative effect of dropping packets and operating when the network is congested has upon the neural computation is demonstrated. Therefore, ideally dropping and losing packets



(A) PIR: 0.03

(B) PIR: 0.30

(C) PIR: 0.60

FIGURE 6.22: The effect of distance between neurons on correlation coding. $\Delta\Delta$ represents the change in the difference between the packet transmission time and the packet receive time.

should be avoided. By using the traffic simulator it can be proposed that operating the network below a packet injection rate of 0.5-0.6 will allow for correct neural computation.

The dropping of packets using the lifetime protocol utilized in the traffic simulator may also be very inefficient to implement. This is due to the requirement for each router to track time and for each packet to store a record of its time of birth. This could seriously hinder the design of the router and complicate the design of the internal buffers.

Equation (5.27) described the theoretical limit on maximum mean neuron firing frequency for a neural-NoC implemented upon a mesh structure with a broadcast routing strategy. From the traffic simulator it has been illustrated that the NoC can maintain deterministic and best case performance at up to 0.5-0.6 packet injection rate. Equation (5.27) can therefore be arranged to determine the minimum operating frequency of the NoC for a required neural network (6.1). This frequency should be used as the maximum time allowed for transmitting a packet between two routers.

$$f_n = f_{s,max} s \frac{k+1}{k} \frac{k-1}{k} \tag{6.1}$$

For example, for a neural network containing 100,000 neurons firing at a mean rate of 100Hz upon 256 processing cores an operating frequency of approximately 10MHz is required. This equates to a latency of packet transmission through the NoC of 2.3us and an equivalent NoC link throughput of 170Mb/s [2]. These speeds are well within what is currently achievable [132][118]. As such, for a neural-NoC the communication should not become the bottleneck of the design and there should be more emphasis on the efficient implementation of the memory and processing structures.

Adding extra buffer space may allow for bursty traffic to be temporarily dealt with, but it does not solve long-term congestion issues as described by the leaky-bucket analogy. Also, sharing of buffer resources should be avoided due to the technical overheads and complexity that this introduces [194] and the deterministic nature of the traffic distribution. Due to the deterministic nature of the traffic distribution buffer resources should be heterogeneously allocated throughout the system. For instance, a core in the corner of the mesh will require smaller output buffers than a core in the centre of the mesh as fewer packets will pass through it.

To overcome the uneven traffic distribution perhaps an original design choice of the traffic simulator should be altered - the mesh topology.

Broadcasting in a mesh results in congestion as the traffic spreads to all the four corners. If a torus topology was used the traffic would be allowed to spread more evenly resulting in less congestion. Each link would have an equal bandwidth and the maximum mean neuron firing rate would double as described in Table 5.3.

Initially a torus topology was discounted because of the increase in power and area within the interconnect fabric. But, as shown in section 6.2.6 the power in the interconnect is only a minor component of the overall system power and not the major contributor as proposed by Vainbrand et al. [133].

6.4 Discussion

The estimated VLSI characteristics of the system are provided in Table 6.2. The proposed neural-NoC is expected to consume 100mW of power and an area of $10cm^2$ for

[2] assuming a 17-bit packet length

TABLE 6.2: Expected performance characteristics for proposed neural-NoC.

Neural Network Parameters	
Size	65,000 neurons
Mean Neuron Firing Rate	10Hz
Mean Degree of Connectivity	1414
System Parameters	
Technology Node	65nm
Voltage	1.2V
System Performance	
No. of Processing Cores	256
Total Memory	2Gb
Memory / Processing Core	8Mb
Total Memory Bandwidth	18.5Gb/s
Total Communication Bandwidth	1.6Gb/s
Power	100mW
Area	$10cm^2$

computing 65,000 neurons in 65nm technology. This is just within the range of the specification outlined within Table 2.1. It is likely that with advancements in technology, particularly reduction in fabrication size, these power and area values will reduce, providing a more efficient and more capable platform.

The NoC system has been shown to not consume a significant proportion of the power and the area and the traffic simulator illustrates that a neural-NoC will be able to comfortably cope with the demand placed upon it. The current constraining factor within the design is the area and power consumption of the memory components needed to store the connectivity information of the neural network. This is due to the large degree of connectivity between neurons.

Currently, at the optimal granularity the memory consumes 90% of the power and 80% of the area of the proposed system. The CAM structure alone consumes 50% of the power. This is due to the large no. of searchers of the CAM and the significant energy consumed for each search. As illustrated by Table 5.4, in the last 10 years the energy/bit/search has reduced by 5-10x which is promising trend for this application.

Alternatively, to reduce the energy consumed within the CAM they could be restructured at a system-level. For instance, a multi-stage CAM could be introduced, which could allow only a sub-section of the CAM to be required to be searched.

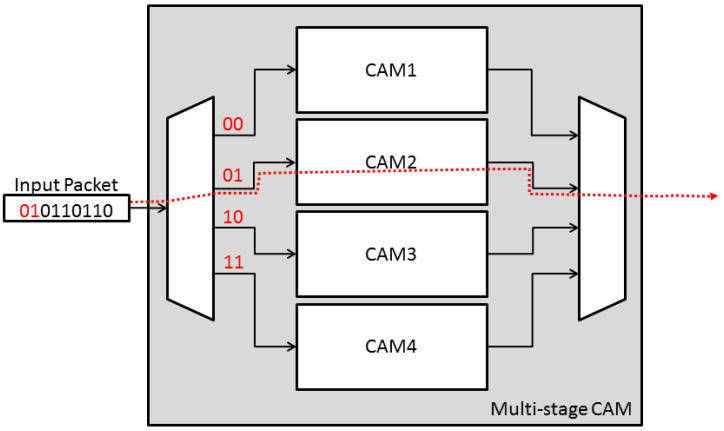


FIGURE 6.23: Multi-stage CAM method to reduce energy consumption. When a packet arrives a multiplexor selects only one of the sub CAMs to search. The selection is based around contents of the input packet.

This process is illustrated in Figure 6.23 where a single CAM block is partitioned into four separate CAMs. When a packet arrives a multiplexor inspects the contents of the package and then selects one of the four CAMs to search. Thus the three remaining CAMs are not searched and do not draw dynamic power. If this method was to be implemented care would need to be taken to ensure that each CAM partition was of equal size to ensure efficient layout. Also, investigations would have to be undertaken to determine the optimum number of partitions.

The study into granularity has shown that choosing the wrong granularity can significantly hinder the outcome of the design. It is proposed that for this neural network topology with 65,000 neurons that 256 neurons should be allocated to each processing, giving 256 processing cores within the network itself.

The optimal granularity is strongly correlated with the synaptic connectivity pattern. It is independent of many NoC design parameters, including topology and operating frequency, and as such, this granularity may be beneficial to multiple design groups with differing project objectives.

Recently, the development of systems with the same scale of processors as are required for this application have been demonstrated, highlighting that the design is feasible. Similar to the neural-NoC system most large multi-core systems suffer from a memory wall [3] and as such the development of 3D processor memory stacks has become popular.

[3]The memory wall is caused by the difference in processing operating speed and the speed at which it can access its required data from memory.

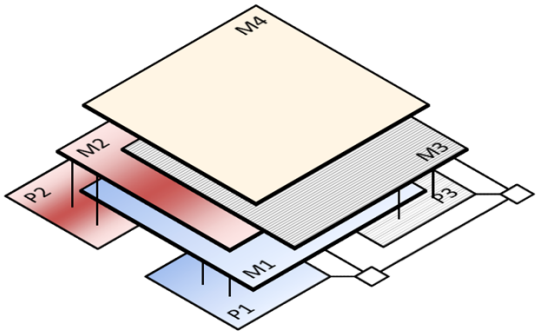


FIGURE 6.24: Potential 3D layout. The memories for each processing core are interleaved above the processing layer. This reduces the area consumption by 4x.

A review of recently demonstrated processor-memory stacks is provided within Table 6.3. As can be seen, the 3D-MAPS project team at Georgia Tech have shown a device with the same order of magnitude of processing cores and memory as are required for this neural-NoC design.

The projects listed in Table 6.3 do not meet the specification requirements of a neural-NoC in terms of energy consumption. However, a neural-NoC will be able to operate at a much lower operating frequency as the system will be designed to work within a biological time-scale. This should significantly reduce the energy consumption.

It is estimated that the area required by each unit-cell of the neural-NoC is $2.5mm^2$, of which $2mm^2$ is memory and $0.5mm^2$ is processing. Therefore, 4 processing cores could be placed within the area occupied by a memory required by a single processor. Hence, by using a 3D arrangement, 4 memories could be stacked on top of a single block containing 4 processing cores. Each processing core could be connected using through-silicon vias (TSV) to its memory upon an individual layer. This design concept is illustrated in Figure 6.24. This would have the effect of reducing the total area consumption by 4x to only $2.5cm^2$.

Typically, processor memory stacks allow for faster communication or reduced power consumption between processing cores due to the reduced link length between cores [215]. However, the neural-NoC is not constrained by the communication delay or energy consumption in the interconnect and it is likely any gains would be offset by the increased costs in routing information between the 3D layers using the through-silicon vias (TSV).

TABLE 6.3: Recent developments in 3D Processor-Memory stacks

Project/Institution	Year	Processing (No. of Cores)	Memory Size (Mb)	Memory Type	Memory Bandwidth (Gbps)	Power (W)	Technology	Chip Area (mm^2)	3D Layers	Reference
Hitachi	2010	8	8	SRAM	19.2	-	65/90	-	3	[210]
Hitachi	2011	16	1000	DRAM	1000	19.5	45	51	2	[211]
IBM	2012	-	176	DRAM	450	-	45	61	4	[212]
3D-MAPS V1, Georgia Tech	2012	64	2	SRAM	512	4	130	25	2	[213]
3D-MAPS V2, Georgia Tech		128	16000	DRAM			130	240	5	
Grenoble	2013	24		SDRAM	102		65	67.2	2	[214]

The separation of memory and processing into different layers would also allow for optimization of the fabrication technology for each process [216]. By optimizing the fabrication technology the area and power overheads, as well as the financial and manufacturing complications[216], could be reduced.

All large-scale systems have used packet-switching on multiplexed wires for neural communication at some level. However, previously implementations have often utilized alternative protocols for local communication [168][119].

If all m processing cores were directly interconnected instead of using a network approach then the system would require $O(m^4)$ in terms of wire length. For the 256-core system proposed this would equate to at least 873m of interconnect wiring as opposed to the 0.174m needed. This of course is also neglecting the complexity of routing such a large number of wires.

Alternatively, Seo et al. [120] proposed using a crossbar matrix approach. Table 5.2 describes how the area consumption of this design approach scales at $O(n^2)$ as opposed to a NoC scaling at $O(n)$. In practice, a 65000-neuron network would require a memory array of 4Gb accessed at a rate of 43Gb/s. Using a multi-core approach the total memory consumption required is reduced to only 2Gb, which is divided into 8Mb units each accessed at only 72Mb/s.

The memory architecture outlined within Figure 5.15 may result in sparse memory initialisation. For instance, within the 256 processing core system, each RAM cell contains 28k entries at 256 bits/entry for a total of 7.2Mb. But, the empirical model suggests that only 370kb of the 7.2Mb is initialised to 1 with the rest being set to 0. Therefore, instead of the approach in Figure 5.15 a sparse matrix representation providing a list of connected neurons could be used, as is illustrated in Figure 6.25. This would lead to 370k entries at only 8 bit/entry requiring only 3Mb. However, there is 10-15x more accesses required of the RAM. Therefore, despite the reduction in terms of area consumption the sparse representation format does require a greater power consumption. The comparison between these two techniques in terms of area and power is illustrated in Figure 6.26.

Figure 6.11 shows how broadcasting compares with multicasting for a neural-NoC in terms of area/power utilization. It is proposed that these techniques could be merged

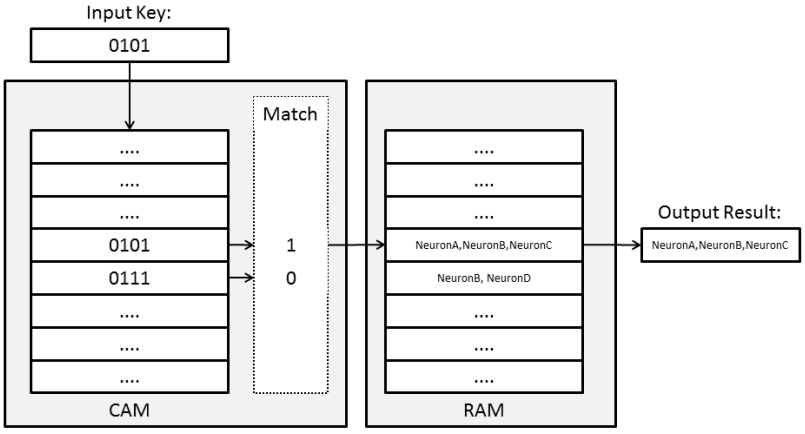
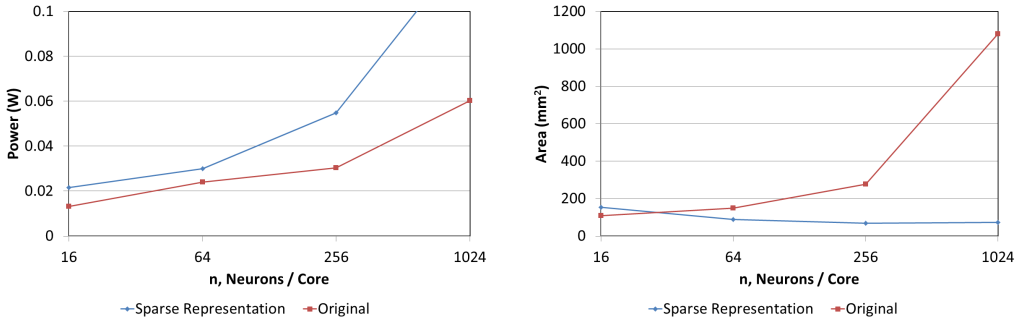


FIGURE 6.25: Illustration of sparse matrix RAM connectivity option. For each match in the CAM a list of connected neurons is provided within the RAM. Each item in this list will be a $log_2 n$-bit identifier if there are n neurons per core.



(A) (B)

FIGURE 6.26: Comparison in terms of area and power consumption between the original RAM methodology and a sparse representation alternative.

to form a narrowcast methodology. This will involve packets being initially broadcast to all destinations until they are prevented from travelling any further by a routing entry at a particular core. These routing entries will already exist if a destination neuron is upon the same core, which should always be the case. With initially broadcasting the requirement to store routing information is removed until the edge of the packets area of travel significantly reducing the area consumption by reducing the number of routing entries from multicasting. Also, it will prevent packets from being transmitted further than is required, which happens in a broadcasting scenario. In effect, narrowcasting will assume the area relationship of broadcasting with the power relationship of multicasting. This proposed routing methodology is illustrated within Figure 6.27 and the expected area/power is highlighted in relationship to broad/multi -casting in Figure 6.28.

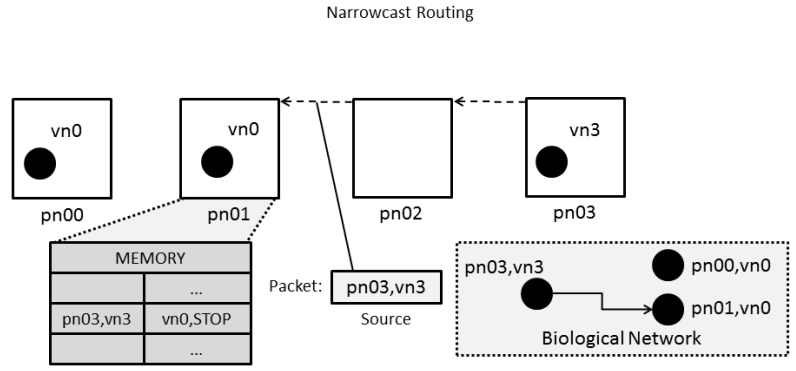


FIGURE 6.27: Narrowcast routing strategy. When the neuron $pn03, vn3$ produces a spike, a packet is broadcast through the network until it is instructed to stop travelling. Each processing core that receives the packet is responsible for filtering the messages so that only connected neurons are updated.

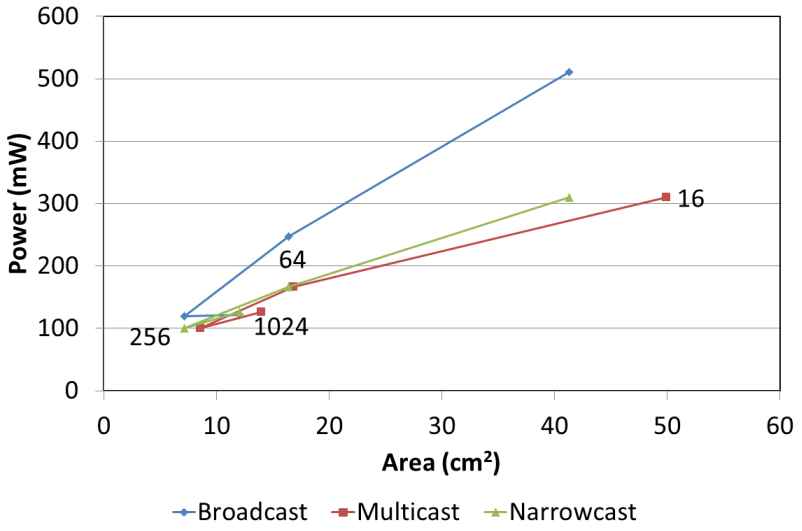


FIGURE 6.28: Narrowcasting area vs power relationship. Narrowcasting achieves the same power performance as multicasting but with the same area performance of broadcasting.

6.4.1 Recent Developments

In August 2014, IBM released their latest work within the neural network-on-chip (NNoC) field within the journal Science [217]. This paper describes a single chip capable of simulating 1,000,000 neurons and 256,000,000 synapses. There are key architectural similarities between the work of this thesis and [217].

Firstly, both designs utilize a granularity of 256 neurons per processing core, which has been shown within this thesis to be the most efficient.

TABLE 6.4: Comparison between proposed NNoC and NNoC implemented in [217]

Feature	This Thesis	[217]
Application	Neural Prosthesis	Object Recognition and others
Neurons	65,000	1,000,000
Synapses per Neuron	1000	256
Total Synapses	65,000,000	256,000,000
Processing Cores	256	4096
Granularity	256	256
Technology	65nm	28nm
Size of Memory (Gb)	2	0.4
Power (mW/cm2)	10	20
Area (cm2)	10	4.3
Learning	No	No
Routing Style	Dimension Order	Dimension Order
Router Design	5-port	5-port
Network Topology	Mesh	Mesh

Secondly, the network topology is identical. Both designs use a mesh structure with 5-port routers.

