M.Sc. Thesis

Energy-efficient multipath ring network for heterogeneous clustered neuronal arrays

Ardelean Andrei B.Sc.

Abstract

Simulating large spiking neural networks (SNN) with a high level of realism in a field programmable gate array (FPGA) requires efficient network architectures that satisfy both resource and interconnect constraints, as well as changes in traffic patterns due to learning processes. Based on a clustered SNN simulator concept, in this thesis, an energy-efficient multipath ring network topology is presented for the neuron-to-neuron communication. It is compared in terms of its mathematical properties with other common network topology graphs after which the traffic distributions across it and a two dimensional torus network are estimated and contrasted. As a final characterization step, the energy-delay product of the multipath topology is estimated and compared with other low power architectures. In addition, a simplified binary tree is suggested as a network layer for handling configuration and input/output data that uses a custom channel protocol without the need for routing tables.

Energy-efficient multipath ring network for heterogeneous clustered neuronal arrays

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

MICROELECTRONICS

by

Ardelean Andrei B.Sc.
born in Timișoara, Romania

This work was performed in:

Circuits and Systems Group
Department of Microelectronics & Computer Engineering
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Delft University of Technology

Copyright © 2017 Circuits and Systems Group
All rights reserved.

DELFT UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF
MICROELECTRONICS & COMPUTER ENGINEERING

The undersigned hereby certify that they have read and recommend to the Faculty of Electrical Engineering, Mathematics and Computer Science for acceptance a thesis entitled “**Energy-efficient multipath ring network for heterogeneous clustered neuronal arrays**” by **Ardelean Andrei B.Sc.** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: 29.11.2017

Chairman:

prof.dr.ir. Alle-Jan van der Veen

Advisor:

dr.ir. Rene van Leuken

Committee Members:

dr.ir. Amir Zjajo

dr.ir. Stephan Wong

dr. ir. Carlo Galuzzi

Abstract

Simulating large spiking neural networks (SNN) with a high level of realism in a field programmable gate array (FPGA) requires efficient network architectures that satisfy both resource and interconnect constraints, as well as changes in traffic patterns due to learning processes. Based on a clustered SNN simulator concept, in this thesis, an energy-efficient multipath ring network topology is presented for the neuron-to-neuron communication. It is compared in terms of its mathematical properties with other common network topology graphs after which the traffic distributions across it and a two dimensional torus network are estimated and contrasted. As a final characterization step, the energy-delay product of the multipath topology is estimated and compared with other low power architectures. In addition, a simplified binary tree is suggested as a network layer for handling configuration and input/output data that uses a custom channel protocol without the need for routing tables.

Contents

Abstract	v
1 Introduction	1
1.1 Problem description	1
1.2 Objective	2
1.3 Contributions	2
1.4 Thesis outline	3
2 State of the art and background concepts	5
2.1 The neuron	5
2.2 Neuron models	5
2.3 Previous work	5
2.4 Other works	7
2.5 Communication network considerations	10
3 Neural network communication schemes and system structure	13
3.1 Metrics of interest	13
3.2 Neural communication schemes	14
3.2.1 Feed forward - FF	14
3.2.2 Complete hidden layer - CHL	16
3.2.3 Layered full lateral inhibition - LFLI	17
3.2.4 Layered neighbor lateral inhibition - LNLI	17
3.2.5 Variations	17
3.3 Results interpretation	18
3.4 Additional traffic components	20
3.5 Proposed physical network structure	22
3.6 Network topology graphs	22
4 Layered network	27
4.1 Extraction, insertion and configuration layer	27
4.1.1 Design criteria	27
4.1.2 Proposed topology	27
4.1.3 Routing scheme and router design	28
4.2 Topological layer	28
4.2.1 Multipath ring	30
4.2.2 Two dimensional torus	31
4.2.3 Traffic model	32
4.2.4 MATLAB implementation	37
4.2.5 Traffic estimation	38
4.3 Conclusion	44

5	Energy-delay product	47
5.1	Mathematical derivation	48
5.1.1	Multipath ring	49
5.1.2	Mesh	50
5.1.3	Tree and Inverse clustering	50
5.1.4	Inverse clustering with mesh	51
5.2	EDP estimation	52
6	Conclusion and future work	57
6.1	Conclusion	57
6.2	Future work suggestions	58
A	Parameter value table for Inverse clustering with mesh network	65
B	MATLAB	67
B.1	Rule structure	67

List of Figures

2.1	The three compartment based extended Hodgkin Huxley neuron model	6
2.2	Comparison of different neuron models in terms of computational complexity and biophysical accuracy.	6
2.3	A cluster as used in the proposed system	7
2.4	Dataflow of extended Hodgkin Huxley PhC. The box on the left is the axon and soma calculation. The box on the right is the dendrite calculation.	8
3.1	The representation of a biological network as a directed graph	14
3.2	Feed forward communication scheme	15
3.3	Complete hidden layer communication scheme	16
3.4	Layered full lateral inhibition communication scheme	18
3.5	Layered neighbor lateral inhibition communication scheme	19
3.6	Deep feed forward communication scheme	19
3.7	Average degree as a function of the number of hidden neurons in the four communication schemes	21
3.8	Histograms of output degrees	21
3.9	Proposed system structure	22
3.10	Comparison of topology properties	24
4.1	Binary tree structure of the Extraction, insertion and configuration layer	28
4.2	Steps in the routing scheme	29
4.3	Router structure for downstream traffic with a parallel data bus	29
4.4	Multipath ring topologies of size 13 with various skip values	30
4.5	A routing scheme for the multipath ring topology. The red arrows indicate the path taken by a message originating from cluster zero to every possible destination cluster.	32
4.6	A routing scheme for the two dimensional torus topology. The red arrows indicate the path taken by a message originating from cluster five on its path to every possible destination cluster.	33
4.7	Cluster C0 metrics associated with the two analyzed topologies	34
4.8	All the metrics in a size 5 and skip 2 MPR network	35
4.9	The feed forward scheme of size 8-10-2	38
4.10	Average maximum throughput values	40
4.11	Average throughput for exterior links in the MPR network	41
4.12	Average throughput for interior links in the MPR network	42
4.13	Average difference between maximum and minimum throughput in the MPR and 2D torus network	43
4.14	The system structure with a split topological layer	44
4.15	The average maximum throughput values in the MPR and 2D torus network in the case of a split topological layer	45

4.16	The average maximum throughput values in the MPR and 2D torus network in the case of a split topological layer with different threshold values for LNLI and CHL	45
4.17	The effect of the placing algorithm on average throughput in the MPR and 2D torus network	46
5.1	The energy and delay components that make up the path from cluster i to j	49
5.2	Comparison between binary tree and inverse clustering networks of size 8. The numbers on the links indicate proportionality factors between each corresponding E_{link}	50
5.3	The inverse clustering with mesh network topology and the associated E_{link} proportionality factors. The inset shows the same network arranged as a mesh with the connections in the first inverse cluster layer visible.	51
5.4	EDP estimations for different network topologies of 32 clusters vs. Manhattan distance for different delay weights	53
5.5	Energy-delay product as a function of index difference in the multipath ring of size 32	54
5.6	Global energy-delay products for all communication schemes with traffic resulting from random neuron placements	55

List of Tables

2.1	Common network topologies	12
3.1	Maximum input and output neuron degrees	20
3.2	Aproximated maximum input and output neuron degrees for large networks	20
3.3	Mathematical characteristics of various network topologies	23
4.1	The placement of neurons inside clusters	37
4.2	Two dimensional torus network layout for various network sizes	39
A.1	Parameter values for Inverse clustering with mesh network of size 32 . .	66

Introduction

1.1 Problem description

Spiking neural networks (SNN) have become increasingly more attractive in current scientific research due to their ability to closely mimic biological neural behavior such as the encoding of information in the timing of spikes and their amplitude and in spike train patterns and transfer rate [1]. However, such high level of realism comes at the cost of substantial computational effort and high throughput demands.