Finally, both designs use packets to transmit synaptic information between cores. In both designs these packets are routed between cores using dimension-order routing.

This IBM paper has shown that a design similar to that proposed within this chapter is feasible and practical. Although the IBM design does demonstrate more neurons in a smaller area this is compensated by the decrease in the number of synapses per neuron and a smaller technology size of only 28nm. Table 6.4 provides a comparison between the two design, clearly detailing the overall similarities.

Chapter 7

Neural Network Case Study 2 - Granular Layer Rehabilitation

"Data! Data! Data!" he cried impatiently. "I can't make bricks without clay"

The Adventure of the Copper Beeches, Sherlock Holmes

Sir Arthur Conan Doyle

The work in this chapter was done in collaboration with Jun Wen Luo who was responsible for the design of the central processing cores and undertaking the results described in section 7.4.5. However, the focus of this chapter is primarily upon the design and operation of the network-on-chip infrastructure which was the sole responsibility of the author.

In this second case study the methodology described in Chapter 5 is utilized to study the optimal design of a network-on-chip for the realisation of a cerebellar prosthesis. The proposed design is then implemented fully within an FPGA and results demonstrating its performance are provided.

Animal and human motor movements, such as walking, running or even riding a bicycle, rely upon the coordinated timing and activation of muscles. The cerebellum is the vital controller within this process and it is responsible for the encoding of time within the millisecond range, known as passage-of-time encoding (POT) [218]. Incorrect encoding within the timing of movements may result in dysmetria, a condition caused by damage

to the cerebellum and which results in a patient being unable to control body movements reliably [219].

It is proposed that a neuroprosthesis can be created that will allow damage to the cerebellum to be overcome. For this, an efficient computational platform that can mimic the complex function of the cerebellar neural network will be important.

The encoding of time within the cerebellum is believed to be controlled by the granular layer, that primarily consists of Granule and Golgi cells. Many models have been developed that replicate the function of the granular layer. The random projection model [220][221] is a popular, robust and plausible model and is therefore used as the foundation for this study.

Previously, implementations of this model have been completed using software [221], which are slow to process. Although a more recent GPU implementation exists [103] that performs faster, a more suitable platform is required for biological experiments. An FPGA platform with flexible input and output connections will allow for in-vivo experiments allowing further investigation into the biological features of the network. Lessons learned in the development of this platform may also be beneficial for a neuroprosthetic system. Previous hardware implementations of the cerebellum, such as a VLSI mixed-signal programmable array [74], have considered only simplified models of the cerebellum.

In the following section the random projection model is introduced, with a particular focus upon its network properties. This network is then evaluated using the previously described methodology to determine the optimal design parameters. Next, the implementation of the neural-NoC is described before providing results of the system's performance that illustrate how it can be used in a neural prosthesis.

7.1 Passage-of-Time Computational Model

The cerebellum granular layer consists of granule and Golgi cells. The model proposed in [221] contains over 1000 Golgi cells and 100,000 granule cells. The granule cells are collected within a granular cluster, which send excitatory inputs to a local Golgi cell. The Golgi cells are randomly connected in return back to the granule clusters. On

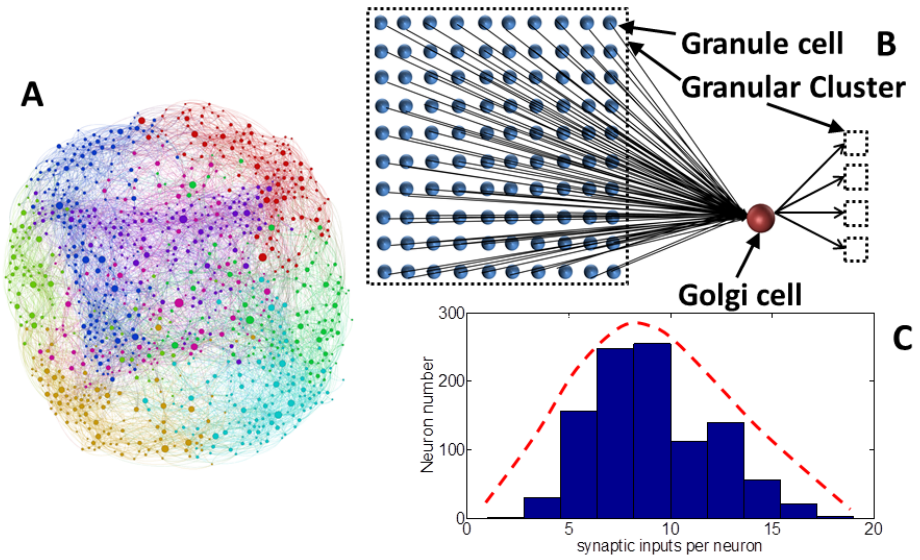


FIGURE 7.1: The network properties of the granular layer model. (A) The connectivity between cell units. The different colours represent the grouping of cell units. Image produced by *Gephi* graphing software. (B) Illustration of a cell unit. 100 granule cells make up a single granular cluster. The output of each of these cells is connected to a single Golgi Cell. The output of the Golgi cell is randomly connected to a selection of granular clusters. (C) The distribution of the number of connections between Golgi cells and granular clusters.

average, a Golgi cell is connected to only 8 granule clusters. This topology is illustrated within Figure 7.1.

Each granule and Golgi cell is modelled as a conductance-based, leaky integrate-and-fire unit. The development of these neuron models is described in [222][1]. For the development of the neural-NoC, the state of the neuron model can be simplified to a simple all-or-nothing digital pulse, generated if the cell produces an action potential. The Golgi cells activate at a mean rate of 40Hz.

The Golgi-to-granule cell transfer of information is studied using the network-on-chip empirical model described earlier. The empirical model is used to determine the optimum routing methodology and the optimum network-on-chip size to achieve the best performance. The network of Golgi cells is mapped to processing cores within the network-on-chip using the simulated annealing algorithm described previously.

Clearly, the properties of the neural network model described here differ greatly from the previous case study in terms of the number of connections. As such, the optimal

[1]The neuron models were developed by J. W. Luo, PhD Student, Newcastle University

design parameters of the NoC will be different to what was previously highlighted in Chapter 6.

7.2 NoC Analysis

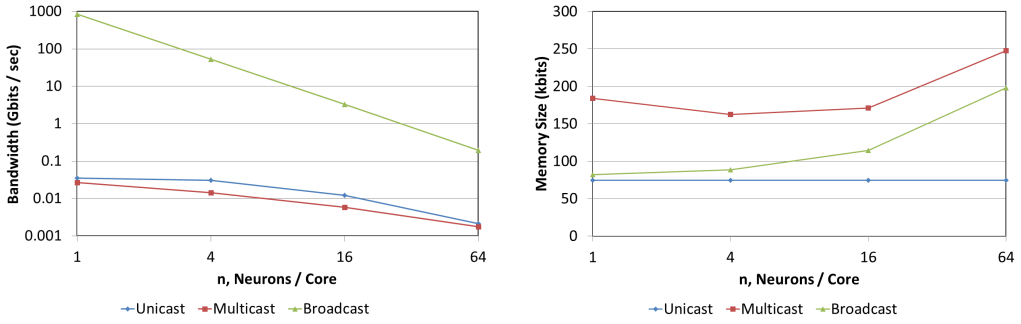
The regular structure between a cluster of granule cells and their nearest Golgi cell highlighted in Figure 7.1(B) allows some simplifications to be made to the neural-NoC design. Each granule cell may be updated sequentially and their output summated in order to form the input to a Golgi cell. This removes any requirement to transfer granule cell spiking information through a network, leaving only the Golgi-to-granule information.

All three of the standard routing methodologies described in section 5.3 have been analysed with the granular model to determine the system bandwidth, the memory overheads and the power and area requirements.

In Figure 7.2a, the bandwidth for each routing scheme is compared with the number of Golgi cells per processing core. Having more Golgi cells per core means the NoC can be smaller and therefore in all instances the bandwidth reduces.

In the previous case study, broadcasting performed similarly to multicasting, but with this network model it performs much worse. This is caused by the each Golgi cell being connected to only 8 granule clusters on average as opposed to the previous example where the connectivity was greater than 1000. As the connectivity is very low packets are only required to be transmitted to a minority of the destinations within the chip. Multicasting offers a reduced bandwidth in comparison to unicasting, although only by a factor of 2.

Figure 7.2b illustrates the size of the memory tables. For unicasting, the size of the memory tables required remains constant for each granularity of processing core. At fine granularities broadcasting offers a similar level of memory size, but this increases greatly as the number of neurons per core grows. For multicasting, the burden imposed by the requirement to store extra routing information can be clearly seen in the difference between it and broadcasting memory sizes. Multicasting requires approximately 2x as much memory as unicasting - an inverse of the bandwidth relationship.



(A) (B)

FIGURE 7.2: Comparison of the bandwidth (A) and memory requirements (B) for three different routing strategies implementing the granular layer model upon a neural-NoC for a varying level of processing core granularity.

Bandwidth and memory size can be used to determine the power and area relationships of each proposed design as described in the methodology. This is illustrated in Figure 7.3a. Due to the simpler communication requirements of the granular model the neural-NoC performance becomes much more reliant upon the processing platform. Therefore, as shown in this figure, the power versus area relationships remain similar for all three different routing strategies. The contribution of the processing, the communication and the memory towards the overall power consumption for a unicast routing startegy is highlighted in Figure 7.3c

However, by removing the computational contribution to power and area it is possible to analyse the effect of the different routing mechanisms upon the power and area relationships, as highlighted in Figure 7.3b. It is clear in this figure that unicasting provides the best performance characteristics. Multicasting requires a similar level of power consumption but a far greater investment in area due to the CAM/RAM requirements. The significantly increased level of bandwidth in broadcasting means that it will result in approximately 2 orders of magnitude increase in terms of power consumption.

In Figure 7.3b, the effect of granularity upon the power and area consumption in the communication infrastructure is highlighted. For unicasting a very fine granularity, with a minimum number of neurons per core provides the most efficient platform.

Also, by reducing the number of neurons per core the update of that core can be completed in less time. A large number of smaller cores can complete the same computation quicker than a small number of large cores. This is beneficial for the granular layer

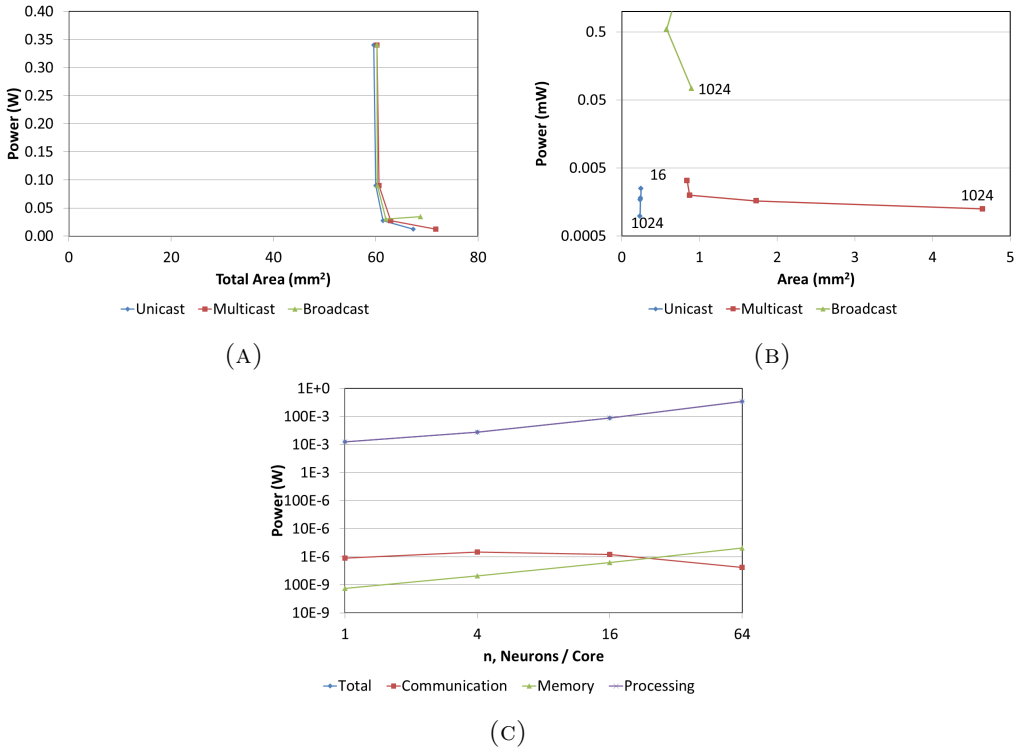


FIGURE 7.3: Power and area relationships for a neural-NoC implementing the granular layer model. (A) The power/area is dominated by the processing platform and as such, the different routing methodologies do not have a great impact. (B) Without the processing platform included the power and area of each routing method can be compared. Unicasting is a clear favourite for this scenario. The numbers within the figure represent the number of processing cores- i.e the granularity of the system. Clearly, the granularity greatly impacts the communication overheads. (C) The contribution to the overall power consumption of different components of the neural-NoC.

neural-NoC platform as a secondary objective is to run large-scale simulations in accelerated time. Therefore, with a fine granularity a greater level of performance is achieved whilst also reducing the cost in terms of area and power consumption.

7.3 Design

For initial verification of the design concept and for use in exploratory investigation by neuroscientists an FPGA-based design is selected for the first stage in development. However, it is shown in the previous section how the design will comfortably scale to an ASIC platform that will meet the specification of a neural prosthesis device provided in Table 2.1 if required.

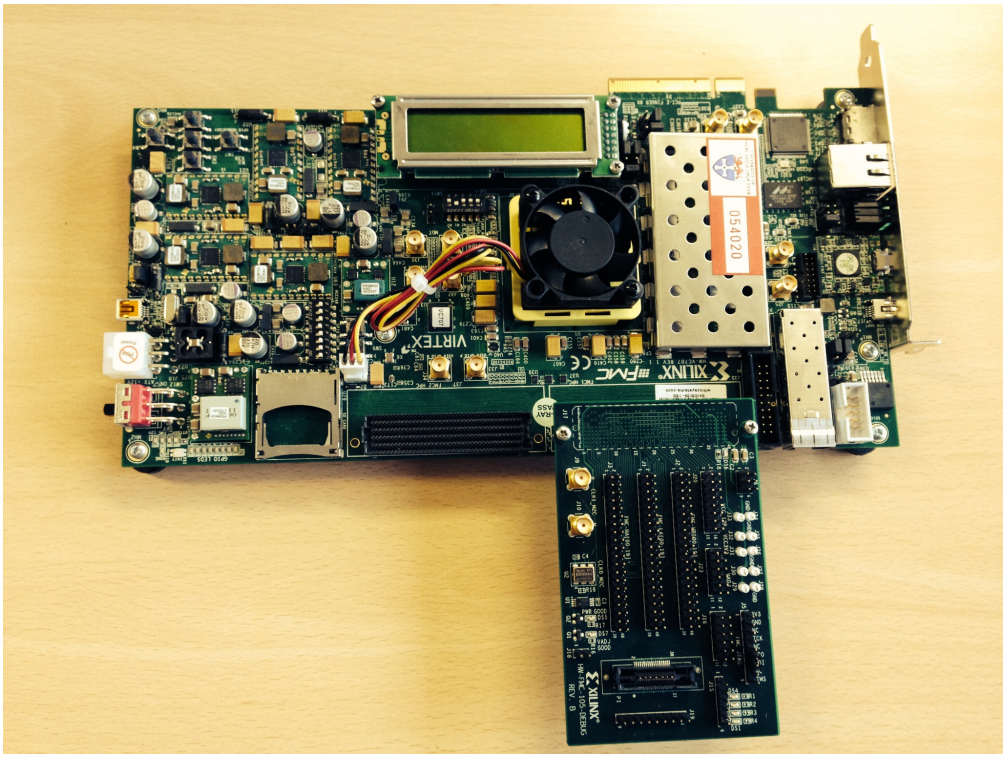


FIGURE 7.4: A Xilinix VC707 evaluation board containing a Xilinx Virtex-7 XC7VX485T FPGA.

The chosen FPGA is a Xilinx Virtex-7 XC7VX485T installed upon a Xilinx VC707 evaluation board. This FPGA contains nearly 500,000 logic cells, 2800 arithmetic units, 37Mb in on-chip block memory and 700 user I/O pins [148]. The evaluation board supplies all the required configuration circuitry alongside additional memory and input/output resources. An image of this setup is provided in Figure 7.4.

The neural processing platform, which implements the conductance-based, leaky integrate-and-fire model was developed by Jun Wen Luo and is described in [222]. As mentioned previously, a single Golgi cell receives the summated response from a single cluster of granule cells. It is the responsibility of the NoC to manage only the communication of events from the Golgi cells to their connected granular clusters. This process of communication separation is illustrated in Figure 7.5.

Each processing core implements 2000 granule cells and 20 Golgi cells. Ideally, each core would implement fewer cells to reduce the power and area consumption, as described in the previous section. However, the available resources upon the FPGA and the chosen core design methodology constrains the granularity to 20 cells. Each processing core requires approximately 50 arithmetic units, and there are 2800 arithmetic units

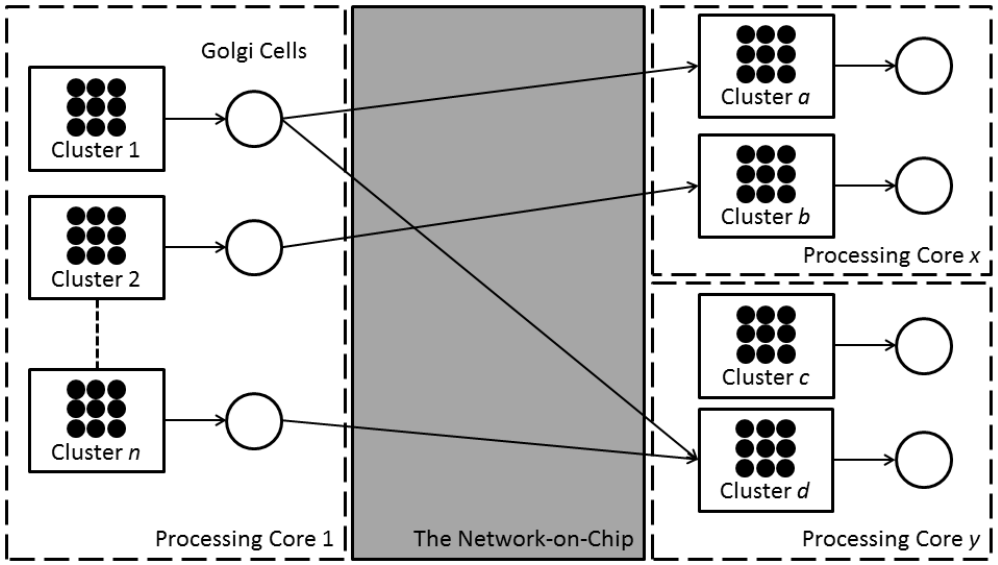


FIGURE 7.5: Illustrating the separate communication structures within the granular layer neural-NoC platform. Communication between granular clusters and Golgi cells is handled upon the processing core. Communication from Golgi cells to granular clusters is managed by the NoC infrastructure.

available upon the FPGA, meaning 56 processing cores available. The granular layer modelled contains 100,000 granule cells and 1000 Golgi cells, so with 50 processing cores each core is required to implement 20 Golgi cells. The NoC design however is not reliant upon the chosen granularity and the number of processing cores within the network is flexible.