Multiple neuron models such as leaky integrate-and-fire [2], Izhikevich [3] and Hodgkin Huxley [4] have been developed and implemented in hardware, with each providing different levels of realism. However, the need to create flexible simulating hardware capable of implementing at the same time multiple model types with large number of neurons persists as current implementations focus on a single neuron model.

Because neural networks present a high level of parallelism in their behavior, field programmable gate arrays (FPGA) prove to be suitable platforms for SNN simulation [5]. In addition, they also benefit from inherent flexibility that allows them to change simulated network topologies and neuron models with ease. However, the interconnect fabric in FPGA is limited in terms of availability, power and delay [6]. These downsides come from the flexible architecture - a LUT implementation of a function will always be larger than the ASIC variant which leads to inefficient area usage. Also, switches between the interconnect links are very large as they are designed to be configured in all possible permutations, but only serve one in each implementation. To mitigate these issues, network topologies have to be developed that efficiently use FPGA resources.

Problems when simulating spiking neural networks on FPGA also arise from the limited number of available memory blocks. The large number of neuron to neuron connections with their associated parameters (such as weights) require large amounts of memory and serve as bottlenecks in FPGA implementations [5, 7]. To overcome this, some have proposed the use of external memory banks [8, 9], but this can limit the system performance because of longer access times.

In addition to the aforementioned issues, neuron-to-neuron communication schemes are continuously changing during simulation time because of learning processes [10]. Connections between neurons that existed at the beginning of the simulation can disappear while other new ones can be formed. Assuming an intelligent placement of the neurons based on an initial communication scheme, the resulting changes can mitigate the effect of all the optimizations. These modifications in traffic can lead to inefficiencies in the overall system (such as inadmissible latency and loss of packets due to contention) if the simulating platform is not flexible enough to accommodate them. Solutions might include the use of dynamic routing algorithms or neuron placement reconfiguration, but also the design of efficient network architectures that can man-

age such changes due to their topological properties (small diameter, high degree of connectivity etc.).

Also, with efficient computing blocks for neuron models being developed that use hard cores in the FPGA fabric (for example DSP slices) and can function at higher clock rates, large scale real-time neuron simulations are no longer computational but communication bound [9]. This puts even more importance on the development of highly efficient network topologies that lead to less node-to-node communications due to their inherent characteristics.

1.2 Objective

Previous effort has been made to reduce throughput and resource costs by grouping neuron simulating hardware, memory blocks and control circuitry into units referred to as *clusters* [11, 12, 13]. The reuse of hardware for computational intensive mathematical operations such as exponentiation has successfully resulted in a reduction of resource consumption without increasing overall latency. Moreover, the sharing of memory blocks and exploiting data locality has decreased the required throughput in the cluster-to-cluster interconnect.

The objective of this work is to determine an efficient cluster interconnect network architecture that can be implemented on FPGA and offers high energy-efficiency at low implementation costs and also has enough flexibility to accommodate the continuously changing neuron-to-neuron communication scheme. Different topologies are compared in terms of throughput and energy-delay product in order to determine the one with better inherent properties. The mathematical properties of network topology graphs such as clustering coefficient and diameter are also explored and used in the selection process.

1.3 Contributions

The contributions made by this work can be summarized as follows:

- The characterization of a multipath ring topology as an energy-efficient dataflow architecture - the multipath ring was initially compared with other common topologies based on mathematical analysis of their graph representations. The two topologies that presented the most attractive characteristics (this one and the two dimensional torus) were then compared in terms of traffic distribution for various network sizes and communication schemes. Finally, the energy-delay product of the multipath ring and other topologies was estimated and compared.
- The simplified binary tree used for input/output data and the configuration of neurons - based on the specific characteristics of this type of data traffic, optimizations can be made that allow saving of hardware resources as well as reduce protocol overhead. The end result is a network structure that does not require routing tables and that is comprised of simple routers.

- The development of a simple and abstract method of estimating throughput in a network - for an objective comparison that focuses only on network topology and not on other protocol or physical implementation related specifics, the network behavior was transposed into an abstract model. This was then used to estimate traffic distribution throughout the network so that the candidate topologies could be evaluated.
- The development of a simple and abstract method for estimating energy-delay product in a network - the multipath ring was compared with other topologies also in terms of energy and delay. Not having an actual physical implementation, energy and delay parameters are unknown but can be estimated by abstract components that are then used to approximate the final result.

1.4 Thesis outline

The thesis is organized as follows: Chapter 2 summarizes previous research on which this thesis is based, basic concepts related to the studied topic and state of the art research in the field, while Chapter 3 presents the analyzed neuron communication schemes and defines the proposed system structure as well as assumptions on which it is based. Chapter 4 focuses on the system architecture and its analysis and compares the multipath ring and two-dimensional torus network topologies. The energy-delay product estimation for different network topologies is presented in Chapter 5, while Chapter 6 concludes the thesis.

2.1 The neuron

The *neuron*, also called a *nerve cell*, is a fundamental component of the central and peripheral nervous system in all living beings. It is an electrically excitable cell that receives, processes and then transmits information in the form of electrical and chemical signals.

Neurons mainly consist of three parts, the *dendrites*, through which inputs from other neurons are received, the *soma* or the cell body and the *axon* which acts as the neuron output. A voltage gradient is maintained across the cell membrane due to varying ion concentrations in the intra and extra cellular medium. If the voltage across the membrane changes by a significant amount, an electrochemical pulse known as *action potential* is generated and propagated along the axon activating the connections with other neurons. These connections between neurons are called *synapses*, with cells in the human brain being capable of reaching 10000 connections with their neighbors. A series of interconnected neurons is referred to as a *neural network*.

2.2 Neuron models

As mentioned in the Introduction, a multitude of models have been proposed to describe the behavior of a neural cell, but this work will be based on a system designed for the extended Hodgkin Huxley [14]. This model describes the *Inferior Olivary Nucleus (ION)* as a multiple compartmental cell, with each compartment having a state potential updated every simulation cycle that depends on dendritic potentials and ion currents. The model structure is shown in Figure 2.1. It includes an extra compartment to model the axon hillock as well as the possibility to connect the dendritic compartment to other neighboring cells.

Compared with other neuronal models, the Hodgkin Huxley is the most biophysically accurate (Figure 2.2) as it describes the membrane potential and activation of ion currents. It can also exhibit all behaviors of biological spiking neurons (tonic spiking, phasic spiking, bursting etc.) by tuning the tens of parameters that it comprises.[15] However, this level of realism requires a large computational effort and very few neural simulators implement it for cases where large number of neurons are desired.

2.3 Previous work

This thesis is based on work done in [11, 12, 13] where a hardware implementation of the extended Hodgkin Huxley model for the Xilinx Virtex 7 FPGA is presented with the overall system achieving real-time operation with a $50\mu s$ simulation step. Data

Figure 2.1: The three compartment based extended Hodgkin Huxley neuron model.[14]

Figure 2.2: Comparison of different neuron models in terms of computational complexity and biophysical accuracy.[15]

Figure 2.3: A cluster as used in the proposed system [12].

locality is exploited, where neurons are connected with decreasing probability as the distance between them increases. Neighboring neurons are grouped in structures called *clusters* composed of double floating point computational units (*Physical Cell - PhC*), shared memory blocks and controlling circuits (Figure 2.3). The cluster allows for instant communication between any cells that are located inside it. Each of the PhC time-shares the calculation of multiple cells to save resources as the simulation step is larger than the FPGA operating frequency (period). [12]

The PhC dataflow is shown in Figure 2.4 where the splitting of calculations in two groups is visible. The first group is associated with the soma and axon hillock compartments which are independent of the neighboring neurons. The dendritic calculations, which depend on coupling between neurons, are computed in the second calculation group.