A general overview of the neural-NoC design is provided in Figure 7.6. To implement 1000 Golgi cells, 50 cores are arranged in a rectangular mesh. Each processing core is attached to an interface module that is responsible for packetizing spike events from the core and inserting the packets into the network. It is also responsible for receiving the packets and buffering the inputs until required by the granule cells.

The interfaces are each connected to a router which inspects the contents of the packets to determine the directions of which to route the packets. The packets are transmitted between routers using a 4-phase handshake. The network itself is mesochronous, meaning each module operates at the same frequency but with an unknown phase difference [223].

A special processing core is attached to the network to provide an external interface. Packets can be sent to/from this core to inspect/update the status of the system. This special processing core is referred to as a listening module. A configuration module is supplied, which is responsible for communicating the connectivity of the network to the

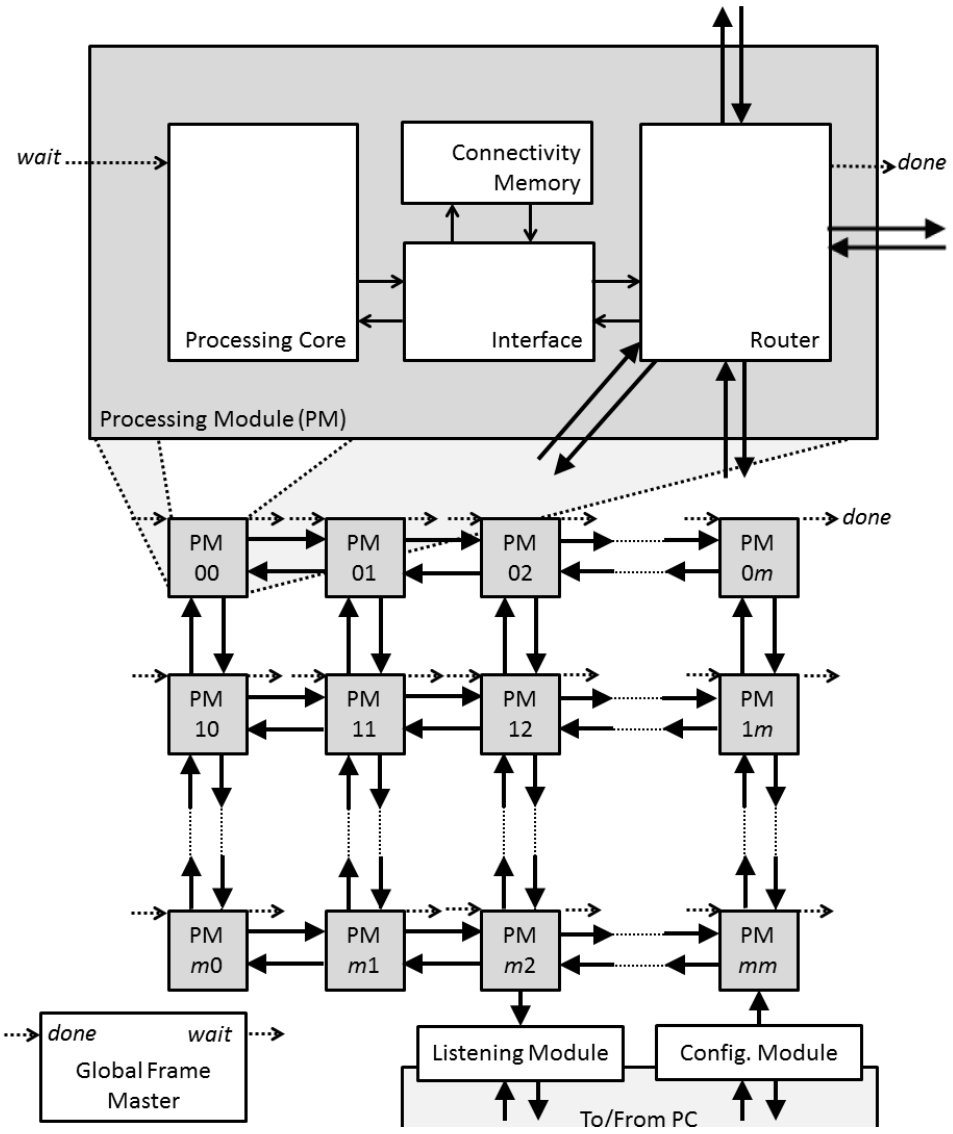


FIGURE 7.6: Granular Layer Neural-NoC Design Overview.

processing cores within the system. The global frame master is utilized to manage the timing and arbitration system.

In the following sections each of these components are described in further detail.

7.3.1 Time

The processing core updates a new granule cell on each clock cycle. Therefore, after 100 cycles a cluster of granule cells has been calculated and the response is fed to the Golgi cell. If there are n Golgi cells upon the core then in $n * 100$ cycles all of the Golgi cells

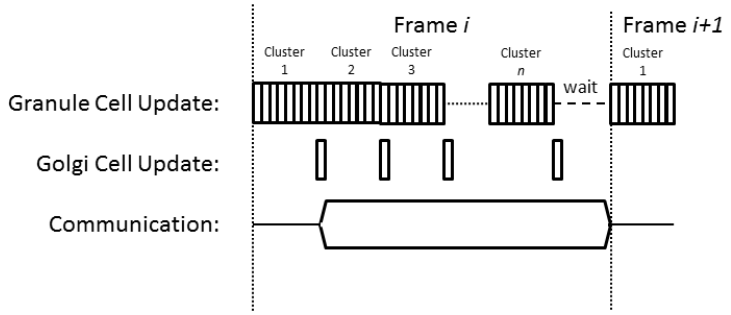


FIGURE 7.7: Illustration of frame-based encoding for the neural-NoC. Packets are communicated whilst the processing cores compute the updated states of the cells. If the communication of packets exceeds the computation time then the cores are paused until the communication is complete, then the next frame can begin.

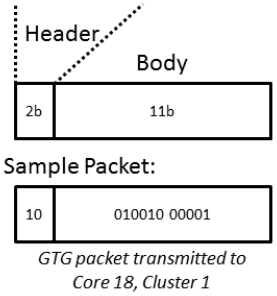
have been updated. This time is known as a processing frame and is equivalent to a single iteration within the software model.

This processing frame is managed globally throughout the chip using a single clock cycle from the global frame master. This is an identical approach to that of IBM in [217].

The communication of packets from Golgi cells to granular clusters happens concurrently with the computation. If packets are still travelling through the network at the end of a time-frame then the succeeding time-step is stalled until the network has finished transferring packets. This novel neural-frame-based encoding ensures that all packets are transmitted to their destination in the same frame as of which they are generated. On receipt the packets are buffered and utilized by the granular clusters in the succeeding frame. This has the benefits of: maintaining compatibility with the synchronous software model; removing timing errors caused by packet transit delays; and allowing for a coherent accelerated time mode of operation.

This frame-based process is illustrated in Figure 7.7.

With 20 Golgi cells per processing core the processing frame will require a minimum of 2000 cycles to complete. At an operating frequency of 50MHz this requires $40\mu s$. Each frame is equivalent to 1ms of biological time hence, there is a 25x speed-up over real-time. However, if the neural-NoC is required to operate in real-time for use in hybrid bio-electronic experiments the global frame-master could be programmed to stall the network such that each time frame is only initialised every ms.



Header	Type	Body Format	
		6b	5b
00	Golgi Event (GE)	Source Core ID	Source Golgi ID
01	Broadcast (BC)	-	-
10	Golgi-to-Granule (GTG)	Target Core ID	Target Golgi ID
11	Config. (CS)	Target Core ID	Config Info

FIGURE 7.8: Packet types and format. The 13-bit packets can be of four different types, which is denoted by the 2-bit header value.

7.3.2 Packet Format

The packet format is illustrated in Figure 7.8. In the current design packets are of a fixed 13-bit length. The two most significant bits are the packet header representing the type of packet. The remaining 11-bits make up the packet body. Packets can be of four different types: Golgi-to-granule (GTG); Golgi event (GE); configuration start/stop (CS); and broadcast (BC).

Golgi-to-granule The most common type of packet is GTG. This packet is produced when a particular Golgi cell produces a spike. On production of a spike the interface inspects the connectivity memory to determine a list of granular clusters that are connected to the spiking Golgi cell. For each connection a separate packet is generated. The 11-bit body of the packet contains the destination of the connected cluster. The first 6-bits of this body represent the processing core where the cluster is located and the remaining 5-bits represent the cluster upon that core. No identification of the spiking Golgi cell is attached to the packet as it is unnecessary for this particular granular layer model. Although, if the packet length was extended it could be easily included.

The routers use the 11-bit body to route the packets through the network to the correct destination. All GTG packets are expected to be transmitted within a single processing frame. If they are not, then the network is stalled until all messages are finished.

Golgi Event The GE packet is also produced when a Golgi cell produces a spike. This packet is used to inform the listening element that a particular Golgi cell has spiked. The listening element may communicate this information outside of the device if requested.

The packet body contains the 11-bit identifier of the spiking Golgi cell. The routers on inspection of the packet header route the packet towards the listening element.

Configuration The CS packets are used to setup the connectivity of the network. These packets target a specific processing core within the network. On receipt of the packet the interface module enters configuration mode. Any further packets received by this interface are then inserted into the connectivity memory in the order that they are received overwriting the previous contents. When a second CS packet is received by the interface it leaves configuration mode and resumes its normal tasks. The 6 most significant bits of the packet body represent the target processing core. The configuration process is outlined in further detail in section 7.3.10.

Broadcast Broadcast packets are transmitted to all destinations within the network. The contents of their body change depending upon the reasoning behind the transmission of the packet.

7.3.3 Interconnection Link

The interconnection link is responsible for managing the transmission of packets between two routers. The design utilizes a full-parallel topology whereby the whole 13-bit packet is transferred in a single cycle along 13 separate wires. The transmission of the packet is managed using a 4-phase handshake protocol. The 4-phases are described below and are annotated in Figure 7.9.

1. Master initialises the data lines, requests permission to send and waits for acknowledgement.
2. Slave responds with acknowledgement and accepts the data if there is room in its buffer.
3. Master turns off request.
4. Slave turns off acknowledgement.

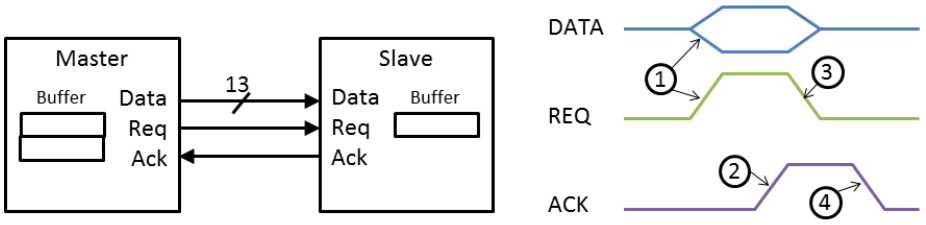


FIGURE 7.9: Interconnection link design and 4-phase handshake protocol.

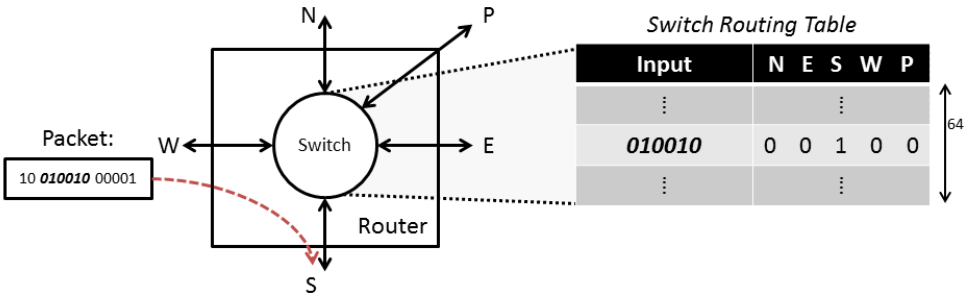


FIGURE 7.10: The switch uses a 64x5-bit LUT to determine the direction in which to route a packet.

7.3.4 Router

The router contains 4 input channels, 4 output channels and a switch. The input channels act as a slave in the interconnection link described above and contain a single register as a buffer. When data is available in the register a signal is used to alert the switch.

The output channels represent the masters in the interconnection link. Each output channel contains an individual FIFO buffer, configured with a depth of 2. A signal is attached to the switch to which details the availability of the buffer. A central global buffer per router is avoided due to the complexity of managing shared access during busy periods [194]

The switch uses a round-robin technique to service the input channels. If an input channel has data available the switch inspects the packets contents to determine the required output channel(s). If the output channels are busy then the input channel is left unserviced and the switch moves onto the next channel. If all input channels are empty then the switch is disabled to conserve power.

The routing of packets is deterministic and is defined by routing LUTs which are pre-configured at compile-time.

The LUTs makes intelligent use of the FPGA resources to be implemented as efficiently as possible. For example, the current packet format limits the number of destinations in the network to $2^6 = 64$. The switch must inspect the most significant 6-bits of the body and determine whereabouts the target core is located in relation to itself and whether to transmit the packet along the north, east, south or west channels or to its own processing core. Hence, there are 64 possible input values and 5 different responses, which can be represented in a 64x3-bit LUT. A single Virtex-7 FPGA slice contains a four 64x1-bit LUT [160]. Therefore only a single slice is required to store the routing information for each switch. The FPGA has 75,900 slices available. This process is illustrated in Figure 7.10.

7.3.5 Interface

The interface contains two major components, a transmitting and a receiving component.

The transmit component is attached to the output of the processing core. Every 100 clock cycles the processing core transmits a pulse stating that a Golgi cell has been calculated and updated. This is accompanied by a 5-bit identifier and a 1-bit signal representing if the cell produced a spike. If the cell did produce a spike the interface uses the identifier as an index into a LUT (RAM1) to fetch a pointer to a second LUT (RAM2). The contents of this second LUT is a list of lists of the Golgi-to-granule connections of this core. The pointer references the start of the list for the particular Golgi cell. The interface then produces a packet for the initial connection in the list and passes the packet to the router. Once the packet has been transferred it attempts to transfer the next packet in the list. This continues until all packets in the list have been transmitted. The end of the list is represented by a Golgi event packet. This process is illustrated in Figure 7.11.

Figure 7.12 illustrates the receiving component of the interface. This is responsible for summating the received packets and transmitting the summated values at the correct time to the granular clusters upon the processing core. For this, a flip-flop buffer arrangement is used to maintain correct operation. When a packet is received a value of

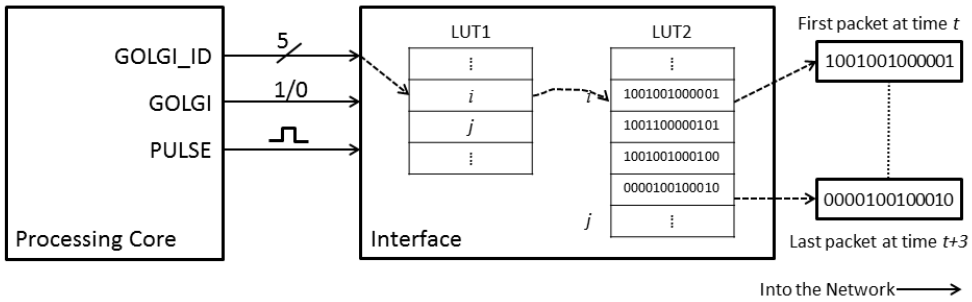


FIGURE 7.11: Interface packet transmit methodology. The interface detects when a Golgi cell produces a spike, $GOLGI_ID$ is used as pointer into $LUT1$, which provides an index into $LUT2$ that contains the list of Golgi-to-granule connections. The interface then produces an individual packet for each connection.

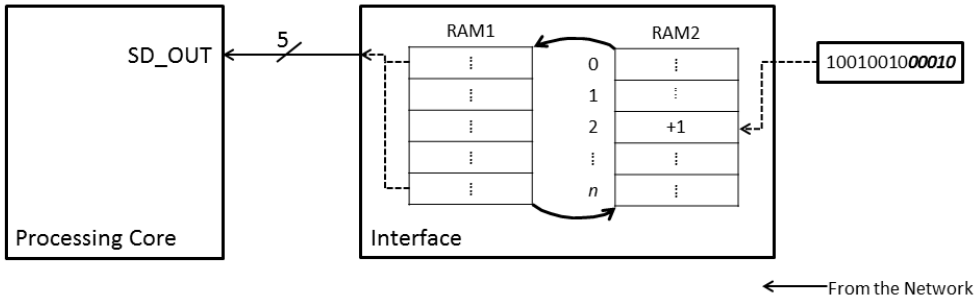


FIGURE 7.12: Interface packet receive methodology. On receipt of a GTG packet the interfaces uses the least significant 5-bits as an index into the RAM2. The value stored in RAM2 is incremented to represent receipt of a synaptic event. Simultaneously the interface is retreiving values from RAM1 and transmitting the synaptic values to the core for processing. At the end of a time frame RAM1 and RAM2 are swapped.

1 is inserted into the target granular clusters corresponding row in buffer A. If a second packet is received for the same cluster then this value is incremented. Meanwhile the processing core is retrieving the previous frames input contents from buffer B. At the end of the processing frame buffer B is cleared and swapped with buffer A.

7.3.6 Unit Cell

The unit cell of the network consists of the router, interface, and processing core along with associated routing and connectivity tables wrapped into a single VHDL module.

7.3.7 Network

A network consists of multiple unit cells interconnected. The process of creating any size of network was automated through development of a Matlab^{TM}script.

7.3.8 Global Frame Master

The global frame master uses a clock-gating routine to control the system time. The frame master receives a signal from all of the routers within the network. This signal represents the state of the router: '1' if the router has packets tranversing through it, '0' if the router is quiet. At the end of the frame, if any routers are still busy the clock to the processing cores are disabled. Only when all the routers are quiet is the clock enabled to allow the processing cores to continue.

7.3.9 Listening Module

The listening module receives a GE packet from every Golgi cell when they produce a spike. The objective of this element is then to pass on the list of Golgi events to an external user using a PC.

With 1000 Golgi cells within the system and an expected 25x speed-up over real-time there are 25M possible events to transmit per second. However, with an expected mean firing rate of 40Hz there is only a 4% chance of a Golgi cell producing an event. As such, a rate of only 1M events/second is expected. With 10bits required to represent the identity of a Golgi cell data throughput of 10Mb/s is needed. This data rate is comfortably within the operating range of the USB, Ethernet and PCI protocols available upon the evaluation board.

To reduce development lead-time a simple UART-based protocol was utilized for debugging. Unfortunately, this can only realistically achieve a data throughput of up to 100kb/s, meaning only a subset of the Golgi events may be transmitted or the neural-NoC may be paused whilst the FPGA-to-PC interface transmits the required data.

7.3.10 Configuration Mode

The configuration mode is used to setup the connectivity between Golgi cells and granular clusters. This can be done either before the model begins to operate or during operation, as long as the simulation is paused. The process of configuration is described below:

1. Halt network.
2. Transmit CS configuration packet to target core.
3. Target core receives CS packet and enters configuration mode.
4. Transmit BC packet containing Golgi-to-granule connection information.
5. Packet received by target core and is entered into connectivity memory.
6. Repeat steps (4) and (5) until all connections are initiated.
7. Transmit CS configuration packet to target core.
8. Target core receives CS packet and leaves configuration mode.
9. Optionally, move on to configure another core.
10. Restore network to operating mode.

To ensure that packets arrive in a consistent order with which they are transmitted a delay is inserted between transmission. Alternatively, the global frame master could be utilized to ensure that the previous packet has reached its destination before transmission of the next packet.

Assuming that there is a 10 cycle delay between the transmission of each configuration packet, there is an operating frequency of 50MHz, and with 1000 Golgi cells there is 8000 Golgi-to-granule connections it is reasonable to predict that using the protocol defined above full configuration of the network will require 1.6ms.

7.3.11 Self-test Mode

A built-in self-test (BIST) routine validates that all routers are functioning and the inter-connection links are operating correctly. An initial packet is generated by an individual router and transmitted to a pseudo-random router within the network. On receipt of this packet the router updates the packet's contents and selects an alternative router to forward the packet onto. This process is repeated until the packet is received by the initial router, where the packet's contents can be checked for consistency to confirm the network is operating as expected.

TABLE 7.1: FPGA Area Utilization for Granular Layer Model. FPGA: Xilinx Virtex-7 XC7VX485T

	Processor	Router	Interface	Connectivity Tables	Module	Total	Utilization
Slice Registers	2884	441	245	38	3676	176424	29%
Slice LUTs	4379	724	369	0	5592	268455	88%
BRAM	20	0	0	0	20	960	93%
DSP48E1s	48	0	0	0	48	2304	82%

The self-test routine may be initiated when the network is first powered. If the routine fails the user can be informed through an on-board status LED or through the FPGA-to-PC interface.

7.4 Results

To validate the correct functionality of the neural-NoC each individual component of the design was extensively tested before integration into the complete system. In this section, a detailed description of the testing of the network is provided along with illustrations of the performance of the granular layer model.

7.4.1 FPGA Implementation

The system design was synthesized, placed, and routed for the target platform using Xilinx ISE 14.6. The final design consisted of 48 processing cores arranged in a 6x8 mesh. This allowed for 960 Golgi cells and 96000 granule cells to be modelled.

The FPGA area utilisation results are available in Table 7.1. For this case study it can be clearly seen that the processing core is more costly in terms of area than the network components. Also the processing core had a maximum operating frequency of 4x slower than the network. As such, two separate clock domains were used with a 4-phase handshake at the interface between the two domains.

7.4.2 Configuration Test

The configuration mode was tested using a behavioural level simulation. A test-bench was developed to stimulate the network with the process outlined in section 7.3.10.

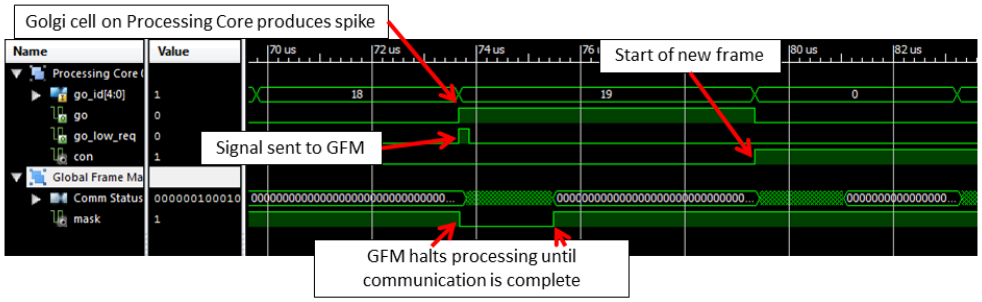


FIGURE 7.13: Global frame master demonstration.

This involved the test-bench storing a list of the connections, and then sequentially transmitting the information of each connection to the target core of the configuration mode. The target core contained a pre-defined list of expected connections. When the pre-defined list matched the received list a signal was raised indicating that the configuration had been successful.

With an operating frequency of 156MHz a single processing core was able to be configured in $12\mu s$.

7.4.3 Global Frame Master

The operation of the global frame master is demonstrated in Figure 7.13. In this figure, a Golgi cell produces a spike, which causes packets to be transmitted through the network. As it is the 20th Golgi cell on the processing core the time frame is about to end. The successive time frame cannot begin until the communication is complete. The global frame master inspects the status of all the routers, only when all the routers declare that they are silent does the global frame master allow the processing to continue.

7.4.4 Network

The network operation was initially tested separately from the processor implementation. This allowed for:

```
1  at 3862400 ps(2): Note: GX,2,1$1=000&3862400 ps (/scaledtb/uut/MJ12/XLXI_2/).
2  at 3888 ns(2): Note: TX,2,1$1=4206&3888000 ps (/scaledtb/uut/MJ12/XLXI_2/).
3  at 3926400 ps(2): Note: TX,2,1$1=4306&3926400 ps (/scaledtb/uut/MJ12/XLXI_2/).
4  at 3984 ns(2): Note: TX,2,1$1=4561&3984000 ps (/scaledtb/uut/MJ12/XLXI_2/).
5  at 4041600 ps(2): Note: TX,2,1$1=4225&4041600 ps (/scaledtb/uut/MJ12/XLXI_2/).
6  at 4092800 ps(2): Note: RX,3,0$000=4206&4092800 ps (/scaledtb/uut/MJ03/XLXI_2/).
7  at 4099200 ps(2): Note: TX,2,1$1=4263&4099200 ps (/scaledtb/uut/MJ12/XLXI_2/).
8  at 4156800 ps(2): Note: TX,2,1$1=4465&4156800 ps (/scaledtb/uut/MJ12/XLXI_2/).
9  at 4214400 ps(2): Note: TX,2,1$1=4656&4214400 ps (/scaledtb/uut/MJ12/XLXI_2/).
10 at 4272 ns(2): Note: TX,2,1$1=4403&4272000 ps (/scaledtb/uut/MJ12/XLXI_2/).
11 at 4272 ns(2): Note: RX,3,1$000=4465&4272000 ps (/scaledtb/uut/MJ13/XLXI_2/).
```

LISTING 7.1: Sample output generated by behavioural testing of network-on-chip design. Line 1 represents a Golgi cell producing a spike on Core (1-2). The first packet with contents 0x4206 is generated on line 2. This packet is received by Core (0-3) on line 6.

- The simulations to be completed in less-time. Due to the complexity of the processor simulation required an extended processing time. For instance, a post-translate simulation[2] required 12 hours for 1 seconds of operation.
- Fine-grain control of the input to the network. For instance, the rate of firing rate could be easily controlled.

To verify the network latency and throughput characteristics behavioural-level Monte-Carlo testing of the implementation was performed using Xilinx ISim 14.6.

Each interface module was connected to a model processor. This model processor generated Golgi cell events randomly at a defined mean rate. The interface then interpreted these events and generated the necessary packets as defined in their connectivity lists. The generated packets were then transmitted to the required destinations throughout the network.

All of the packet events were logged by use of the *report*[3] VHDL command. An example of the generated log file is provided in Listing 7.1. A Matlab^{TM}script was generated to interpret the log file and determine the latency and throughput characteristics.

In Figure 7.14 the latency results are illustrated for the sample network containing 48 processing cores in a 6x8 mesh layout. The figure shows 5 box plots with each box plot being obtained from over 10,000 data samples. The whiskers represent approximately $\pm 3\sigma$ with the outliers beyond this range.

[2] A post-translate simulation is a simulation of the circuit design after the VHDL has been converted into real FPGA sub-components.

[3] The *report* command displays a string on the command line. This command is often used when debugging.

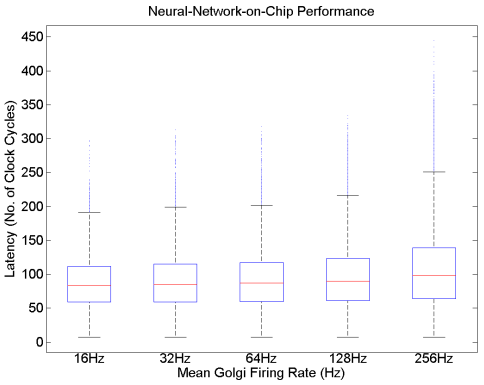
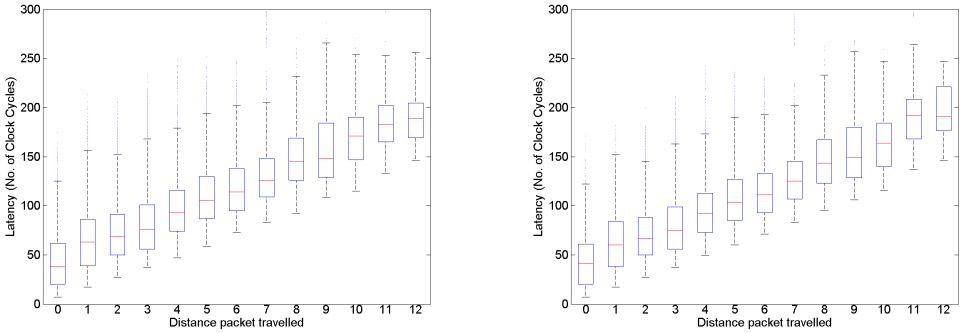


FIGURE 7.14: Neural-NoC performance. Latency of packet transmission versus mean firing rate.



(A) (B)

FIGURE 7.15: Latency of packet transmission against distance packet is required to travel. (A) Mean Golgi firing rate of 32Hz. (B) Mean Golgi firing rate of 64Hz.

As the mean firing rate increases from 16Hz to 256Hz the median latency increases slightly from 84 to 98 clock cycles. Since the processing cores operate approximately 6x slower than the network components each packet can be considered to be transmitted on average in 16.3 processing cycles.

No packets were lost at any of the measured frequencies. Within typical cerebellar systems, the Golgi cells fire at a rate of up to 100Hz [221], which is within the defined network performance characteristics.

In Figure 7.15 the latency is shown against the distance that the packet is required to travel for two different mean firing rates. As the packets travel further through the network the average latency increases. From this figure, the zero-load latency of each step through the network can be determined as approximately 10 clock cycles.

7.4.5 System

For final system verification the network-on-chip design was integrated with the granule-Golgi processing cores. This section illustrates three different experiments completed to confirm the correct operation.

Firstly, Figure 7.16 illustrates spike raster plots of the neural-NoC response demonstrating passage-of-time encoding. Figure 7.16(A) shows the response of 40 random granule cells and Figure 7.16(B) 40 random Golgi cells. The granule cells can be seen to go through bursting phases whilst the Golgi cells maintain a regular spiking interval.

Figure 7.16c compares the NoC response with the software implementation described in [221] using a similarity index metric. For further explanation of this metric refer to [221]. As can be seen by this figure, the response of the NoC and the software is closely correlated. It is believed that the small variation in response is caused by the translation of the processing core from software to hardware. This translation involves switching from floating-point to fixed-point arithmetic and using a simplified Euler integration technique.

Secondly, for further verification an experiment conducted in [221] is repeated to demonstrate the effects of blocking certain ion channels within the neural models. This has the effect of removing the temporal correspondence within the granule cells and they instead fire randomly. In Figure 7.17a, the ion channels within the granule cells are blocked resulting in the granule cells being unable to integrate the mossy fibre input signals over extended periods of time, resulting in sparse firing. Alternatively, in Figure 7.17b, the ion channels within the Golgi cells are blocked resulting in excessive firing from the granule cells as they no longer receive inhibitory feedback from the Golgi cells. The similarity index response highlighted in Figure 7.17c corresponds with the expected response provided in [221] illustrating correct operation.

Finally, an FPGA cerebellar prosthetic was connected to a model of a multichannel recording/stimulating system and the closed-loop response was recorded and compared with the expected response. This experiment was conducted by project colleague Jun Wen Luo and is described in [222].

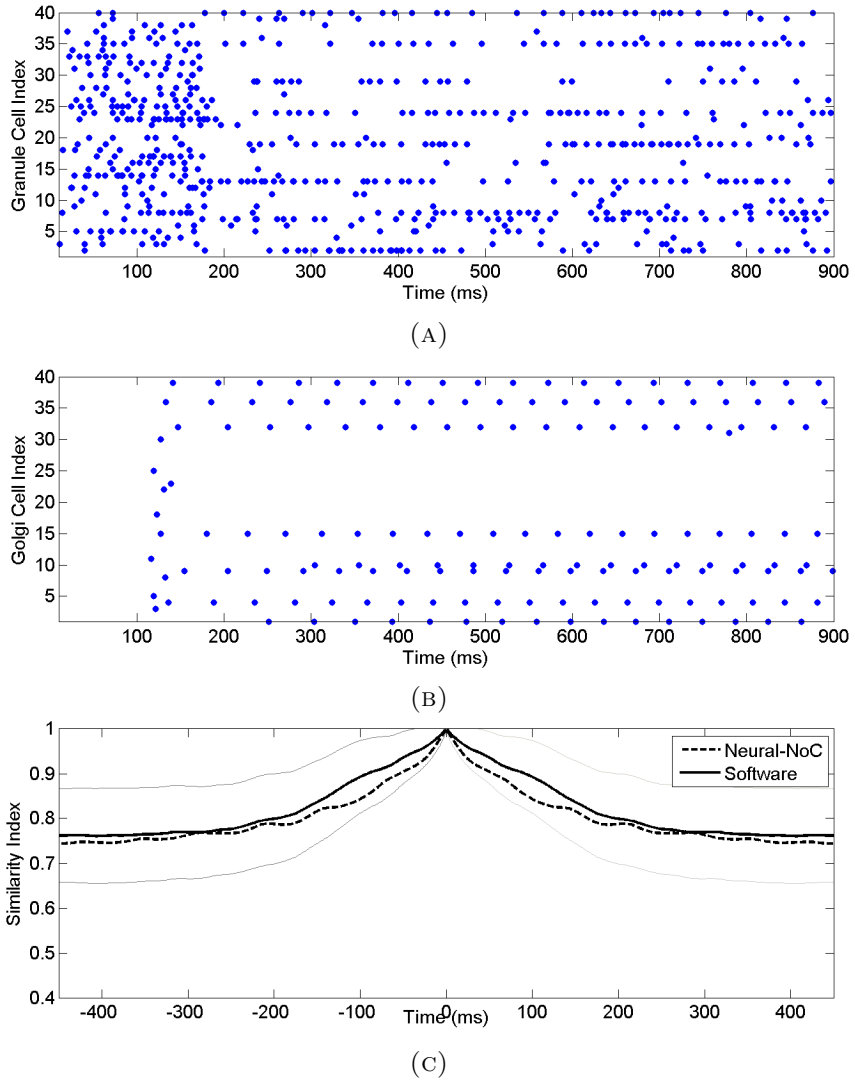
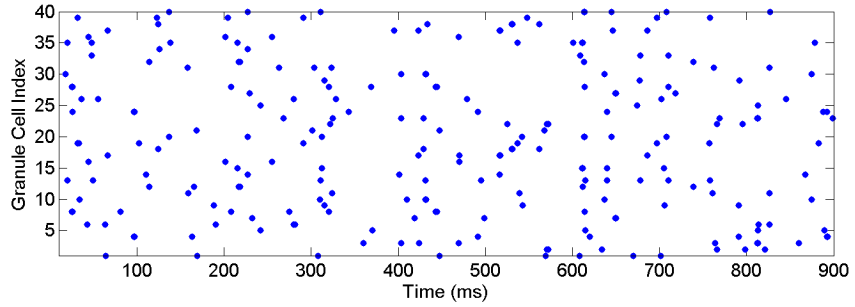


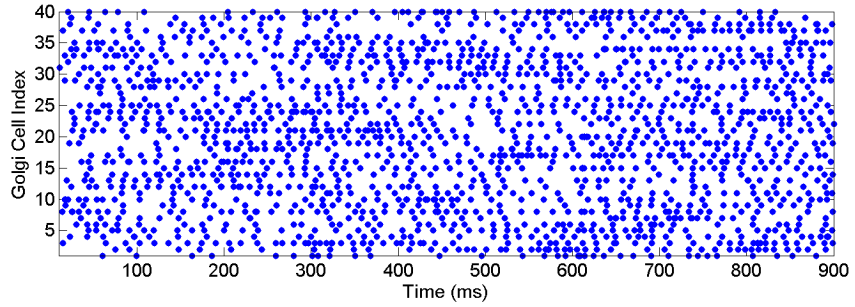
FIGURE 7.16: Neural-NoC system dynamics. (A) Granule cell behaviour. (B) Golgi cell behaviour. (C) Similarity index between software and hardware.