Each cluster has three controllers, one that handles internal bookkeeping, one for incoming data packets and one for outgoing data packets and memory management. The system is designed in such a way that the activity of each cluster is independent on the activities of other clusters in the network. It sends its outputs in the network untargeted and it is up to every other cluster to determine if the packets are needed for their computation.

Based on the presented design, this work will assume that all intra-cluster neuron-to-neuron communications are already efficiently handled by the cluster and thus the network will only need to handle inter-cluster communications and interactions between the user and system, such as configuration and injection of stimuli.

2.4 Other works

Apart from the implementation presented in section 2.3, several approaches have been suggested for simulating SNN with various degrees of biological realism on multiple platforms such as CPU and/or GPU, analog-digital hybrids, ASIC and FPGA. Among them it is worth mentioning:

Figure 2.4: Dataflow of extended Hodgkin Huxley PhC [11]. The box on the left is the axon and soma calculation. The box on the right is the dendrite calculation.

- **SpiNNaker** [16] is a massively parallel computer that is targeted to accommodate up to one million ARM microprocessor cores with a communication architecture optimized for carrying large numbers of small data packets. The microprocessor cores are distributed throughout 57000 nodes [17], each one containing 18 ARM cores, 128 MB of shared memory, 96kB of local memory, a router and general purpose circuitry. Six communication links connect each node in a triangular fashion folded on the surface of a toroid [17] with a routing scheme based on routing tables with 1024 entries. The system is designed to simulate leaky integrate-and-fire or Izhikevich neuron models with the order of 1000 input synapses to each neuron, one ARM core being capable of modeling 100 neurons and theoretically reach 10 million connections/s [16].
- **Neurogrid** [18] is a hybrid architecture that consists of Neurocores, a combination of 256x256 analog neuron cells with digital routing logic and memory. The communication network topology is a binary tree for which a version of up-down routing with a point-to-point and then branching phase was developed in order to guarantee that the network is deadlock-free [19]. The routing path is encoded in the packet header with each up or down and left and right turns represented by a single bit, depending on the phase in which the packet is at that moment.
- **TrueNorth** [20] is a digital ASIC implementation of a SNN developed by IBM, consisting of 4096 neurosynaptic cores that each simulate 256 integrate-and-fire neurons with 256 configurable synapses. The cores are distributed in a two-dimensional mesh where X-Y routing is employed to eliminate deadlocking. The end result is a energy-efficient system that can be easily scaled to multi-chip configurations.
- **NeptonCore** [21] is a digital neural core implemented on a Xilinx Virtex 4 FPGA that simulates a simplified version of the Hodgkin Huxley model called the Huber-Braun model [22] that only focuses on clinically relevant measures. In

total, 1600 neurons are simulated with a model step size of $100\mu\text{s}$. [23] However, the design is still at the stage of having only one core that takes up around 16% of FPGA resources.

- **PLAQIF** (piecewise-linear approximation of quadratic integrate and fire) is a digital neuron model implemented by [24] on a Xilinx Virtex 5 FPGA. A network consisting of 161 neurons and 1610 synapses was simulated at a 1ms simulation step. Due to a design decision to make the dendrite structure flexible during learning processes, FPGA resources are heavily used, with the design occupying 85% of the device.
- [25] is a successful FPGA implementation of an Izhikevich model based SNN used for character recognition. The design uses 25 processing elements (PE) to compute all the required calculations for a total of 9000 neurons. The design was implemented on a Xilinx Virtex II Pro FPGA and consumed 79% of available resources. The disadvantage of this architecture is that all computations are in fixed point representation.
- **Brainbow** [7] is a project aimed to simulate Izhikevich modeled neurons using a 1ms time step. A computation core is built around a single multiplier in a Xilinx Virtex 4 FPGA that is then used to simulate a network of 117 neurons with only 3.3% resource usage. However, the limitations of this architecture arise from the memory usage and as a result, future implementations would require external RAM memory modules.
- [8] presents a 1 million Integrate and Fire neuron network simulated on a Xilinx Virtex 5 FPGA. As with previously mentioned FPGA implementations, the architecture consists of multiple computational cores that each simulate a number of neurons in a time multiplexed fashion. The main difference of this approach is that synaptic weights and output connection addresses are implemented using off chip RAM which allows for a much higher memory density.
- **Bluehive** [9] is a multi chip computing machine for real-time SNN simulation. It is designed around the Izhikevich model and employs a 16 bit fixed point arithmetic. A first implementation of the system using 4 Altera Stratix IV FPGA was created, which simulates 256k neurons and 256M synapses. As with [8], neuron model and synaptic parameters are stored in off chip RAM modules. Four processing engines are implemented on each FPGA in a small tree network, while the topology of choice for the multi FPGA network is a 3D torus.
- **Minitaur** [26] is an event-driven FPGA based SNN accelerator that uses leaky integrate and fire neuron models. 32 *minitaur* cores with 2 DSP blocks each are implemented on a Xilinx Spartan 6 FPGA that is controlled by a PC. The cores are connected to a common 128MB RAM block and receive their tasks from a parallel distributor unit. The spike events arrive via USB and after Minitaur computes the output spikes they are sent back via USB to the PC. The system was tested with 10k neurons with 200k synapses and 1785 neurons with 647k synapses for

hand written digit recognition and text classification. The overall system latency is limited by the operating system but reduced power consumption was observed.

- [5] represents a framework for designing large scale SNN simulations. The architecture is targeted towards a Xilinx Virtex 6 FPGA and uses the leaky integrate and fire neuron model. The fundamental architectural block is the layer, which contains buffers, SRAM and computational blocks for integrating impulses and performing soma/axon calculations. Each layer can handle all the computations for 64k time multiplexed neurons. The implemented system contains 23 layers connected in a serial fashion with all-to-all connections between neurons, totaling 1.5M neurons and 92G synapses.

Overall, the various implementations mentioned above each present specific advantages and disadvantages that arise from the platform of choice. ARM and GPU based systems benefit from the flexibility of being able to change neuron models but lack the ability to efficiently map the highly parallel nature of SNN and as a result are also the slowest implementations. ASICs provide the base for the most energy-efficient systems that also exhibit the highest simulation speeds but are extremely rigid, as the neuron models are fixed.

A good compromise between the two are FPGA based platforms that are capable of efficiently executing parallel computations while at the same time allowing a reconfiguration of the networks as desired by the user. The limited number of available resources, however, restricts the network size but this can be overcome by multi-chip solutions.

The majority of available systems now use simple models such as integrate and fire or Izhikevich that do not provide accurate data regarding the internal behavior of the neuron, instead focusing on its input and output behavior. As a result, these are the models of choice because they can be easily implemented and very large neural networks constructed. The challenge of simulating highly biological accurate neurons using the Hodgkin Huxley model in large numbers on FPGA remains.

2.5 Communication network considerations

The communication network is tasked with delivering data packets between all the clusters in the system in an efficient manner. It consists of a number of routers interconnected with links atop of which a routing protocol manages packet delivery and is encoded in the routers themselves.

The links are composed of wires and can be categorized in two types parallel and serial. The parallel ones can be operated at low clock rates to reduce power consumption but use large amounts of hardware resources especially when data has 32 or 64 bit format as in this case. Contrastingly, serial links operate at high clock speeds and require serialization and de-serialization hardware but consume less interconnect resources. The links ultimately define the performance and power consumption in the network and the main design objectives are to provide fast, reliable and low-power interconnects between the nodes of the network.[27]

The multipath ring characterization done in this thesis will not define the type of links but instead will make them more abstract by considering the link as an ideal medium for transmitting data from one end to another. The actual implementation type can be selected once the throughput is determined. However, when estimating delay and power consumption of various network topologies, the links will be given delay and energy metrics.

Routers are usually composed of a number of ports and a switch matrix. The ports can be further classified in network ports that are connected to links in the network and local ports that are dedicated to the IP core attached to the router, in this case the cluster. Apart from the physical interconnect infrastructure, additional circuitry is present that handles the flow of data through the router based on a set of predefined rules called the *routing protocol/scheme*.

Performance of a network can be evaluated through a number of parameters:

- *Bandwidth* - the maximum rate of data propagation measured in bits per second (bps). This parameter will not be used in this work because the physical network layer is not implemented yet so no protocols are defined.
- *Throughput* - the maximum traffic accepted by the network measured in messages per clock cycle.[28] Throughout this thesis, throughput will be measured in messages per simulation step.
- *Latency* - the time it takes for a message to be completely received by the target, counting from the beginning of transmission, measured in time units. This parameter will be used to rank different network graphs as a first characterization step of the multipath ring. It will not be directly calculated but will be qualitatively determined based on mathematical properties such as average path length and network diameter.

The overall structure of router connections is referred to as a *topology* and it can be *direct*, where each router is connected to an IP core (cluster) and *indirect* where some routers are only used to propagate messages through the network while others are connected to cores and can serve as message sources or destinations.[27] Both topology types are analyzed in the thesis but networks with the former type are preferred because they are more hardware efficient. For the same reason, *regular networks* where every router has the same number of ports are the only ones analyzed.

Table 2.1 contains a collection of network topologies along with their advantages and disadvantages. Included among them is also the multipath ring and two dimensional torus that will be further described in sections 4.2.1 and 4.2.2.

The routing protocol is the set of rules based on which a router output port is selected for a specific message that needs to pass through. Usually, this implies the use of routing tables and address information present in the packet header. Two routing categories can be identified: *static routing* in which the path a message follows between two nodes is always the same and is determined before runtime and *dynamic routing*, in which the routing tables can be modified during runtime depending on the current status of the network (if for example the original path is congested). The analysis done

Table 2.1: Common network topologies

Type	Description	Advantages	Disadvantages
Ring	Every router is connected to two others and a circular path is formed.	-Simple routing scheme -Simple HW implementation	-Large diameter -Deadlock possibility
Multipath ring	Similar to the ring but with an extra set of connections between routers that are further apart.	-Small diameter -Flexible -Simple HW implementation	-Larger average path length for big networks
Binary tree	There is only one connection between any two connected routers, forming a parent-child hierarchy. Each node has two children.	-Small diameter -Recursive structure	-Long interconnects between routers at root increase delay and power consumption
Mesh	Routers are arranged in an orthogonal array and each connected to its immediate neighbors.	-Efficient layout implementations -Simple routing	-Large diameter
Two dimensional torus	Similar to the mesh but the routers on the edges have wrap-around connections.	-Small diameter -Simple routing	-Longer delay in wrap-around connections

in this thesis will presume that static routing is used which allows simple mathematical methods to be applied when estimating traffic.

The state of the art implementations presented in the previous section employ three network topologies: binary tree, mesh and torus with both static (Neurogrid and TrueNorth) and adaptive (SpiNNaker) routing. The use of adapting routing allows the system to more efficiently utilize available resources at runtime but these optimizations are limited by the network structure. Also, highly adaptive algorithms are expensive in terms of virtual channels and as a result have large area overhead. Reducing the number of virtual channels would reduce the efficiency of the approach.[\[29\]](#)

An interesting alternative would be to change the network topology as changes in traffic patterns take place, such as the addition or removal of routers in the network [\[30\]](#) or changing link positions [\[31\]](#). Unfortunately these approaches consume a lot of hardware resources and represent an over complication of the network topology. If, however, a flexible architecture that requires only small number of reconfigurable links and routers to manifest large changes in traffic distribution were developed, these techniques could be implemented with less hardware overhead. Investigations in terms of traffic redistribution in the multipath ring were carried out that indicate towards a flexible architecture that can represent a good candidate for these kinds of applications.

Neural network communication schemes and system structure

3

A set of preliminary analysis steps are necessary in order to explore the design space of the system. There are two factors that must be taken into account:

- The neural network communication scheme is different from the physical network implementation due to hardware constraints. As noted in section 2.1 neurons can reach up to 10000 synapses each which makes it difficult to translate them into hardware representations, let alone networks that contain large numbers of neurons, because the end result would be slow or would require a very large amount of resources. Also, section 2.3 showcased a number of optimizations in the form of clusters that are used to boost system performance, structures that have no biological equivalent.
- The neuron communication scheme will dynamically change during runtime in an uncontrolled fashion. Throughout the simulation process, multiple parameters such as concentration levels used in the model and cell-to-cell connections change as a result of user intervention or learning processes undergone by the network. These changes have a significant impact on traffic behavior and if the physical network cannot handle them, they can lead to deadlocking, loss of packets and inability to maintain real-time operation.

For simplification purposes, the neural network communication scheme is characterized by a set of simple metrics presented in the following section that are then used as guidelines for designing the overall system structure.

3.1 Metrics of interest

The neuron-to-neuron communication scheme is commonly represented in the form of a graph in which a neuron is represented by a node (vertex) and a connection is a directed edge (Figure 3.1). The effect a neuron has on the network is linked to the number of connections it has with other neurons, a number equal to the degree d_i of the node, where i is the neuron (node) index.

Typically, neurons can be classified in separate categories (or groups called *layers*) according to their function. [32] Neurons that receive stimuli from sources not in the network are called *input neurons* and those who send responses outside of the network are appropriately called *output neurons*. Between them, *hidden neurons* comprise the neural network bulk and do not interact with the outside.

The average degree d_{AVG} can be computed as a function of the total number of neurons N and the number of neurons in the input, hidden and output layers: N_I , N_H and N_O . If directionality is also considered, the degree is split into its two components: c_+ , the number of input connections and c_- , the number of output connections. It is

Figure 3.1: The representation of a biological network as a directed graph

clear that (3.1) stands (note that the indexes have been dropped to make the equation easily readable). The maximum values of these two parameters are linked to the worst case scenario with respect to the number of data packets being sent in the network.

$$d = c_+ + c_- \quad (3.1)$$

3.2 Neural communication schemes

The multipath ring characterization was performed with four types of neuron communication schemes, which are presented in the following subsections. For simplicity, all the neurons are denoted by I for the input layer, H for the hidden layer and O for the output one. Connections between neurons are represented by arrows that indicate the directionality of the link.

3.2.1 Feed forward - FF

Arguably the most common [33], the *feed forward* scheme (Figure 3.2) is characterized by a signal path that goes from the input to the output layer through a number of hidden layers. Because the information flow through the network is strictly in one direction, with no cycles or loops being allowed, the outputs only depend on the current state of the inputs.

Each I neuron output is connected to all the H neurons' inputs in the hidden layer immediately following the input layer, and each O neuron's input is connected to all H neurons' outputs in the hidden layer preceding the output. In the case of multiple hidden layers, the same rules apply, with H neurons from one layer taking their inputs from all the neurons in the preceding layer and transmitting their outputs to all the neurons in the following layer.