7.5 Discussion

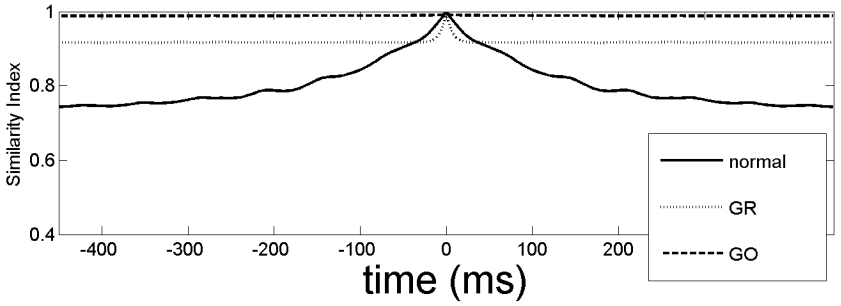
The system has demonstrated correct operation in the previous section. However, it is important to critically analyse the performance and the design of the system to ensure that future developments are focused in the correct areas. In this section, the implemented neural-NoC is analysed from three perspectives: firstly, the system's capability; secondly, the component design; and finally, the system's practicality.



(A)

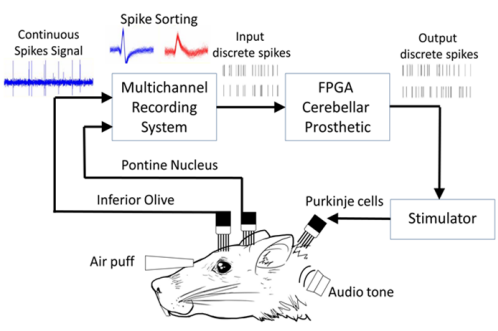


(B)

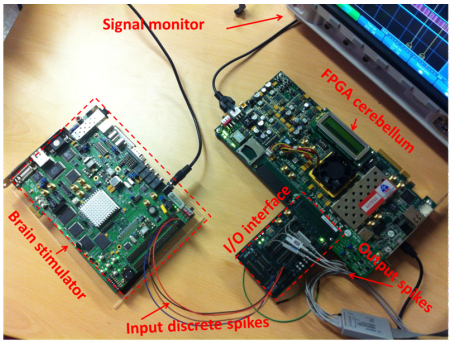


(C)

FIGURE 7.17: Neural-NoC system dynamics with with certain ion channel inhibition. (A) Granule cell output with certain ion channels channels blocked in granule cells. (B) Granule cell output with ion channels blocked in Golgi cells. (C) Similarity index of normal behaviour, and the behaviour of (A) and (B).



(A)



(B)

FIGURE 7.18: Neuroprosthesis laboratory experimental setup. (A) Bio-silicon closed-loop system example. (B) Experimental equivalent.

7.5.1 Capability

The neural network studied in this case study is simpler than most biological neural networks. Despite this perceived simplicity, previous implementations have struggled to achieve comparable performance.

For example, the software (C-coded) implementation described in [221] requires an exponential increase in runtime as the system complexity grows. For a system of the same size as demonstrated here a Intel Quad Core with 8GB of RAM requires 30 seconds to compute. Beyond 100,000 granule cells the implementation runs out of memory resources and can no longer operate.

Alternatively, Yamazaki and Igarashi [103] have shown how a GPU can process 100,000 granule cells in real-time. Similar GPU implementations have been shown to typically achieve a 20x speed increase over equivalent CPU implementations [101][102]. However, as mentioned previously, GPUs suffer from memory and communication weaknesses making them unsuitable for large-scale neural processing. Also, in terms of neuroprosthesis GPUs are inhibited by the inflexibility of their input/output connections.

The FPGA-based neural-NoC utilizes distributed local memory banks to avoid the associated problems of GPU-based designs. Namely, the problem of sharing memory resources and strict memory access patterns.

The processing time required by the neural-NoC approach remains constant regardless of the current state of the system. Only the extra communication time managed by the global frame master may result in delays to the execution time. Due to the parallel nature of this communication and the clustering within the granular layer model this communication time should not increase with increasing model complexity. As such, a model containing 1,000,000 granule cells should complete in the same time as that of 100,000 granule cells. A 100,000 granule cell system has been demonstrated completing 1sec of real-world activity in 25.6ms.

However, currently a larger granular neural-NoC can not be implemented due to the constraints of the FPGA. A bigger FPGA could provide a platform for a larger system. But, the current FPGA platform can be considered top-of-the-range so it is unrealistic to request more resources. Alternatively, by reducing the size of the implemented processing core, more cores could be placed upon the FPGA to allow for a bigger granular layer

system. It can be seen by comparing Table 7.1 with Table 4.5 and Table 4.8 that the resources required by the granule cell processing cores is far greater than the previously developed Hodgkin-Huxley and Izhikevich cores described in Chapter 4.

7.5.2 Design

The designed NoC meets the specified requirements of the system and performs correctly. There has been no design time invested in many common NoC features that have been deemed unnecessary for this application.

For instance, the implemented NoC uses a *store-and-forward* routing mechanism whereby the entire packet is received by a router before transmission to the next router. Alternative mechanisms such as *wormhole*[4] or *virtual cut-through*[5] may reduce latency [194] of transmission and have been previously used by spiking neural network platforms, such as EMBRACE [224]. However, due to the low traffic levels of the granular layer model *store-and-forward* was determined to be sufficient and most suitable in terms of the time required to develop.

EMBRACE also utilize adaptive routing, whereby packets can change the route to their destination depending upon the local instantaneous traffic levels [167]. Again, the low-traffic levels and the fact that the designed system is operating well below its threshold mean that adaptive routing is an unnecessary overhead in terms of design time and silicon area utilization.

SpiNNaker utilize adaptive *emergency* routing to overcome failures in point-to-point links, primarily between chips [4]. This adaptive routing is required due to the high expected rate of failure of components when constructing a system on the scale of SpiNNaker. However, in a single-chip FPGA-based platform point-to-point link damage is expected to be uncommon. A number of FPGA-specific fault correction techniques, such as wear-levelling [225], could be integrated into the system if it is to be used in a critical application.

[4] In wormhole routing the beginning of the packet may be forwarded to its destination before the whole packet is received. [194]

[5] Virtual cut-through is similar to wormhole but the network guarantees that the route of the packet is available to transmit before beginning [194]

SpiNNaker also utilizes link encoding between network components [4] with the aim of reducing power consumption by reducing the number of bit transitions [194][201]. Similar techniques could be easily implemented within the current granular layer NoC design. However, once again the low-traffic levels mean that the NoC links consume only a nominal amount of power in terms of the overall consumption as illustrated by Figure 7.3c.

7.5.3 Practicality

The system has demonstrated high raw performance. However, it's potential can only be fulfilled by overcoming simple practicalities that are often neglected.

For instance, a neuroscientist may wish to inspect the state of all 100,000 granule cells at each time step; therefore, the state of each granule cell would need to be transmitted externally from the FPGA. In accelerated mode this will require an external communication channel with a bandwidth of at least 4Gb/s. On top of the development of this channel, time would need to be invested in producing PC software able to receive and interpret the data.

In terms of closed-loop experiments, 100,000 granule cells may be calculated upon the FPGA but there may be less than 100 output channels available that are capable of communicating with biological cells. This is a common problem in bio-silicon experiments [7].

Weinstein et al. [144] illustrated a software package to efficiently program FPGA-based neural models. This allowed for the high performance of FPGAs to become accessible to non-specialists interested in accelerating their models, particularly neuroscientists. Also, SpiNNaker has developed a compiler to translate the common PyNN programming language to their implementation [226]. A similar software tool would be required for the granular layer model if it is to be commonly used by non-engineers. For example, a tool could accept a network connectivity file as an input, partition the cells of the network into processing cores, and then send this partitioning and connectivity information over a PC-to-FPGA interface to the connectivity module in the NoC. This would remove from the user any need to get involved with any HDLs[6] or programming of the FPGA.

[6] Hardware description language, such as Verilog or VHDL

7.6 Summary

In this section, a NoC hardware architecture has been developed to implement a model of the cerebellar granular layer involving up to 100,000 neurons. The design has been shown to operate in real-time for biological experiments and accelerated time for large-scale simulations to explore the dynamics of the cerebellar network.

Chapter 8

Conclusions

"Having gathered these facts, Watson, I smoked several pipes over them, trying to separate those which were crucial from others which were merely incidental."

The Crooked Man, Sherlock Holmes

Sir Arthur Conan Doyle

8.1 Summary of Motivation

The aim of this study is to extend the research into neural modelling platforms capable of integration with neural prosthesis. Chapter 2 provides a thorough review of the recent progress in neural prostheses, which have the potential to treat previously thought of as incurable diseases, including motor conditions like tetraplegia, sensory conditions such as loss of sight or hearing, and even cognitive diseases. There is a clear and evident trend in all of these systems towards closed-loop systems which rely upon two way communication between an implant and a patient's neurological tissue.

This closed-loop paradigm requires real-time, online computation of many complex features and it is believed that by mimicking the brain's circuits this high level of functionality and complexity can be reproduced, potentially adding significant benefit to many types of neural prosthesis. However, within a neural prosthesis, this computational platform must be portable, necessitating significant energy and size constraints.

The desire of wanting to recreate the computation of neural circuits is fundamental to many different projects, as the brain shows a remarkable ability in many disciplines, from design features such as automatic repair and redundancy to high-level algorithms such as pattern recognition. But, despite the clear link between prosthesis and neural modelling platforms the research literature shows a clear gap of integration of both aspects into a single system.

Therefore, this thesis has explored the electronic requirements for the computation within neuroprosthesis systems.

8.2 Synthesis of Results

This thesis has generated results through initially developing the theoretical underpinnings of a design concept in order to evaluate the optimal design parameters. These optimal design parameters are then further evaluated and studied through practical implementation. This design process has been used in two design studies, first of all into neural processing and secondly into neural network requirements.

For the neural processing, the design theory process flow is widely used in many DSP applications but it has rarely been used in such detail for a neuromorphic system - particularly considering the unique constraints of neural prosthesis. The neural network design study combines many different design process flows, including the neural processing aspect, into a single methodology for the determination of optimal neural network operation for the first time.

In a similar method to Vainbrand et al. [133], this study has attempted to focus first of all on the design parameters through a design theory study before implementing a complete electronic platform. The results of [133] have been extended to consider additional design features, alongside the specific constraint of neural prosthesis applications.

The importance of an initial design theory study has been highlighted in this thesis through the results shown, particularly the high reliance of the final system performance upon the granularity of the system - a factor that has to be chosen at a very early stage in the design process. For example, incorrect selection of this parameter has been shown to cause an impact of between 3x and 5x in the power and area results.

The issue of granularity has only been considered previously by Cassidy et al. [127]. However, the issue of power which is fundamental to long-term neural prosthesis and many other applications was not considered as it was not within their design scope.

The two major design studies have produced results related to granularity and other design processes in three major areas.

Computation Firstly, implementation of neuron processing calculators is very common and done using a variety of technologies and techniques. For instance, software running on PCs or supercomputers is very popular due to its practicality, but it is commonly agreed that the calculation and performance will always be limited by the underlying architecture, which is not always suitable for neural processing. With software approaches the underlying CPU will always have overheads in terms of power and area for components which are fundamentally not required by the application.

Similarly, analogue techniques are popular, perhaps due to the fundamental idea that a neuron membrane works similarly to how a transistor functions. However, it has been shown not to be quite that simple due to scaling and capacitance issues and analogue techniques inherently suffer from not being reconfigurable.

This thesis has investigated the alternative that has gained more traction in recent years - dedicated digital circuit design. The thesis has considered multiple neuron models to investigate the difference between options and on multiple design paradigms.

The Hodgkin-Huxley model implemented upon an FPGA was shown to be feasible in Chapter 4. Izhikevich suggests that this model is too computationally expensive to be worthwhile in comparison to other models such as his own or the leaky integrate and fire system. However, as described in section 4.6 there is only a 15x difference in performance as opposed to the expected 100x [136]. This is caused by the limitations in memory, as opposed to the arithmetic requirements considered by Izhikevich.

This has implications for future development of neural systems, as engineers should not necessarily preclude HH on the basis that it is too computationally expensive, if they do desire to have the increased computational realism. After all, in later chapters the processing has been shown not to be the bottleneck upon the overall system anyway.

For instance, even with the Izhikevich model only 65000 neurons can be implemented in the proposed design due to area limitations imposed by the network.

Communication Secondly, a large focus within the research literature is upon communication between of spikes between neurons, due to the widely held belief that the volume of communication is a bottleneck. The primary reason for the holding of this belief is due to the implementation of neural network systems upon platforms that are not suitable, such as, Von Neumann architectures or large supercomputers that have been designed with different communication patterns in mind. This thesis has shown that communication bottlenecks can be mitigated through the correct selection of design parameters and the efficient implementation of networks suitable for neural network style communication.

The scale of connectivity within neural networks introduce problems for storing the network connectivity information, but the rate of injection into these networks is very low in comparison with typical silicon systems. As such, the burden moves from a communication centric to a memory centric issue.

Memory Finally, the thesis has contributed to the field of knowledge surrounding memory issues in neural-networks-on-chip by studying the design requirements of different networks and the limitations imposed by certain current technologies. This clearly indicates where future research in different memory structures may benefit neural-networks-on-chip design. It is clear that memory is the dominating factor in terms of area and power of any system, but this can be mitigated somewhat by the correct use of granularity as illustrated in all of the practical case studies. This choice of granularity is applicable across many different neural network-on-chip systems not just those targeted towards neural prosthesis.

Correct granularity and the use of local on-chip distributed memory significantly reduces the memory and bandwidth restrictions inherent in many systems. It has been shown how even traditional high performance memory devices such as GPUs are not particularly suitable for neural network-on-chip due to the memory access patterns of neural networks being inherently different to those of typical GPU applications.

8.3 Implications

The major implications of the project are:

Firstly, demonstrated across all aspects of the thesis is that achieving a high performance highly efficient neural network-on-chip system is achievable. However, it relies upon the correct choice of platform and design parameters to meet the application requirements

Secondly, the granularity of a neural network-on-chip is fundamental to the performance. Choosing a granularity should always have reasoning and justification to match the project specifications. For instance, in the second network case study the granularity was adapted to meet the performance criteria as opposed to increase efficiency of area/power relationships.

Finally, although dedicated hardware is fundamental to achieving high performance, this does not mean that we should move away from the digital design approach. The thesis illustrates that digital can achieve what is required and it will continue to scale with technology. Also, its reconfigurability will never truly be matched by analogue design techniques.

8.4 Limitations

Firstly, this study introduces a methodology for investigating the optimum design of a neural network-on-chip. The key limitation to the methodology is the lack of consideration towards online learning within the model. Implementing online learning will add complexity to the system and will likely change some of the key findings in terms of power and area usage and optimum granularity. However, this limitation is mitigated by:

- The addition of online learning to the methodology will not change any of the existing methodology described, only add to it.
- Online learning is, and is always likely to be, an added extra, which is non-vital, perhaps even detrimental to many system. In fact, a large proportion of the key neural network-on-chip studies have also discounted learning from their designs,

such as [217]. Many common signal processing algorithms use artificial neural networks with offline learning. For a neural prosthesis, learning would not be required except perhaps for a memory prosthetic.

Secondly, a key limitation to the study is that no ASIC device has been manufactured to provide verification of the results of the conceptual studies and the design methodology.

This does not impact the accuracy of the results, as many of the theoretical principles, such as granularity, do not fully rely upon direct relationship with implemented technology, but are systematic. For instance, a different granularity will always require a different number of bytes in memory, which will always require a different power and area usage.

Finally, consideration has been taken at all points through the design cycle to investigate the most up to date technology for inclusion within the methodology. For instance, the use of the latest developments in CAM when studying memory usage. However, the neural network-on-chip application requires designs to be taken from a wide pool of component designs, from the optimal design of a transistor, through to the most efficient layout of components, through to 3D design principles. This however, will of course make the full practical design more difficult to achieve and more costly to implement. For instance, although 3D technology is suggested, this is not currently at the same level of maturity of 2D design techniques in terms of design tools and power/area usage estimations. Once again the fundamental design parameters are not limited by this factor as they will continue to scale with technology. For instance, designing a more efficient arithmetic unit may not necessarily alter the optimum way to design an Izhikevich neuron, it may just reduce its total area and power usage.

8.5 Future Work

Successful implementation of a neural prosthesis using a neural network-on-chip will rely upon a cross-disciplinary team of neuroscientists, engineers, physicians and others. As such the options for future work in this field is varied, ranging from investigating the networks to be modelled to the design of low leakage transistors. The CANDO project recently started at Newcastle University is an excellent example of the opportunities

involved in neural prosthesis development and the wide range of skills required for its development.

It is clear from investigating the research literature in the field that a lot of the existing work in neuromorphic technology is based around a set of incomplete assumptions about the biology. Some of these have been mentioned in the thesis, such as the true detail required in spikes, such as specific timing or amplitude. Also, true information about the encoding of information within the spikes is fundamental to fully diagnose brain state and operation. There are further biological unknowns that have not been discussed, such as the function of astrocytes. It is not generally known in the field if these other cells impact upon the computation of the brain.

From an engineering perspective, the neural network-on-chip performance will continue to improve with improvements in technology that is fundamental to the operation. For instance, it has been shown already that dramatic improvements in CAM efficiency in recent years significantly improves potential performance.

As this study has focused upon providing a design methodology to find the optimum design parameters to underpin a particular design, an obvious future work project would be implement a neural network-on-chip for a specific neural prosthesis, such as that currently being considered as part of the CANDO project, in order to treat epilepsy.

8.6 Concluding Statement

The aim has been to extend the knowledge of neural modelling designs for neuroprosthesis. This has been achieved through developing a methodology that other research groups can use to develop new systems. Information can be taken and extended upon by research groups in their own designs.

The methodology for studying both processing and networks on chip for neural networks can be easily implemented by other neural network designers who wish to take applications forward. This scenario has been illustrated within Chapter 7, where the existing work undertaken throughout Chapter 5 and Chapter 6 was translated to a closely related topic with slightly different design parameters, in order to create a prototype working system.

Bibliography

[1] National Research Council, *National Research Council (US) Committee on Research at the Intersection of the Physical and Life Sciences. Research at the Intersection of the Physical and Life Sciences.* Washington (DC): National Academies Press (US), July 2010.

[2] H. Markram, "The blue brain project." *Nature reviews. Neuroscience*, vol. 7, no. 2, pp. 153–60, Feb. 2006.