To compute the average degree, we can start from (3.2) where N is the total number of neurons and N_I , N_H and N_O are the number of input, hidden and output neurons

Figure 3.2: Feed forward communication scheme

respectively.

$$N = N_I + N_H + N_O \quad (3.2)$$

Based on the feed forward characteristics, the degrees of every neuron category are known:

$$\begin{aligned} d_I &= N_H + 1 \\ d_H &= N_I + N_O \\ d_O &= N_H + 1 \end{aligned} \quad (3.3)$$

which can be used together with (3.2) to compute the average degree:

$$\begin{aligned} d_{AVG} &= \frac{N_I d_I + N_H d_H + N_O d_O}{N} = \\ &= \frac{N_I(N_H + 1) + N_H(N_I + N_O) + N_O(N_H + 1)}{N} = \\ &= \frac{2N_I N_H + 2N_O N_H + N_I + N_O}{N} = \\ &= \frac{(N_I + N_O)(2N_H + 1)}{N} = \frac{(N - N_H)(2N_H + 1)}{N} = \\ &= \frac{-2N_H^2 + N_H(2N - 1) + N}{N} \end{aligned} \quad (3.4)$$

Figure 3.3: Complete hidden layer communication scheme

3.2.2 Complete hidden layer - CHL

The feed forward communication scheme is incapable of mimicking biological processes such as memory because the outputs are only dependent on the current inputs. Conversely, in recurrent neural networks neurons are allowed to have connections that form a cycle, either by connecting to their own inputs or to the inputs of neurons in the previous layer. This allows the network to exhibit dynamic temporal behavior by using the internal "memory" to process an arbitrary sequence of inputs.

The most prominent example of such a topology is the Hopfield network [34] in which all the neurons are connected with each other but not back to themselves. All connections are symmetric i.e., the data transfer between two nodes is bidirectional.

The bidirectional associative memory (BAM) is a type of recurrent neural network but as opposed to Hopfield it can provide output patterns of different sizes compared to the input. BAM contains two layers of neurons that are fully connected to one another and repeatedly send signals back and forth until an equilibrium is reached. Either layer can be used as input or output.

In this analysis, recurrent neural networks will be represented by the *complete hidden layer* scheme. As the name suggests, it has a hidden layer composed of neurons that are all connected to each other with bidirectional links as well as having their inputs and outputs connected to the I and O neurons respectively. By adjusting weights in the fully and bidirectionally connected hidden layer, all recurrent neural networks can be modeled so there is no need to include their schemes in the analysis.

The presence of the tightly connected component in the scheme will cause significant strain on the network topology as all hidden neurons have very large degrees. An example of a complete hidden layer scheme is shown in Figure 3.3, with the average degree shown in (3.5). The thicker arrows that originate or are directed towards the hidden layer represent the N_H individual connections between every hidden neuron and the corresponding input/output one.

$$d_{AVG} = \frac{-N_H^2 + N_H(2N - 2) + N}{N} \quad (3.5)$$

3.2.3 Layered full lateral inhibition - LFLI

So far, the two presented neuron communication schemes contain what is called excitatory neurons. An excitatory synapse increases the probability of an action potential being generated in the targeted neuron. In contrast, research in biology has also detected inhibitory behavior between neurons [35] where the synapse decreases the action potential probability.

To model this, inhibitory neurons are introduced into the communication scheme of the feedforward network. CHL could have also been chosen but due to its inherently large number of connections the extra inhibitory neurons would not have had a significant effect on the estimated traffic. The intended behavior is as follows: once a neuron from the hidden layer fires an action potential, the inhibitory neuron connected to it will fire and will stop the other neurons from following suit.

In the *layered full lateral inhibition* scheme, the neurons in the hidden layers are split into two groups: those with normal behavior and those with inhibitory functions. For each normal neuron (H) in the hidden layer, there exists an inhibitory one (H') that takes its input from the former and outputs to all the other non inhibitory neurons in the layer (Figure 3.4).

For computing the average degree, N_H will account for the total number of neurons in the hidden layer, both normal and inhibitory, in order to keep (3.2) true. The average degree is in this case:

$$d_{AVG} = \frac{N_H(2N - 1) + N}{N} \quad (3.6)$$

3.2.4 Layered neighbor lateral inhibition - LNLI

In order to more closely mimic biological behavior where the inhibitory influence is manifested in regions of the neural network and not across its entirety, the number of output connections from the inhibitory neurons can be limited.

The *layered neighbor lateral inhibition* is similar to the full inhibition one described previously, with the change that an inhibitory neuron's outputs only connect to the immediate neighbors of its paired normal behavior neuron (Figure 3.5). The average degree is expressed in (3.7).

$$d_{AVG} = \frac{-2N_H^2 + N_H(2N + 2) + N + 2}{N} \quad (3.7)$$

3.2.5 Variations

All the communication schemes presented in the previous subsections consisted of a single hidden layer. It is of course possible to extend them to multiple layers and form what is referred to as *deep neural networks* like the one shown in Figure 3.6. However, throughout this thesis communication schemes were limited to having only one large hidden layer to simplify the mathematical analysis but the software platform can easily incorporate these cases as well.

In addition, there is one more communication scheme usually used in simulations - the *randomly connected neuron network (RNDC)*. It is defined by random exponential

Figure 3.4: Layered full lateral inhibition communication scheme

local connectivity, which means each neuron sends its spikes to a small set of neighbors [36]. Although this represents a realistic scenario it will not be considered in this analysis as it is not comparable with the other four communication schemes (no layered structure) and the results are not reproducible due to the random neuron connection assignments.

3.3 Results interpretation

Figure 3.7 shows the average degrees (a measure of how tightly the neurons are interconnected and as a consequence the network hardware requirements) of the four communication schemes as the number of neurons in the hidden layer increases from 10% to 90% of the total number of neurons N . The vertical axis is in percentages of N as well in order to normalize the results. Note that for the inhibitory schemes, only the number of normal behavior hidden neurons is taken as input and as a consequence, the two graphs only reach half of the range. This is done in order to make the networks

Figure 3.5: Layered neighbor lateral inhibition communication scheme

Figure 3.6: Deep feed forward communication scheme

Table 3.1: Maximum input and output neuron degrees

	Maximum c_+	Maximum c_-
FF	N_I, N_H	N_O, N_H
LFLI	$N_I + N_H/2 - 1$	$N_O + 1, N_H/2$
LNLI	$N_I + 2, N_H/2$	$N_O + 1, N_H/2$
CHL	$N_I + N_H - 1$	$N_O + N_H - 1$

Table 3.2: Aproximated maximum input and output neuron degrees for large networks

	Maximum c_+	Maximum c_-
FF	N_H	N_H
LFLI	$N_I + N_H/2$	$N_H/2$
LNLI	$N_H/2$	$N_H/2$
CHL	$N_I + N_H$	$N_O + N_H$

more similar to the FF scheme where the hidden neurons communicate with both the previous and following layers, a trait which is not shared by the inhibitory neurons.

As expected, CHL and LFLI have the highest average degrees that increase rapidly as the hidden layer becomes larger, but fortunately, these two topologies are the least likely to appear during simulations as inhibition tends to affect local neuron groups in biological cases[37] and tightly interconnected networks like CHL may be unstable and difficult to train [33]. The other two topologies have a maximum average degree of around 50% of N which puts relatively relaxed constraints on the physical network implementation.

Table 3.1 shows the maximum input and output degrees for the four different communication schemes. If we consider the network to be very large, then we can assume (3.8) and the new values for the nodal degrees are shown in Table 3.2.

$$N_H \gg N_O \approx N_I \quad (3.8)$$

By plotting the output degree c_- histograms (Figure 3.8) a pattern can be observed: there are two distinct groups of neurons, one that is comprised of neurons with a small number of output connections and one with neurons with a large number. The existence of these two distinct behaviors suggests that a separation of the traffic across two layers would allow for higher efficiency to be achieved. This, however, comes at the high cost of physical resources and power and further analysis is required in order to establish if the gains are high enough to justify these losses (Section 4.2.5.2).

3.4 Additional traffic components

Apart from the neuron-to-neuron communications, data traffic through the system also contains configuration packets, input stimuli and output data, all of which have specific characteristics:

- *Configuration data* is sent during setup or between simulation steps. It consists

Figure 3.7: Average degree as a function of the number of hidden neurons in the four communication schemes

Figure 3.8: Histograms of output degrees

of initialization or update values of neuron model parameters with each targeted cluster receiving up to 29 double precision floating point packets per neuron [11]. They represent ion concentrations and currents along with axon and soma potentials.

- *Input (stimulus) data* is sent between simulation steps, it originates from a system

Figure 3.9: Proposed system structure

controller (user) and is targeted towards input neurons. Analogous to the configuration data, it consists of multiple floating point packets sent to each cluster. They represent voltage samples of various impulses or trains of impulses.

- *Output data* is sent during runtime towards the user and it originates from specific neurons/clusters previously selected by her/him. It consists of various parameters' values and has packets of varying size.

Because these behaviors have little in common with the neuron-to-neuron communication and are also deterministic in nature, it is worthwhile to create a physical layer dedicated to them that will have a more suited topology that increases overall system efficiency.

3.5 Proposed physical network structure

Based on the above mentioned reasons, the suggested system topology, shown in Figure 3.9, consists of two separate layers: *Extraction, insertion and configuration layer* which handles the output/input data transfers and the *Topological layer* which handles the neuron to neuron communication.

These two layers are physically independent and only connect through the routers dedicated to each cluster. They have distinct topologies and follow different routing protocols which will be described in the following chapter.

3.6 Network topology graphs

There exists a wide range of topologies that can be used for either one of the system layers, each with its own set of advantages and disadvantages (Section 2.5). In this

Table 3.3: Mathematical characteristics of various network topologies

	Diameter	Clustering coefficient	Average node degree	Number of links	Number of routers	Average path length
Ring	$\lfloor \frac{N}{2} \rfloor + 2$	0	3	$2N$	N	Computed with MATLAB
Multipath ring skip 1	$\lfloor \frac{N}{4} \rfloor + \lfloor \frac{N}{2} \rfloor \bmod 2 + 2$	0.5	5	$3N$	N	
Multipath ring skip 2	$\lfloor \frac{N}{6} \rfloor + \lfloor \frac{N}{2} \rfloor \bmod 3 + 2$	0	5	$3N$	N	
Multipath ring skip 3	$\lfloor \frac{N}{8} \rfloor + \lfloor \frac{N}{2} \rfloor \bmod 4 + 2$	0	5	$3N$	N	
Binary tree	$2\log_2 N + 2$	0	3	$2N - 2$	$N - 1$	
Mesh	$2(\sqrt{N} - 1) + 2$	0	5	$3N - 2\sqrt{N}$	N	
Two-dimensional torus	$\sqrt{N} + 1$	0	5	$3N$	N	
Complete graph	3	1	N	$\frac{N^2 + N}{2}$	N	

subsection, multiple such interconnect schemes will be analyzed based on the mathematical properties extracted from their graph representations. A total number of N clusters is assumed.

The following graph topologies are analyzed:

- *Ring* - a circular graph, where each node is connected to two others.
- *Multipath ring* - a regular graph with all node degrees equal to 4. It is an improved version of the ring graph through the addition of internal connections between distant nodes. The number of nodes these connections jump over is referred to as *skip*.
- *Binary tree* - an acyclic graph in which each node starting from a root has two children, apart from the ones on the bottom layer (the leaves). The number of edges is one less than the number of nodes.
- *Mesh* - a rectangular grid of nodes.
- *Two-dimensional torus* - an improved version of the mesh where the nodes on the edges are connected with wrap-around links.
- *Complete graph* - a graph in which each pair of nodes is connected by an edge.

For each topology, a set of metrics was calculated (with approximations) as a function of total number of clusters N . However, modifications were made that bring the graph properties closer to the real-life implementation: it is assumed that one extra connection originates from every node that would correspond to a router connected to a cluster in the physical network (the leaves in the binary tree and every node in the other topologies). The analyzed metrics are:

- *Average node degree, number of links* and *number of routers* - the metrics are self explanatory and will serve as a measure of hardware complexity.

(a) Network of 9 clusters

(b) Network of 25 clusters

Figure 3.10: Comparison of topology properties

- *Diameter* - the longest shortest path in the network graph.
- *Clustering coefficient* - the average measure of how complete the neighborhood of every node is. It varies between 0 and 1 and together with the average path length indicate the small world characteristic of the graph. [38]
- *Average path length* - the average length of all shortest paths in the network. It is always less than (or equal in the case of a complete graph) with the diameter and together with the average path length indicate the small world characteristic of the graph.

It must be noted that the previously mentioned modification of adding the extra links is not used when calculating the clustering coefficient to ensure that the maximum possible value of 1 corresponds to the complete graph.

These metrics were chosen because they provide insight into hardware resource usage and complexity as well as efficiency in terms of data packet transmission. The metrics' values were either already well established in graph theory (such as diameter for ring and mesh), derived by hand (multipath ring) or obtained through a MATLAB script. The former is the case for average path length which does not have an easily derivable value. In order to compute it for the seven topologies, the following method was used: the adjacency matrix of all the networks was created for size N after which all minimum paths were determined with the MATLAB function *graphallshortestpaths.m* and then averaged.

Table 3.3 summarizes the characteristics for each topology, while Figure 3.10 is a graphical representation between the latency and hardware complexity of the topologies. This qualitative measure is based on the relations in (3.9) for a network size of 9 and 25 clusters (the reasoning behind these sizes is found in section 4.2.5).

$$\begin{aligned} \text{Complexity} &\propto f(\text{Average node degree, Number of links, Number of routers}) \\ \text{Latency} &\propto g(\text{Diameter, Clustering coefficient, Average path length}) \end{aligned} \quad (3.9)$$

The plots indicate that for a large network size the multipath ring and torus topologies are the attractive implementations, with the MPR of skip 1 being the best choice. As the network size decreases, differences between MPR and torus become more evident but the group maintains its advantages.

The complete graph was included because it represents the extreme in the spectrum in terms of smallest latency and greatest complexity even though it is clearly not implementable in hardware.

Following these results, the thesis will focus on the multipath ring and two dimensional torus topologies, which will be analyzed in the following chapter.

Layered network

The two layers of the proposed system architecture are described in this chapter with emphasis on the design criteria, routing protocols and traffic estimations.

4.1 Extraction, insertion and configuration layer

As previously mentioned, this layer is dedicated to transferring configuration and input (stimulus) data to each cluster (targeting each neuron) and extracting the desired parameters and transmitting them to the user. The particularities of such type of data traffic are exploited so as to create an efficient network topology.

4.1.1 Design criteria

The downstream traffic (towards the clusters) in this layer has the particular trait of consisting of a large number of packets sent to the same destination. In the majority of cases, routing decisions are made based on consulting tables after the destination address of the packet is read from its header. The presence of this header and the time it takes to access the routing tables represent a protocol overhead. Even if a more complex scheme of reading the address header once and then delivering the following packets up to a special ending one were used, routing tables would still be required. For this reason, a *channel* protocol is suggested as an efficient replacement that eliminates the use of routing tables and reduces the overhead.

The upstream traffic (from the clusters) presents no remarkable characteristic traits, but as mentioned in Section 3.4 it does not require a large bandwidth and is predictive. Upstream routing can be simplified to the point where no routing tables are used at all because there is only one possible destination - the user. In this case, routers are only tasked with handling arbitration in case of conflicts between data packets.