[3] E. Farquhar and P. Hasler, "A bio-physically inspired silicon neuron," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 52, no. 3, pp. 477–488, Mar. 2005.

[4] S. B. Furber, D. R. Lester, L. A. Plana, J. D. Garside, E. Painkras, S. Temple, and A. D. Brown, "Overview of the SpiNNaker System Architecture," *IEEE Transactions on Computers*, no. January, 2012.

[5] F.-G. Zeng, S. Rebscher, W. Harrison, X. Sun, and H. Feng, "Cochlear implants: system design, integration, and evaluation." *IEEE reviews in biomedical engineering*, vol. 1, pp. 115–42, Jan. 2008.

[6] J. S. Perlmutter and J. W. Mink, "Deep brain stimulation." *Annual review of neuroscience*, vol. 29, pp. 229–57, Jan. 2006.

[7] T. W. Berger, D. Song, R. H. M. Chan, V. Z. Marmarelis, J. Lacoss, J. Wills, R. E. Hampson, S. A. Deadwyler, and J. J. Granacki, "A Hippocampal Cognitive Prosthesis: Multi-Input, Multi-Output Nonlinear Modeling and VLSI Implementation." *IEEE Trans NSRE*, vol. 20, no. 2, pp. 198–211, Mar. 2012.

[8] Newcastle University, "CANDO," 2014. [Online]. Available: http://www.cando.ac.uk/

[9] E. E. Fetz, "Operant Conditioning of Cortical Unit Activity," *Science*, vol. 163, no. 3870, pp. 955–958, Feb. 1969.

[10] C. T. Moritz, S. I. Perlmutter, and E. E. Fetz, "Direct control of paralysed muscles by cortical neurons." *Nature*, vol. 456, no. 7222, pp. 639–42, Dec. 2008.

[11] J. E. ODoherty, M. A. Lebedev, P. J. Ifft, K. Z. Zhuang, S. Shokur, H. Bleuler, and M. Nicolelis, "Active tactile exploration using a brainmachinebrain interface," *Nature*, pp. 1–5, Oct. 2011.

[12] J. K. Chapin, K. A. Moxon, R. S. Markowitz, and M. Nicolelis, "Real-time control of a robot arm using simultaneously recorded neurons in the motor cortex." *Nature neuroscience*, vol. 2, no. 7, pp. 664–70, July 1999.

[13] J. Wessberg, C. R. Stambaugh, J. D. Kralik, P. D. Beck, M. Laubach, J. K. Chapin, J. Kim, S. J. Biggs, M. Srinivasan, and M. Nicolelis, "Real-time prediction of hand trajectory by ensembles of cortical neurons in primates." *Nature*, vol. 408, no. 6810, pp. 361–5, Nov. 2000.

[14] M. Velliste, S. Perel, M. C. Spalding, A. S. Whitford, and A. B. Schwartz, "Cortical control of a prosthetic arm for self-feeding." *Nature*, vol. 453, no. 7198, pp. 1098–101, June 2008.

[15] C. Ethier, E. R. Oby, M. J. Bauman, and L. E. Miller, "Restoration of grasp following paralysis through brain-controlled stimulation of muscles," *Nature*, vol. 485, no. 7398, pp. 368–371, Apr. 2012.

[16] D. A. Borton, M. Yin, J. Aceros, and A. Nurmikko, "An implantable wireless neural interface for recording cortical circuit dynamics in moving primates." *Journal of neural engineering*, vol. 10, no. 2, May 2013.

[17] L. R. Hochberg, M. D. Serruya, G. M. Friehs, J. Mukand, M. Saleh, A. H. Caplan, A. Branner, D. Chen, R. D. Penn, and J. P. Donoghue, "Neuronal ensemble control of prosthetic devices by a human with tetraplegia." *Nature*, vol. 442, no. 7099, pp. 164–71, July 2006.

[18] S.-P. Kim, J. D. Simeral, L. R. Hochberg, J. P. Donoghue, G. M. Friehs, and M. J. Black, "Point-and-click cursor control with an intracortical neural interface system

by humans with tetraplegia." *IEEE Trans NSRE*, vol. 19, no. 2, pp. 193–203, Apr. 2011.

[19] L. R. Hochberg, D. Bacher, B. Jarosiewicz, N. Y. Masse, J. D. Simeral, J. Vogel, S. Haddadin, J. Liu, S. S. Cash, P. van der Smagt, and J. P. Donoghue, "Reach and grasp by people with tetraplegia using a neurally controlled robotic arm," *Nature*, vol. 485, no. 7398, pp. 372–375, May 2012.

[20] W. Wang, J. L. Collinger, A. D. Degenhart, E. C. Tyler-Kabara, A. B. Schwartz, D. W. Moran, D. J. Weber, B. Wodlinger, R. K. Vinjamuri, R. C. Ashmore, J. W. Kelly, and M. L. Boninger, "An electrocorticographic brain interface in an individual with tetraplegia." *PloS one*, vol. 8, no. 2, p. e55344, Jan. 2013.

[21] J. L. Collinger, B. Wodlinger, J. E. Downey, W. Wang, E. C. Tyler-Kabara, D. J. Weber, A. J. C. McMorland, M. Velliste, M. L. Boninger, and A. B. Schwartz, "High-performance neuroprosthetic control by an individual with tetraplegia." *Lancet*, vol. 381, no. 9866, pp. 557–64, Feb. 2013.

[22] R. L. Testerman, M. T. Rise, and P. H. Stypulkowski, "Electrical stimulation as therapy for neurological disorder." *IEEE engineering in medicine and biology magazine : the quarterly magazine of the Engineering in Medicine & Biology Society*, vol. 25, no. 5, pp. 74–8, 2006.

[23] W. Koller, R. Pahwa, K. Busenbark, J. Hubble, S. Wilkinson, a. Lang, P. Tuite, E. Sime, a. Lazano, R. Hauser, T. Malapira, D. Smith, D. Tarsy, E. Miyawaki, T. Norregaard, T. Kormos, and C. W. Olanow, "High-frequency unilateral thalamic stimulation in the treatment of essential and parkinsonian tremor." *Annals of neurology*, vol. 42, no. 3, pp. 292–9, Sept. 1997.

[24] J. Rodriguez-Romaguera, F. H. M. Do Monte, and G. J. Quirk, "Deep brain stimulation of the ventral striatum enhances extinction of conditioned fear," *Proceedings of the National Academy of Sciences*, vol. 2012, pp. 1–6, May 2012.

[25] A. Viswanathan, J. Jimenez-Shahed, J. F. Baizabal Carvallo, and J. Jankovic, "Deep brain stimulation for Tourette syndrome: target selection." *Stereotactic and functional neurosurgery*, vol. 90, no. 4, pp. 213–24, Jan. 2012.

[26] G. Brindley and W. Lewin, "The sensations produced by electrical stimulation of the visual cortex," *The Journal of physiology*, pp. 479–493, 1968.

[27] A. Eshraghi, R. Nazarian, F. F. Telischi, S. M. Rajguru, E. Truy, and C. Gupta, "The cochlear implant: historical aspects and future prospects." *Anatomical record (Hoboken, N.J. : 2007)*, vol. 295, no. 11, pp. 1967–80, Nov. 2012.

[28] "Cochlear - Implants for Children and Adults." [Online]. Available: http://www.cochlear.com

[29] J. D. Loudin, D. M. Simanovskii, K. Vijayraghavan, C. K. Sramek, a. F. Butterwick, P. Huie, G. Y. McLean, and D. V. Palanker, "Optoelectronic retinal prosthesis: system design and performance." *Journal of neural engineering*, vol. 4, no. 1, pp. S72–84, Mar. 2007.

[30] M. S. Humayun, J. D. Weiland, G. Y. Fujii, R. Greenberg, R. Williamson, J. Little, B. Mech, V. Cimmarusti, G. Van Boemel, G. Dagnelie, and E. de Juan, "Visual perception in a blind subject with a chronic microelectronic retinal prosthesis," *Vision Research*, vol. 43, no. 24, pp. 2573–2581, Nov. 2003.

[31] J. Weiland and M. Humayun, "Intraocular retinal prosthesis," *IEEE Engineering in Medicine and Biology Magazine*, vol. 25, no. 5, pp. 60–66, Sept. 2006.

[32] K. Stingl, K. U. Bartz-Schmidt, D. Besch, A. Braun, A. Bruckmann, F. Gekeler, U. Greppmaier, S. Hipp, G. Hörtdörfer, C. Kernstock, A. Koitschev, A. Kusnyerik, H. Sachs, A. Schatz, K. T. Stingl, T. Peters, B. Wilhelm, and E. Zrenner, "Artificial vision with wirelessly powered subretinal electronic implant alpha-IMS." *Proceedings. Biological sciences / The Royal Society*, vol. 280, no. 1757, Apr. 2013.

[33] E. M. Schmidt, M. J. Bak, F. T. Hambrecht, C. V. Kufta, D. K. O'Rourke, and P. Vallabhanath, "Feasibility of a visual prosthesis for the blind based on intracortical microstimulation of the visual cortex." *Brain : a journal of neurology*, Apr. 1996.

[34] E. Zrenner, "Will retinal implants restore vision?" *Science*, vol. 295, no. 5557, pp. 1022–5, Feb. 2002.

[35] M. Mc Laughlin, T. Lu, A. Dimitrijevic, and F.-G. Zeng, "Towards a closed-loop cochlear implant system: application of embedded monitoring of peripheral and central neural activity." *IEEE transactions on neural systems and rehabilitation*

engineering : a publication of the IEEE Engineering in Medicine and Biology Society, vol. 20, no. 4, pp. 443–54, July 2012.

[36] S. Little, A. Pogosyan, S. Neal, B. Zavala, L. Zrinzo, M. Hariz, T. Foltynie, P. Limousin, K. Ashkan, J. FitzGerald, A. L. Green, T. Z. Aziz, and P. Brown, "Adaptive deep brain stimulation in advanced Parkinson disease." *Annals of neurology*, vol. 74, no. 3, pp. 449–57, Sept. 2013.

[37] W.-M. Chen, H. Chiueh, T.-J. Chen, C.-L. Ho, C. Jeng, M.-D. Ker, C.-Y. Lin, Y.-C. Huang, C.-W. Chou, T.-Y. Fan, M.-S. Cheng, Y.-L. Hsin, S.-F. Liang, Y.-L. Wang, F.-Z. Shaw, Y.-H. Huang, C.-H. Yang, and C.-Y. Wu, "A Fully Integrated 8-Channel Closed-Loop Neural-Prosthetic CMOS SoC for Real-Time Epileptic Seizure Control," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 1, pp. 232–247, Jan. 2014.

[38] T. W. Berger, R. E. Hampson, D. Song, A. Goonawardena, V. Z. Marmarelis, and S. A. Deadwyler, "A cortical neural prosthesis for restoring and enhancing memory." *Journal of neural engineering*, vol. 8, no. 4, Aug. 2011.

[39] T. W. Berger, D. Song, R. H. M. Chan, and V. Z. Marmarelis, "The Neurobiological Basis of Cognition: Identification by Multi-Input, Multioutput Nonlinear Dynamic Modeling," *Proceedings of the IEEE. Institute of Electrical and Electronics Engineers*, vol. 98, no. 3, pp. 356–374, Mar. 2010.

[40] H. G. Kranz, "Diagnosis of partial discharge signals using neural networks and minimum distance classification," *IEEE Transactions on Electrical Insulation*, vol. 28, no. 6, pp. 1016–1024, 1993.

[41] P. Bonifazi, F. Difato, P. Massobrio, G. L. Breschi, V. Pasquale, T. Levi, M. Goldin, Y. Bornat, M. Tedesco, M. Bisio, S. Kanner, R. Galron, J. Tessadori, S. Taverna, and M. Chiappalone, "In vitro large-scale experimental and theoretical studies for the realization of bi-directional brain-prostheses." *Frontiers in neural circuits*, vol. 7, p. 40, Jan. 2013.

[42] X. Liu, S. Ramirez, P. T. Pang, C. B. Puryear, A. Govindarajan, K. Deisseroth, and S. Tonegawa, "Optogenetic stimulation of a hippocampal engram activates fear memory recall," *Nature*, Mar. 2012.

[43] S. Ramirez, X. Liu, P.-A. Lin, J. Suh, M. Pignatelli, R. L. Redondo, T. J. Ryan, and S. Tonegawa, "Creating a false memory in the hippocampus." *Science*, vol. 341, no. 6144, pp. 387–91, July 2013.

[44] S. Nabavi, R. Fox, C. D. Proulx, J. Y. Lin, R. Y. Tsien, and R. Malinow, "Engineering a memory with LTD and LTP," *Nature*, June 2014.

[45] M. Pais-Vieira, M. Lebedev, C. Kunicki, J. Wang, and M. Nicolelis, "A brain-to-brain interface for real-time sharing of sensorimotor information." *Scientific reports*, vol. 3, p. 1319, Jan. 2013.

[46] J. D. Simeral, S.-P. Kim, M. J. Black, J. P. Donoghue, and L. R. Hochberg, "Neural control of cursor trajectory and click by a human with tetraplegia 1000 days after implant of an intracortical microelectrode array." *Journal of neural engineering*, vol. 8, no. 2, p. 025027, Apr. 2011.

[47] W. Al-Atabany, B. McGovern, K. Mehran, R. Palmini, and P. Degenaar, "A processing platform for optoelectronic/optogenetic retinal prosthesis." *IEEE transactions on bio-medical engineering*, pp. 1–10, Nov. 2011.

[48] M. Yip, R. Jin, H. Nakajima, K. Stankovic, and A. Chandrakasan, "A Fully-Implantable Cochlear Implant SoC with Piezoelectric Middle Ear Sensor and Energy-Efficient Stimulation in 0.18um HVCMOS," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2014.

[49] P. Berrang, H. Bluger, S. Jarvin, and A. Lupin, "Patent: Totally Implantable Cochlear Prosthesis," 2003.

[50] J. Dethier, P. Nuyujukian, S. I. Ryu, K. V. Shenoy, and K. Boahen, "Design and validation of a real-time spiking-neural-network decoder for brain-machine interfaces." *Journal of neural engineering*, vol. 10, no. 3, June 2013.

[51] Medtronic, "Activa RC: Multi-program rechargeable neurostimulator," Tech. Rep., 2008.

[52] J. R. Wolpaw and D. J. McFarland, "Control of a two-dimensional movement signal by a noninvasive brain-computer interface in humans." *Proceedings of the National Academy of Sciences of the United States of America*, vol. 101, no. 51, pp. 17 849–54, Dec. 2004.

[53] R. R. Harrison, P. T. Watkins, R. J. Kier, R. O. Lovejoy, D. J. Black, B. Greger, and F. Solzbacher, "A Low-Power Integrated Circuit for a Wireless 100-Electrode Neural Recording System," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 1, pp. 123–133, Jan. 2007.

[54] J. Olivo, S. Carrara, and G. De Micheli, "Energy harvesting and remote powering for implantable biosensors," *Sensors Journal, IEEE*, vol. 11, no. 99, pp. 1–1, 2011.

[55] N. Mano, "A 280 microW cm-2 biofuel cell operating at low glucose concentration." *Chemical communications*, no. 19, pp. 2221–3, May 2008.

[56] L. Halámková, J. Halámek, V. Bocharova, A. Szczupak, L. Alfonta, and E. Katz, "Implanted biofuel cell operating in a living snail." *Journal of the American Chemical Society*, vol. 134, no. 11, pp. 5040–3, Mar. 2012.

[57] A. Zebda, S. Cosnier, J.-P. Alcaraz, M. Holzinger, A. Le Goff, C. Gondran, F. Boucher, F. Giroud, K. Gorgy, H. Lamraoui, and P. Cinquin, "Single glucose biofuel cells implanted in rats power electronic devices." *Scientific reports*, vol. 3, p. 1516, Jan. 2013.

[58] E. Katz and K. MacVittie, "Implanted biofuel cells operating in vivo methods, applications and perspectives feature article," *Energy & Environmental Science*, vol. 6, no. 10, p. 2791, 2013.

[59] J. M. Rabaey, M. Mark, D. Chen, C. Sutardja, S. Gowda, M. Wagner, and D. Werthimer, "Powering and communicating with mm-size implants," in *2011 Design, Automation & Test in Europe*. IEEE, Mar. 2011, pp. 1–6.

[60] Medtronic, "Parkinsons Disease." [Online]. Available: http://www.medtronic.com/patients/parkinsons-disease/living-with/replacement/

[61] ——, "Activa PC Neurostimulator."

[62] ——, "System Eligibility Battery Longevity," Tech. Rep., 2010.

[63] American Speech Language Hearing Association, "Cochlear Implants," Tech. Rep., 2004.

[64] M. Yin, D. a. Borton, J. Aceros, W. R. Patterson, and A. V. Nurmikko, "A 100-Channel Hermetically Sealed Implantable Device for Chronic Wireless Neurosensing Applications," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 7, no. 2, pp. 115–128, Apr. 2013.

[65] S. Kim, P. Tathireddy, R. Normann, and F. Solzbacher, "Thermal impact of an active 3-D microelectrode array implanted in the brain." *IEEE transactions on neural systems and rehabilitation engineering : a publication of the IEEE Engineering in Medicine and Biology Society*, vol. 15, no. 4, pp. 493–501, Dec. 2007.

[66] S. Mrozek, F. Vardon, and T. Geeraerts, "Brain temperature: physiology and pathophysiology after brain injury." *Anesthesiology research and practice*, vol. 2012, p. 989487, Jan. 2012.

[67] C. M. Lopez, A. Andrei, S. Mitra, M. Welkenhuysen, W. Eberle, C. Bartic, R. Puers, R. F. Yazicioglu, and G. G. E. Gielen, "An Implantable 455-Active-Electrode 52-Channel CMOS Neural Probe," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 1, pp. 248–261, Jan. 2014.

[68] W.-S. Liew, X. Zou, L. Yao, and Y. Lian, "A 1-V 60-W 16-channel interface chip for implantable neural recording," in *2009 IEEE Custom Integrated Circuits Conference*, no. Cicc. IEEE, Sept. 2009, pp. 507–510.

[69] T. Jochum, T. Denison, and P. Wolf, "Integrated circuit amplifiers for multi-electrode intracortical recording." *Journal of neural engineering*, vol. 6, no. 1, p. 012001, Feb. 2009.

[70] V. Karkare, S. Gibson, C.-h. Yang, H. Chen, and D. Marković, "A 75W, 16-Channel Neural Spike-Sorting Processor with Unsupervised Clustering," in *VLSI Circuits (VLSIC), 2011 Symposium on*, 2011, pp. 252–253.

[71] S. Kelly and J. Wyatt, "A power-efficient neural tissue stimulator with energy recovery," *Biomedical Circuits and Systems, IEEE*, vol. 5, no. 1, pp. 20–29, 2011.

[72] T. Berger, M. Baudry, R. Brinton, J.-S. Liaw, V. Marmarelis, A. Yoondong Park, B. Sheu, and A. Tanguay, "Brain-implantable biomimetic electronics as the next era in neural prosthetics," *Proceedings of the IEEE*, vol. 89, no. 7, pp. 993–1012, July 2001.

[73] T. W. Berger, A. Ahuja, S. H. Courellis, S. Deadwyler, G. Erinjippurath, G. Gerhardt, G. Gholmieh, J. J. Granacki, R. Hampson, M. C. Hsaio, J. LaCoss, V. Z. Marmarelis, P. Nasiatka, V. Srinivasan, D. Song, A. R. Tanguay, and J. Wills, "Restoring lost cognitive function." *IEEE engineering in medicine and biology magazine*, vol. 24, no. 5, pp. 30–44, 2005.

[74] S. A. Bamford, R. Hogri, A. Giovannucci, A. H. Taub, I. Herreros, P. F. M. J. Verschure, M. Mintz, and P. Del Giudice, "A VLSI Field-Programmable Mixed-Signal Array to Perform Neural Signal Processing and Neural Modeling in a Prosthetic System." *IEEE transactions on neural systems and rehabilitation engineering*, vol. 20, no. 4, pp. 455–67, July 2012.

[75] S. Furber and S. Temple, "Neural systems engineering." *Journal of the Royal Society*, vol. 4, no. 13, pp. 193–206, Apr. 2007.

[76] V. Gilja, P. Nuyujukian, C. a. Chestek, J. P. Cunningham, B. M. Yu, J. M. Fan, M. M. Churchland, M. T. Kaufman, J. C. Kao, S. I. Ryu, and K. V. Shenoy, "A high-performance neural prosthesis enabled by control algorithm design." *Nature neuroscience*, vol. 15, no. 12, pp. 1752–7, Dec. 2012.

[77] T. Yamazaki and S. Tanaka, "Computational models of timing mechanisms in the cerebellar granular layer." *Cerebellum*, vol. 8, no. 4, pp. 423–32, Dec. 2009.

[78] R. D. Traub, D. Contreras, M. O. Cunningham, H. Murray, F. E. N. LeBeau, A. Roopun, A. Bibbig, W. B. Wilent, M. J. Higley, and M. A. Whittington, "Single-column thalamocortical network model exhibiting gamma oscillations, sleep spindles, and epileptogenic bursts." *Journal of neurophysiology*, vol. 93, no. 4, pp. 2194–232, Apr. 2005.

[79] N. Kopell, C. Borgers, D. Pervouchine, P. Malerba, and A. Tort, "Gamma and Theta Rhythms in Biophysical Models of Hippocampal Circuits," in *Hippocampal Microcircuits*, V. Cutsuridis, B. Graham, S. Cobb, and I. Vida, Eds. New York, NY: Springer New York, 2010, pp. 423–457. [Online]. Available: http://link.springer.com/10.1007/978-1-4419-0996-1

[80] R. R. Trippi and E. Turban, *Neural Networks in Finance and Investing: Using Artificial Intelligence to Improve Real World Performance*. New York, NY, USA: McGraw-Hill, 1992.

[81] M. Negnevitsky, *Artificial Intelligence: A Guide to Intelligent Systems*, 2nd ed. Pearson Education Limited, 2005.

[82] S. Haykin, *Neural Networks and Learning Machines*. Prentice Hall, 2008.

[83] M. S. Lewicki, "A review of methods for spike sorting: the detection and classification of neural action potentials." *Network*, vol. 9, no. 4, pp. 53–78, Nov. 1998.

[84] M. S. Chae, Z. Yang, M. R. Yuce, L. Hoang, and W. Liu, "A 128-channel 6 mW wireless neural recording IC with spike feature extraction and UWB transmitter." *IEEE transactions on neural systems and rehabilitation engineering : a publication of the IEEE Engineering in Medicine and Biology Society*, vol. 17, no. 4, pp. 312–21, Aug. 2009.

[85] V. Karkare and S. Gibson, "A 130-$ mu $ W, 64-Channel Neural Spike-Sorting DSP Chip," *Solid-State Circuits, IEEE*, vol. 46, no. 5, pp. 1214–1222, 2011.

[86] M. Rizk, I. Obeid, S. H. Callender, and P. D. Wolf, "A single-chip signal processing and telemetry engine for an implantable 96-channel neural data acquisition system." *Journal of neural engineering*, vol. 4, no. 3, pp. 309–21, Sept. 2007.

[87] U. Frey, J. Sedivy, F. Heer, R. Pedron, M. Ballini, J. Mueller, D. Bakkum, S. Hafizovic, F. D. Faraci, F. Greve, K.-U. Kirstein, and A. Hierlemann, "Switch-Matrix-Based High-Density Microelectrode Array in CMOS Technology," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 2, pp. 467–482, Feb. 2010.

[88] E. S. Boyden, F. Zhang, E. Bamberg, G. Nagel, and K. Deisseroth, "Millisecond-timescale, genetically targeted optical control of neural activity," *Nature Neuroscience*, vol. 8, no. 9, pp. 1263–1268, 2005.

[89] V. Poher, N. Grossman, G. T. Kennedy, K. Nikolic, H. X. Zhang, Z. Gong, E. M. Drakakis, E. Gu, M. D. Dawson, P. M. W. French, P. Degenaar, and M. a. a. Neil, "Micro-LED arrays: a tool for two-dimensional neuron stimulation," *Journal of Physics D: Applied Physics*, vol. 41, no. 9, p. 094014, May 2008.

[90] M. Rizk, I. Obeid, S. H. Callender, and P. D. Wolf, "A single-chip signal processing and telemetry engine for an implantable 96-channel neural data

acquisition system." *Journal of neural engineering*, vol. 4, no. 3, pp. 309–21, Sept. 2007.

[91] H. Goto, T. Sugiura, Y. Harada, and T. Kazui, "Feasibility of using the automatic generating system for quartz watches as a leadless pacemaker power source." *Medical & biological engineering & computing*, vol. 37, no. 3, pp. 377–80, May 1999.

[92] S. Suner, M. R. Fellows, C. Vargas-Irwin, G. K. Nakata, and J. P. Donoghue, "Reliability of signals from a chronically implanted, silicon-based electrode array in non-human primate primary motor cortex." *IEEE transactions on neural systems and rehabilitation engineering*, vol. 13, no. 4, pp. 524–41, Dec. 2005.

[93] R. Biran, D. C. Martin, and P. a. Tresco, "Neuronal cell loss accompanies the brain tissue response to chronically implanted silicon microelectrode arrays." *Experimental neurology*, vol. 195, no. 1, pp. 115–26, Sept. 2005.

[94] L. Smith, "Implementing neural models in silicon," *Handbook of Nature-Inspired and Innovative Computing*, 2006.

[95] K. Boahen, "Neuromorphic Chips," *Scientific American*, vol. 292, no. May, 2005.

[96] S. B. Laughlin and T. J. Sejnowski, "Communication in neuronal networks." *Science*, vol. 301, no. 5641, pp. 1870–4, Sept. 2003.

[97] M. L. Hines and N. T. Carnevale, "The NEURON simulation environment." *Neural computation*, vol. 9, no. 6, pp. 1179–209, Aug. 1997.

[98] J. Bower and D. Beeman, *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System*, 1998.

[99] J. C. Bettencourt, K. P. Lillis, L. R. Stupin, and J. a. White, "Effects of imperfect dynamic clamp: computational and experimental results." *Journal of neuroscience methods*, vol. 169, no. 2, pp. 282–9, Apr. 2008.

[100] R. J. Butera, C. G. Wilson, C. a. Delnegro, and J. C. Smith, "A methodology for achieving high-speed rates for artificial conductance injection in electrically excitable biological cells." *IEEE transactions on bio-medical engineering*, vol. 48, no. 12, pp. 1460–70, Dec. 2001.

[101] A. K. Fidjeland and M. P. Shanahan, "Accelerated simulation of spiking neural networks using GPUs," *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, July 2010.

[102] J. M. Nageswaran, N. Dutt, J. L. Krichmar, A. Nicolau, and A. V. Veidenbaum, "A configurable simulation environment for the efficient simulation of large-scale spiking neural networks on graphics processors." *Neural networks : the official journal of the International Neural Network Society*, vol. 22, no. 5-6, pp. 791–800, 2009.

[103] T. Yamazaki and J. Igarashi, "Realtime cerebellum: a large-scale spiking network model of the cerebellum that runs in realtime using a graphics processing unit." *Neural networks : the official journal of the International Neural Network Society*, vol. 47, pp. 103–11, Nov. 2013.

[104] R. Ananthanarayanan, S. K. Esser, H. D. Simon, and D. S. Modha, "The cat is out of the bag," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, no. c, 2009, p. 1.

[105] AritificalBrains, "DARPA SyNAPSE Program." [Online]. Available: http://www.artificialbrains.com/darpa-synapse-program

[106] E. M. Izhikevich and G. M. Edelman, "Large-scale model of mammalian thalamocortical systems." *Proceedings of the National Academy of Sciences of the United States of America*, vol. 105, no. 9, pp. 3593–8, Mar. 2008.

[107] S. Furber and A. Brown, "Biologically-Inspired Massively-Parallel Architectures - Computing Beyond a Million Processors," in *9th Int.Conf. Application of Concurrency to System Design*. IEEE, July 2009, pp. 3–12.

[108] X. Jin, S. Furber, and J. Woods, "Efficient modelling of spiking neural networks on a scalable chip multiprocessor," *Neural Networks*, pp. 2812–2819, 2008.

[109] T. Sharp, F. Galluppi, A. Rast, and S. Furber, "Power-efficient simulation of detailed cortical microcircuits on SpiNNaker." *Journal of neuroscience methods*, vol. 210, no. 1, pp. 110–8, Sept. 2012.

[110] C. Mead, *Analog VLSI and Neural Systems*. Addison Wesley Publising Company, 1989.

[111] M. Mahowald and R. Douglas, "A silicon neuron," *Nature*, 1991.

[112] J. H. B. Wijekoon and P. Dudek, "Integrated circuit implementation of a cortical neuron," *2008 IEEE International Symposium on Circuits and Systems*, pp. 1784–1787, May 2008.

[113] R. J. Vogelstein, U. Mallik, J. T. Vogelstein, and G. Cauwenberghs, "Dynamically reconfigurable silicon array of spiking neurons with conductance-based synapses." *IEEE transactions on neural networks*, vol. 18, no. 1, pp. 253–65, Jan. 2007.

[114] K. Hynna and K. Boahen, "Neuronal ion-channel dynamics in silicon," *Internatinal Symposium on Circuits and Systems*, pp. 3614–3617, 2006.

[115] G. Indiveri, E. Chicca, and R. Douglas, "A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity." *IEEE transactions on neural networks*, vol. 17, no. 1, pp. 211–21, Jan. 2006.

[116] G. Indiveri, B. Linares-Barranco, T. J. Hamilton, A. van Schaik, R. Etienne-Cummings, T. Delbruck, S.-C. Liu, P. Dudek, P. Häfliger, S. Renaud, J. Schemmel, G. Cauwenberghs, J. Arthur, K. Hynna, F. Folowosele, S. Saighi, T. Serrano-Gotarredona, J. Wijekoon, Y. Wang, and K. Boahen, "Neuromorphic silicon neuron circuits." *Frontiers in neuroscience*, vol. 5, no. May, p. 73, Jan. 2011.

[117] J. Schemmel, D. Briiderle, A. Griibl, M. Hock, K. Meier, and S. Millner, "A wafer-scale neuromorphic hardware system for large-scale neural modeling," in *International Symposium on Circuits and Systems*. IEEE, May 2010, pp. 1947–1950.

[118] S. Scholze, H. Eisenreich, S. Höppner, G. Ellguth, S. Henker, M. Ander, S. Hänzsche, J. Partzsch, C. Mayr, and R. Schüffny, "A 32GBit/s communication SoC for a waferscale neuromorphic system," *Integration, the VLSI Journal*, vol. 45, no. 1, pp. 61–75, Jan. 2012.

[119] R. Emery, A. Yakovlev, and G. Chester, "Connection-centric network for spiking neural networks," in *Proceedings of the 2009 3rd ACM/IEEE International Symposium on Networks-on-Chip*. IEEE Computer Society, 2009, pp. 144–152.

[120] J.-s. Seo, B. Brezzo, Y. Liu, B. D. Parker, S. K. Esser, R. K. Montoye, B. Rajendran, J. Tierno, L. Chang, D. S. Modha, and D. J. Friedman, "A 45nm

CMOS neuromorphic chip with a scalable architecture for learning in networks of spiking neurons," *2011 IEEE Custom Integrated Circuits Conference (CICC)*, pp. 1–4, Sept. 2011.

[121] N. Imam, K. Wecker, J. Tse, R. Karmazin, and R. Manohar, "Neural spiking dynamics in asynchronous digital circuits," *The 2013 International Joint Conference on Neural Networks (IJCNN)*, no. 1, pp. 1–8, Aug. 2013.

[122] A. Cassidy and A. G. Andreou, "Dynamical digital silicon neurons," in *2008 IEEE Biomedical Circuits and Systems Conference*. IEEE, 2008, pp. 289–292.

[123] D. Thomas and W. Luk, "FPGA Accelerated Simulation of Biologically Plausible Spiking Neural Networks," *2009 17th IEEE Symposium on Field Programmable Custom Computing Machines*, pp. 45–52, 2009.

[124] T. S. T. Mak, G. Rachmuth, K.-P. Lam, and C.-S. Poon, "A component-based FPGA design framework for neuronal ion channel dynamics simulations." *IEEE transactions on neural systems and rehabilitation engineering*, vol. 14, no. 4, pp. 410–8, Dec. 2006.

[125] S. W. Moore, P. J. Fox, S. J. Marsh, a. T. Markettos, and A. Mujumdar, "Bluehive - A Field-Programable Custom Computing Machine for Extreme-Scale Real-Time Neural Network Simulation," *IEEE 20th Int. Sym. Field-Programmable Custom Computing Machines*, pp. 133–140, Apr. 2012.

[126] C.-S. Poon and K. Zhou, "Neuromorphic silicon neurons and large-scale neural networks: challenges and opportunities." *Frontiers in neuroscience*, vol. 5, no. September, p. 108, Jan. 2011.

[127] A. Cassidy, A. G. Andreou, and J. Georgiou, "Design of a one million neuron single FPGA neuromorphic system for real-time multimodal scene analysis," *2011 45th Annual Conference on Information Sciences and Systems*, pp. 1–6, Mar. 2011.

[128] R. K. Weinstein and R. H. Lee, "Architectures for high-performance FPGA implementations of neural models." *Journal of neural engineering*, vol. 3, no. 1, pp. 21–34, Mar. 2006.

[129] L. Maguire, T. McGinnity, B. Glackin, a. Ghani, a. Belatreche, and J. Harkin, "Challenges for large-scale implementations of spiking neural networks on FPGAs," *Neurocomputing*, vol. 71, no. 1-3, pp. 13–29, Dec. 2007.

[130] P. J. Fox, "Massively Parallel Neural Computation," Ph.D. dissertation, University of Cambridge, 2012.

[131] S. Carrillo, J. Harkin, L. J. McDaid, F. Morgan, S. Pande, S. Cawley, and B. McGinley, "Scalable Hierarchical Network-on-Chip Architecture for Spiking Neural Network Hardware Implementations," *IEEE Transactions on Parallel and Distributed Systems*, pp. 1–1, 2012.

[132] ——, "Scalable Hierarchical Network-on-Chip Architecture for Spiking Neural Network Hardware Implementations," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 12, pp. 2451–2461, Dec. 2013.

[133] D. Vainbrand and R. Ginosar, "Scalable network-on-chip architecture for configurable neural networks," *Microprocessors and Microsystems*, vol. 35, no. 2, pp. 152–166, Mar. 2011.

[134] Wikimedia, "File:Neuron.svg," 2014. [Online]. Available: http://commons.wikimedia.org/wiki/File:Neuron.svg

[135] A. Hodgkin and A. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of physiology*, pp. 500–544, 1952.

[136] E. M. Izhikevich, "Which model to use for cortical spiking neurons?" *IEEE Trans NN*, vol. 15, no. 5, pp. 1063–70, Sept. 2004.

[137] ——, "Simple model of spiking neurons." *IEEE transactions on neural networks*, vol. 14, no. 6, pp. 1569–72, Jan. 2003.

[138] J. Luo, T. Mak, B. Yu, P. Andras, and A. Yakovlev, "Towards neuro-silicon interface using reconfigurable dynamic clamping." in *International Conference of the IEEE Engineering in Medicine and Biology Society*, vol. 2011, 2011, p. 6389.

[139] N. H. E. Weste and D. M. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, 4th ed. Addison-Wesley, 2011.

[140] A. Sharp, L. F. Abbott, and E. Marder, "Artificial electrical synapses in oscillatory networks." *Journal of neurophysiology*, vol. 67, no. 6, pp. 1691–4, June 1992.

[141] T. J. Kispersky, M. N. Economo, P. Randeria, and J. a. White, "GenNet: A Platform for Hybrid Network Experiments." *Frontiers in neuroinformatics*, vol. 5, p. 11, Jan. 2011.

[142] T. Nowotny, A. Szucs, R. D. Pinto, and A. I. Selverston, "StdpC: a modern dynamic clamp." *Journal of neuroscience methods*, vol. 158, no. 2, pp. 287–99, Dec. 2006.

[143] E. L. Graas, E. a. Brown, and R. H. Lee, "An FPGA-based approach to high-speed simulation of conductance-based neuron models." *Neuroinformatics*, vol. 2, no. 4, pp. 417–36, Jan. 2004.

[144] R. K. Weinstein, M. S. Reid, and R. H. Lee, "Methodology and design flow for assisted neural-model implementations in FPGAs." *IEEE transactions on neural systems and rehabilitation engineering*, vol. 15, no. 1, pp. 83–93, Mar. 2007.