4.1.2 Proposed topology

The physical implementation for the Extraction, insertion and configuration layer is a binary tree, with each leaf being a cluster and the root an of-chip connection point (Figure 4.1). Each router has one upstream and two downstream ports and a one bit configuration signal that is used in the channel routing. The relatively small size of the overall network (the network size is expected to be less than 25 clusters based on the FPGA resources used in [12] on a Xilinx Virtex 7 and current mid to top range Xilinx Virtex UltraScale FPGA [39]) makes the binary tree sufficient as a network topology for this layer.

Figure 4.1: Binary tree structure of the Extraction, insertion and configuration layer

4.1.3 Routing scheme and router design

As the upstream traffic has no particularities, this section focuses on the custom routing scheme associated with the downstream one. As mentioned before, in order to reduce overhead, a channel scheme is proposed, the steps of which are as follows:

1. With the $S/\bar{T}$ (*Setup/Transmit*) signal set high, each router reads its corresponding bit (related to the layer on which the router is positioned) from the setup message, and redirects the package to either the left or right output port (Figure 4.2a).
2. On the falling edge of $S/\bar{T}$, the setup is latched into each router and a direct channel is now created between the root and the destination cluster (Figure 4.2b) that will remain in place until $S/\bar{T}$ is reasserted.
3. Packets sent from the root are automatically directed to the destination cluster.

A possible router structure, for downstream traffic, is shown in Figure 4.3 if we assume a parallel data bus is used. During synthesis, each router is tailored to read its specific setup bit depending on the layer at which it is positioned in the tree. The suggested routing scheme allows the router structure to be kept simple and small as no decision making takes place.

4.2 Topological layer

Selecting an appropriate physical network for the Topological layer represents the main challenge of this design. The traffic is patternless and continually changing as the network goes through the learning process and with power and delay as optimization targets, the chosen topology has to have high connectivity with a relatively simple hardware implementation.

(a) With $S/\bar{T}$ asserted, each router reads its corresponding bit from the setup packet and routes it appropriately

(b) With $S/\bar{T}$ set low, the previous setup is latched and a channel is maintained between the root and the targeted cluster

Figure 4.2: Steps in the routing scheme

Figure 4.3: Router structure for downstream traffic with a parallel data bus

Figure 4.4: Multipath ring topologies of size 13 with various skip values

Two network topologies are considered as possible solutions, the *Multipath ring* and *Two dimensional torus*, with their descriptions and analyses in the following subsections.

4.2.1 Multipath ring

4.2.1.1 Characteristics

The Multipath ring (MPR) topology is an example of a regular graph with all node degrees (router ports) equal to 5 (one port dedicated to an associated cluster and 4 to the network) so as to maintain physical implementation complexity low. As with other graphs from the same family, MPR has a high clustering coefficient but a relatively large path length for large networks [38]. However, assuming that data locality is exploited during neuron placement, the probability of a message needing to be sent to a cluster that is far away is small and the longer paths of the MPR will be rarely used. Also, as mentioned before, the expected number of clusters will not exceed 25 which makes the disadvantages of this topology limited.

The MPR links can be classified in two categories: *exterior* links, the ones connecting two nodes that are physically next to each other, and *interior* links, connecting nodes that are farther apart. These former links can have varying length and as a consequence can *skip* over a number of nodes and modify the average path length. Figure 4.4 shows an example of a multipath ring network of size 13 with skips of 1, 2 and 3. A small reduction in path length, from 2 to 1.83, is seen when increasing the skip but this comes at the cost of increasing traffic through the external links (Section 4.2.5).

4.2.1.2 Routing scheme

In order to estimate the traffic through the multipath ring topology, a routing scheme needs to be considered. For simplicity purposes, we assume static routing (the path between two clusters is known before runtime and is always the same), with the following set of rules:

1. If the message arrives from another router and the destination is this router's associated cluster then send it to the cluster controller.
2. If the message arrives from another router and the index distance to the destination cluster is less than or equal to the skip value, send the message through the exterior link. Otherwise, through the interior link.
3. If the message originates from the cluster associated to this router and the distance to the destination cluster index is less than or equal to half the number of clusters, send it counterclockwise. Otherwise, send it clockwise. In both cases, pick an exterior or interior link like in 2.

In essence, a router will prioritize sending a packet in the smallest number of hops and if the message originates from the associated cluster, it will balance the network as much as possible by taking advantage of MPR symmetry. The *index distance* is equal to the Manhattan distance between two routers when only exterior links are considered. Figure 4.5 shows the resulting routing path for every message originating from cluster zero towards the rest. The symmetric nature of this scheme makes the MPR most efficient for an odd number of clusters (the clockwise and counterclockwise branches are equal) but it can also accommodate even numbers by extending one of the branches to the extra cluster. Deadlocking is avoided by not allowing packets to go from an exterior link to an interior one, similar to turn restriction routing in the mesh topology.

4.2.2 Two dimensional torus

4.2.2.1 Characteristics

The two dimensional (2D) torus topology is formed by arranging all the routers in a rectangular lattice with each one connected to its immediate neighbors and with the connections at the edges wrapping around. Figure 4.6 shows a 2D torus with 12 clusters where each router is limited to 5 ports like in the MPR. The benefits of having such a topology come from the layout efficiency of the mesh and the existence of simple deadlock-free routing algorithms (X-Y, west first, north last etc.) combined with the small network diameter given by the wrap-around connections.

4.2.2.2 Routing scheme

As with the MPR, a static routing algorithm is considered for estimating the traffic, in this case, a variation on the XY routing scheme. When a message originates from a cluster, it is sent on one of the four network ports based on the quadrant in which

Figure 4.5: A routing scheme for the multipath ring topology. The red arrows indicate the path taken by a message originating from cluster zero to every possible destination cluster.

the destination is located, after which XY routing rules are followed. Figure 4.6 shows the resulting paths for messages originating from cluster five (chosen for illustration purposes due to its center position in the figure).

4.2.3 Traffic model

4.2.3.1 Assumptions

Due to the large computational effort required by the analysis of neural networks consisting of several thousands of biophysically-accurate neuron cells, a set of assumptions are made that simplify the procedure:

- A neuron cell contributes to the network traffic with an output rate, which depends on the model complexity and is known at the beginning of the simulation.
- All data packets are of same size.
- Apart from the previously mentioned routing rules, no protocol specifics are considered. A data packet is simply sent in the correct direction when it encounters a router.

Figure 4.6: A routing scheme for the two dimensional torus topology. The red arrows indicate the path taken by a message originating from cluster five on its path to every possible destination cluster.

- We assume there is no bandwidth limit in the network links, as this is one of the metrics that will result from the analysis.

Considering these assumptions, we can compute an average output rate λ of all neurons in the network which may be expressed in *packets/step* and use it as a measurement unit for the throughput estimates.

4.2.3.2 Metrics

The traffic estimation is based on three parameters:

- λ_i - number of distinct outgoing messages from cluster i .
- $\mu_{i,j}$ - number of messages from cluster i that are destined for cluster j .
- η - average percentage of outgoing messages common to all destination clusters.

Message refers to a data packet of arbitrary size that is sent from a cluster and contains the relevant neuron-to-neuron communication data. These parameters can be computed once all the neurons are placed in their respective clusters based on the communication scheme between them.

The output metrics of interest are:

- ω_i - Throughput value associated with the counterclockwise direction on exterior links for MPR and East direction for 2D Torus.

Figure 4.7: Cluster C0 metrics associated with the two analyzed topologies

- ω'_i - Throughput value associated with the clockwise direction on exterior links for MPR and West direction for 2D Torus.
- θ_i - Throughput value associated with the counterclockwise direction on interior links for MPR and North direction for 2D Torus.
- θ'_i - Throughput value associated with the clockwise direction on interior links for MPR and South direction for 2D Torus.