[145] Y. Zhang, J. P. Mcgeehan, E. M. Regan, S. Kelly, and J. L. Nunez-Yanez, "Biophysically Accurate Foating Point Neuroprocessors for Reconfigurable Logic," *IEEE Transactions on Computers*, vol. 62, no. 3, pp. 599–608, Mar. 2013.

[146] J. Volder, "The CORDIC trigonometric computing technique," *Electronic Computers, IRE Transactions on*, pp. 330–334, 1959.

[147] J. L. Hindmarsh and R. M. Rose, "A Model of Neuronal Bursting Using Three Coupled First Order Differential Equations," *Proceedings of the Royal Society B: Biological Sciences*, vol. 221, no. 1222, pp. 87–102, Mar. 1984.

[148] Xilinx, "7 Series FPGAs Overview," Tech. Rep., 2014.

[149] F. D. Dinechin and B. Pasca, "Floating-point exponential functions for DSP-enabled FPGAs," in *2010 International Conference on Field-Programmable Technology*. IEEE, Dec. 2010, pp. 110–117.

[150] F. De Dinechin and B. Pasca, "Floating-point exponential functions for DSP-enabled FPGAs," in *Field-Programmable Technology (FPT), 2010 International Conference on*. IEEE, 2010, pp. 110–117.

[151] R. K. Weinstein and R. H. Lee, "Architectures for high-performance FPGA implementations of neural models." *Journal of neural engineering*, vol. 3, no. 1, pp. 21–34, Mar. 2006.

[152] R. Butera and R. Lin, "Key Factors for Improving Dynamic-Clamp Performance," in *Dynamic-Clamp*. New York, NY: Springer New York, 2009, vol. 3, pp. 383–397. [Online]. Available: http://www.springerlink.com/index/10.1007/978-0-387-89279-5

[153] Xilinx and Mathworks, "FPGA Design and Codesign," 2014. [Online]. Available: http://www.mathworks.co.uk/fpga-design/simulink-with-xilinx-system-generator-for-dsp.html

[154] Xilinx, "ISE Design Suite," 2014. [Online]. Available: www.xilinx.com/products/design-tools/ise-design-suite/

[155] C. Koch, *Biophysics of Computation: Information Processing in Single Neurons*. Oxford University Press, 1999.

[156] P. Dayan and L. F. Abbott, *Theoretical Neuroscience: Computing and Mathematical Modeling of Neural Systems*. MIT Press, 2005.

[157] A. a. Prinz, L. F. Abbott, and E. Marder, "The dynamic clamp comes of age." *Trends in neurosciences*, vol. 27, no. 4, pp. 218–24, Apr. 2004.

[158] M. Ambroise, T. Levi, Y. Bornat, and S. Saïghi, "Biorealistic spiking neural network on FPGA," in *2013 47th Annual Conference on Information Sciences and Systems (CISS)*. IEEE, Mar. 2013, pp. 1–6.

[159] D. B. Thomas and W. Luk, "FPGA Accelerated Simulation of Biologically Plausible Spiking Neural Networks," 2009.

[160] Xilinx, "7 Series FPGAs Configurable Logic Block - User Guide," Tech. Rep., 2013.

[161] J. H. B. Wijekoon and P. Dudek, "VLSI circuits implementing computational models of neocortical circuits." *Journal of neuroscience methods*, vol. 210, no. 1, pp. 93–109, Sept. 2012.

[162] Z. Hu, A. Buyuktosunoglu, V. Srinivasan, V. Zyuban, H. Jacobson, and P. Bose, "Microarchitectural techniques for power gating of execution units," *Proceedings of the 2004 international symposium on Low power electronics and design - ISLPED '04*, p. 32, 2004.

[163] J. Butts and G. Sohi, "A static power model for architects," in *IEEE/ACM International Symposium on Microarchitecture.* IEEE, 2000, pp. 191–201.

[164] University of Manchester, "SpiNNaker Home Page," 2014. [Online]. Available: http://apt.cs.manchester.ac.uk/projects/SpiNNaker/

[165] E. M. Izhikevich, J. a. Gally, and G. M. Edelman, "Spike-timing dynamics of neuronal groups." *Cerebral cortex*, vol. 14, no. 8, pp. 933–44, Aug. 2004.

[166] P. A. Merolla, J. V. Arthur, B. E. Shi, and K. A. Boahen, "Expandable Networks for Neuromorphic Chips," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 2, pp. 301–311, Feb. 2007.

[167] S. Carrillo, J. Harkin, L. McDaid, S. Pande, S. Cawley, B. McGinley, and F. Morgan, "Advancing interconnect density for spiking neural network hardware implementations using traffic-aware adaptive network-on-chip routers." *Neural networks : the official journal of the International Neural Network Society*, vol. 33C, pp. 42–57, Apr. 2012.

[168] J. Schemmel, J. Fieres, and K. Meier, "Wafer-scale integration of analog neural networks," *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pp. 431–438, June 2008.

[169] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "Cost considerations in network on chip," *Integration, the VLSI Journal*, vol. 38, no. 1, pp. 19–42, Oct. 2004.

[170] E. F. Krause, *Taxicab Geometry: An Adventure in Non-Euclidean Geometry.* Dover Publications, 1975.

[171] B. P. Dally, William James; Towles, *Principles and Practices of Interconnection Networks.* Elsevier Science, 2004.

[172] J. Wu and S. Furber, "A Multicast Routing Scheme for a Universal Spiking Neural Network Architecture," *The Computer Journal*, vol. 53, no. 3, pp. 280–288, Apr. 2009.

[173] P. Kelly, "Dimension Order Routing," 1998. [Online]. Available: https://www.doc.ic.ac.uk/~phjk/AdvancedCompArchitecture/2001-02/Lectures.old/Ch06/node34.html

[174] M. Khan, D. Lester, L. Plana, A. Rast, X. Jin, E. Painkras, and S. Furber, "SpiNNaker: Mapping neural networks onto a massively-parallel chip multiprocessor," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. IEEE, June 2008, pp. 2849–2856.

[175] UCSB, "Espresso Logic Minimizer." [Online]. Available: http://embedded.eecs.berkeley.edu/pubs/downloads/espresso/index.htm

[176] P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh, "Effect of Traffic Localization on Energy Dissipation in NoC-based Interconnect," *2005 IEEE International Symposium on Circuits and Systems*, pp. 1774–1777, 2005.

[177] B. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell system technical journal*, 1970.

[178] G. Karypis and V. Kumar, "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 359–392, Jan. 1998.

[179] G. Karypis, "METIS - Serial Graph Partitioning," 2014. [Online]. Available: http://glaros.dtc.umn.edu/gkhome/metis/metis/overview

[180] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing." *Science (New York, N.Y.)*, vol. 220, no. 4598, pp. 671–80, May 1983.

[181] A. Jantsch, R. Lauter, and A. Vitkowski, "Power analysis of link level and end-to-end data protection in networks on chip," *2005 IEEE International Symposium on Circuits and Systems*, no. 2, pp. 1770–1773, 2005.

[182] Wikimedia, "SRAM Cell (6 Transistor)," 2009. [Online]. Available: http://commons.wikimedia.org/wiki/File:SRAM_Cell_(6_Transistors).svg

[183] R. Evans and P. Franzon, "Energy consumption modeling and optimization for SRAM's," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 5, pp. 571–579, May 1995.

[184] B. Amrutur and M. Horowitz, "Speed and power scaling of SRAM's," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 2, pp. 175–185, Feb. 2000.

[185] E. Donkoh, "Design and Modeling of Low-Power Register File Memories," Ph.D. dissertation, Oregon State University, 2014.

[186] N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi, "CACTI 6 . 0 : A Tool to Model Large Caches CACTI 6 . 0 : A Tool to Model Large Caches," pp. 0–24, 2009.

[187] S. Wilton and N. Jouppi, "CACTI: an enhanced cache access and cycle time model," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 5, pp. 677–688, May 1996.

[188] Hewlett-Packard, "HP Labs : Cacti," 2008. [Online]. Available: http://www.hpl.hp.com/research/cacti/

[189] B. Calhoun and A. Chandrakasan, "A 256kb Sub-threshold SRAM in 65nm CMOS," *IEEE International Solid State Circuits Conference*, pp. 2592–2601, 2006.

[190] K. Takeda, Y. Aimoto, N. Nakamura, H. Toyoshima, T. Iwasaki, K. Noda, K. Matsui, S. Itoh, S. Masuoka, T. Horiuchi, A. Nakagawa, K. Shimogawa, and H. Takahashi, "A 16-Mb 400-MHz loadless CMOS four-transistor SRAM macro," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 11, pp. 1631–1640, Nov. 2000.

[191] P. Bai, C. Auth, S. Balakrishnan, M. Bost, R. Brain, V. Chikarmane, R. Heussner, M. Hussein, J. Hwang, D. Ingerly, R. James, J. Jeong, C. Kenyon, E. Lee, S.-H. Lee, N. Lindert, M. Liu, Z. Ma, T. Marieb, A. Murthy, R. Nagisetty, S. Natarajan, J. Neirynck, A. Ott, C. Parker, J. Sebastian, R. Shaheed, S. Sivakumar, J. Steigerwald, S. Tyagi, C. Weber, B. Woolery, A. Yeoh, K. Zhang, and M. Bohr, "A 65nm logic technology featuring 35nm gate lengths, enhanced channel strain, 8 Cu interconnect layers, low-k ILD and 0.57 μm/sub 2/ SRAM cell," in *IEEE International Electron Devices Meeting*. IEEE, 2004, pp. 657–660.

[192] W.-h. Lee, A. Waite, H. Nii, H. Nayfeh, V. McGahay, H. Nakayama, D. Fried, H. Chen, L. Black, R. Bolam, J. Cheng, D. Chidambarrao, C. Christiansen, M. Cullinan-Scholl, D. Davies, A. Domenicucci, P. Fisher, J. Fitzsimmons, J. Gill, M. Gribelyuk, D. Harmon, J. Holt, K. Ida, M. Kiene, J. Kluth, C. Labelle, A. Madan, K. Malone, P. McLaughlin, M. Minami, D. Mocuta, R. Murphy, C. Muzzy, M. Newport, S. Panda, I. Peidous, A. Sakamoto, T. Sato, G. Sudo, H. VanMeer, T. Yamashita, H. Zhu, P. Agnello, G. Bronner, G. Freeman, S.-f. Huang, T. Ivers, S. Luning, K. Miyamoto, H. Nye, J. Pellerin, K. Rim, D. Schepis, T. Spooner, X. Chen, and M. Khare, "High performance 65 nm SOI technology with enhanced transistor strain and advanced-low-K BEOL," in *IEEE International Electron Devices Meeting*, vol. 00, no. c. IEEE, 2005, pp. 56–59.

[193] B. Agrawal and T. Sherwood, "Ternary CAM Power and Delay Model: Extensions and Uses," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, no. 5, pp. 554–564, May 2008.

[194] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of Network-on-chip," *ACM Computing Surveys*, vol. 38, no. 1, pp. 1–es, June 2006.

[195] P. Pratim Pande, A. Ganguly, B. Belzer, A. Nojeh, and A. Ivanov, "Novel interconnect infrastructures for massive multicore chips an overview," *2008 IEEE International Symposium on Circuits and Systems*, pp. 2777–2780, May 2008.

[196] ITRS, "International technology roadmap for semiconductors. Tech. rep," Tech. Rep., 2001.

[197] R. Ho, K. Mai, and M. Horowitz, "The future of wires," *Proceedings of the IEEE*, vol. 89, no. 4, 2001.

[198] K. Lee, S.-j. Lee, S.-e. Kim, H.-m. Choi, D. Kim, S. Kim, M.-w. Lee, and H.-j. Yoo, "A 51mW 1.6GHz on-chip network for low-power heterogeneous SoC platform," in *2004 IEEE International Solid-State Circuits Conference.* IEEE, 2004, pp. 152–518.

[199] R. Ho, K. Mai, and M. Horowitz, "Efficient on-chip global interconnects," in *2003 Symposium on VLSI Circuits. Digest of Technical Papers (IEEE Cat. No.03CH37408).* Japan Soc. Appl. Phys, 2003, pp. 271–274.

[200] R. Ho, T. Ono, R. D. Hopkins, A. Chow, J. Schauer, F. Y. Liu, and R. Drost, "High Speed and Low Energy Capacitively Driven On-Chip Wires," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 1, pp. 52–60, Jan. 2008.

[201] W. Bainbridge and S. Furber, "Delay insensitive system-on-chip interconnect using 1-of-4 data encoding," in *Proceedings Seventh International Symposium on Asynchronous Circuits and Systems. ASYNC 2001.* IEEE Comput. Soc, 2001, pp. 118–126.

[202] V. B. Mountcastle, "The columnar organization of the neocortex." *Brain : a journal of neurology*, vol. 120 (Pt 4, pp. 701–22, Apr. 1997.

[203] J. C. Horton and D. L. Adams, "The cortical column: a structure without a function." *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, vol. 360, no. 1456, pp. 837–62, Apr. 2005.

[204] N. M. da Costa and K. Martin, "Whose Cortical Column Would that Be?" *Frontiers in neuroanatomy*, vol. 4, no. May, p. 16, Jan. 2010.

[205] T. Binzegger, R. J. Douglas, and K. a. C. Martin, "A quantitative map of the circuit of cat primary visual cortex." *The Journal of neuroscience : the official journal of the Society for Neuroscience*, vol. 24, no. 39, pp. 8441–53, Sept. 2004.

[206] D. S. Modha, R. Ananthanarayanan, S. K. Esser, A. Ndirango, A. J. Sherbondy, and R. Singh, "Cognitive computing," *Communications of the ACM*, vol. 54, no. 8, p. 62, Aug. 2011.

[207] G. Blasdel, J. Lund, and D. Fitzpatrick, "Intrinsic connections of macaque striate cortex: axonal projections of cells outside lamina 4C," *J. Neuroscience*, vol. 5, no. 12, pp. 3350–3369, 1985.

[208] D. Fitzpatrick, J. Lund, and G. Blasdel, "Intrinsic connections of macaque striate cortex: afferent and efferent connections of lamina 4C," *J. Neuroscience*, 1985.

[209] M. A. Green, "Solar cell fill factors: General graph and empirical expressions," *Solid-State Electronics*, vol. 24, no. 8, pp. 788–789, Aug. 1981.

[210] M. Saen, K. Osada, Y. Okuma, K. Niitsu, Y. Shimazaki, Y. Sugimori, Y. Kohama, K. Kasuga, I. Nonomura, N. Irie, T. Hattori, A. Hasegawa, and

T. Kuroda, "3-D System Integration of Processor and Multi-Stacked SRAMs Using Inductive-Coupling Link," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 4, pp. 856–862, Apr. 2010.

[211] T. Sekiguchi, K. Ono, A. Kotabe, and Y. Yanagawa, "1-Tbyte/s 1-Gbit DRAM Architecture Using 3-D Interconnect for High-Throughput Computing," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 4, pp. 828–837, Apr. 2011.

[212] M. Wordeman, J. Silberman, G. Maier, and M. Scheuermann, "A 3D system prototype of an eDRAM cache stacked over processor-like logic using through-silicon vias," in *IEEE International Solid-State Circuits Conference*. IEEE, Feb. 2012, pp. 186–187.

[213] D. H. Kim, K. Athikulwongse, M. Healy, M. Hossain, M. Jung, I. Khorosh, G. Kumar, Y.-J. Lee, D. Lewis, T.-W. Lin, C. Liu, S. Panth, M. Pathak, M. Ren, G. Shen, T. Song, D. H. Woo, X. Zhao, J. Kim, H. Choi, G. Loh, H.-H. Lee, and S. K. Lim, "3D-MAPS: 3D Massively parallel processor with stacked memory," in *2012 IEEE International Solid-State Circuits Conference*. IEEE, Feb. 2012, pp. 188–190.

[214] D. Dutoit and C. Bernard, "A 0.9 pJ/bit, 12.8 GByte/s WideIO memory interface in a 3D-IC NoC-based MPSoC," *Symposium on VLSI Circuits*, pp. 22–23, 2013.

[215] W. Davis, J. Wilson, S. Mick, C. Mineo, a.M. Sule, M. Steer, and P. Franzon, "Demystifying 3D ICs: The Pros and Cons of Going Vertical," *IEEE Design and Test of Computers*, vol. 22, no. 6, pp. 498–510, June 2005.

[216] R. Patti, "Three-Dimensional Integrated Circuits and the Future of System-on-Chip Designs," *Proceedings of the IEEE*, vol. 94, no. 6, pp. 1214–1224, June 2006.

[217] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, Aug. 2014.

[218] R. B. Ivry and R. M. C. Spencer, "The neural representation of time." *Current opinion in neurobiology*, vol. 14, no. 2, pp. 225–32, Apr. 2004.

[219] J. D. Schmahmann, "Disorders of the cerebellum: ataxia, dysmetria of thought, and the cerebellar cognitive affective syndrome." *The Journal of neuropsychiatry and clinical neurosciences*, vol. 16, no. 3, pp. 367–78, Jan. 2004.

[220] T. Yamazaki and S. Tanaka, "Neural modeling of an internal clock." *Neural computation*, vol. 17, no. 5, pp. 1032–58, May 2005.

[221] ——, "A spiking network model for passage-of-time representation in the cerebellum." *The European journal of neuroscience*, vol. 26, no. 8, pp. 2279–92, Oct. 2007.

[222] J. Luo, G. Coapes, T. Mak, T. Yamazaki, C. Tin, and P. Degenaar, "Restoration of Cerebellum Passage-of-Time Functionality Using a Scalable FPGA-based Granular Layer," in *EMBC*, 2014.

[223] P. Teehan, M. Greenstreet, and G. Lemieux, "A Survey and Taxonomy of GALS Design Styles," *IEEE Design & Test of Computers*, vol. 24, no. 5, pp. 418–428, Sept. 2007.

[224] J. Harkin, F. Morgan, L. McDaid, S. Hall, B. McGinley, and S. Cawley, "A Reconfigurable and Biologically Inspired Paradigm for Computation Using Network-On-Chip and Spiking Neural Networks," *International Journal of Reconfigurable Computing*, vol. 2009, pp. 1–13, 2009.

[225] E. Stott and P. Y. K. Cheung, "Improving FPGA Reliability with Wear-Levelling," *2011 21st International Conference on Field Programmable Logic and Applications*, pp. 323–328, Sept. 2011.

[226] F. Galluppi, A. Rast, S. Davies, and S. Furber, "A general-purpose model translation system for a universal neural chip," *Neural information processing.*, pp. 58–65, 2010.