Figure 4.7 shows a graphical representation of these metrics for both the multipath ring and two dimensional torus. It must be noted that η is only used after the analysis is finished to correct the final results by reducing the effects of overestimating the required throughput of the links. Also, because ω and θ are defined as number of messages, they must be multiplied by the average output rate λ to get the actual throughput values.

4.2.3.3 Problem statement

Assuming a multipath ring network of size N and skip σ , for every cluster n the following equation stands:

$$\sum_{0 \leq i < N} \mu_{i,n} = \omega_{(n-1) \bmod N} + \omega'_{(n+1) \bmod N} + \theta_{(n-\sigma-1) \bmod N} + \theta'_{(n+\sigma+1) \bmod N} \quad (4.1)$$

where we consider $\mu_{n,n} = 0$. For example, in the case of the network shown in Figure 4.8 that has $N = 5$ and $\sigma = 1$, applying (4.1) results in the underdetermined system of equations (4.2). The aim of the analysis is to solve these types of systems of equations in terms of ω, ω', θ and θ' .

Figure 4.8: All the metrics in a size 5 and skip 2 MPR network

$$\begin{cases} \mu_{1,0} + \mu_{2,0} + \mu_{3,0} + \mu_{4,0} = \omega_4 + \omega'_1 + \theta_3 + \theta'_2 \\ \mu_{0,1} + \mu_{2,1} + \mu_{3,1} + \mu_{4,1} = \omega_0 + \omega'_2 + \theta_4 + \theta'_3 \\ \mu_{0,2} + \mu_{1,2} + \mu_{3,2} + \mu_{4,2} = \omega_1 + \omega'_3 + \theta_0 + \theta'_4 \\ \mu_{0,3} + \mu_{1,3} + \mu_{2,3} + \mu_{4,3} = \omega_2 + \omega'_4 + \theta_1 + \theta'_0 \\ \mu_{0,4} + \mu_{1,4} + \mu_{2,4} + \mu_{3,4} = \omega_3 + \omega'_0 + \theta_2 + \theta'_1 \end{cases} \quad (4.2)$$

A similar set of equations can be derived for the two dimensional torus with N_R rows and N_C columns for every cluster n :

$$\begin{aligned} \sum_{0 \leq i < N} \mu_{i,n} = & \omega_{\lfloor \frac{n}{N_C} \rfloor * N_C + (n \bmod N_C - 1) \bmod N_C} + \omega'_{\lfloor \frac{n}{N_C} \rfloor * N_C + (n \bmod N_C + 1) \bmod N_C} + \\ & + \theta_{(\lfloor \frac{n}{N_C} \rfloor + 1) \bmod N_R * N_C + n \bmod N_C} + \theta'_{(\lfloor \frac{n}{N_C} \rfloor - 1) \bmod N_R * N_C + n \bmod N_C} \end{aligned} \quad (4.3)$$

Applying (4.3) to every node will also result in an underdetermined system of equations.

4.2.3.4 Problem solver

A system of equations based on (4.1) and (4.3) is underdetermined, having $4N$ unknown variables and only N equations for a network of size N which makes solving it not a

straightforward task. Various techniques can be used in this situation such as QR decomposition [40], singular value decomposition [41], the pseudo-inverse or least squares [42]. However, in this particular case a simpler solution is to use superposition to solve the system. Because the routing algorithm is static, the path a message originating from any cluster will follow is already known and is independent of other clusters. As a result, the effects of traffic injected by one cluster can be accounted for while ignoring all the rest, which describes the superposition theorem. The first step for solving (4.2), is given in (4.4) where only the effects of cluster C0 are considered. After repeating this step for each cluster, the results can be combined and solution (4.5) extracted.

$$\begin{cases} \omega_0 = \mu_{0,1} & \omega'_0 = \mu_{0,4} & \theta_0 = \mu_{0,2} & \theta'_0 = \mu_{0,3} \\ \omega_1 = 0 & \omega'_1 = 0 & \theta_1 = 0 & \theta'_1 = 0 \\ \omega_2 = 0 & \omega'_2 = 0 & \theta_2 = 0 & \theta'_2 = 0 \\ \omega_3 = 0 & \omega'_3 = 0 & \theta_3 = 0 & \theta'_3 = 0 \\ \omega_4 = 0 & \omega'_4 = 0 & \theta_4 = 0 & \theta'_4 = 0 \end{cases} \quad (4.4)$$

$$\begin{cases} \omega_0 = \mu_{0,1} & \omega'_0 = \mu_{0,4} & \theta_0 = \mu_{0,2} & \theta'_0 = \mu_{0,3} \\ \omega_1 = \mu_{1,2} & \omega'_1 = \mu_{1,0} & \theta_1 = \mu_{1,3} & \theta'_1 = \mu_{1,4} \\ \omega_2 = \mu_{2,3} & \omega'_2 = \mu_{2,1} & \theta_2 = \mu_{2,4} & \theta'_2 = \mu_{2,0} \\ \omega_3 = \mu_{3,4} & \omega'_3 = \mu_{3,2} & \theta_3 = \mu_{3,0} & \theta'_3 = \mu_{3,1} \\ \omega_4 = \mu_{4,0} & \omega'_4 = \mu_{4,3} & \theta_4 = \mu_{4,1} & \theta'_4 = \mu_{4,2} \end{cases} \quad (4.5)$$

In the case of such a small network, the resulting throughput values consist of only one μ parameter and as such they are exact and no scaling with η is required. However, for bigger networks like the one showed in Figure 4.5 the solution is more complex, as in (4.6). Note that all μ parameters originating from the same cluster have been scaled with $1 - \eta$. The reason for this is as follows: assume cluster C0 sends a total of 5 data packets, $d1$, $d2$, $d3$, $d4$ and $d5$ into the network, destined for clusters C1 and C2. Out of the five, $d1$, $d2$, $d3$ and $d4$ are destined for C1 and $d2$, $d4$ and $d5$ for C2, which means both $\mu_{0,1}$ and $\mu_{0,2}$ count $d2$ and $d4$ resulting in an overestimation of the throughput if all these messages traveled through a common link. To mitigate this overestimation, the values are scaled with $1 - \eta = \frac{3}{5}$, the percentage of packets that are not in common, to result in a closer estimation of the actual throughput.

Note that the routing scheme intervenes during each superposition step when computing the parameters. It is now clear that a static algorithm must be used otherwise it is impossible to apply superposition (adaptive routing requires the effect of all clusters to be considered, which contradicts the superposition assumptions).

$$\begin{cases} \omega_0 = (1 - \eta) * (\mu_{0,1} + \mu_{0,2} + \mu_{10,1} + \mu_{10,2}) + \mu_{12,1} + \mu_{9,1} \\ \omega'_0 = (1 - \eta) * (\mu_{0,12} + \mu_{0,11} + \mu_{3,12} + \mu_{3,11}) + \mu_{1,12} + \mu_{4,12} \\ \theta_0 = (1 - \eta) * (\mu_{0,3} + \mu_{0,4} + \mu_{0,5} + \mu_{0,6}) + \mu_{10,3} \\ \theta'_0 = (1 - \eta) * (\mu_{0,10} + \mu_{0,9} + \mu_{0,8} + \mu_{0,7}) + \mu_{3,10} \end{cases} \quad (4.6)$$

Table 4.1: The placement of neurons inside clusters

Cluster ID	Neuron IDs
0	0, 1, 2, 3
1	4, 5, 6, 7
2	8, 9, 10, 11
3	12, 13, 14, 15
4	16, 17, 18, 19

4.2.4 MATLAB implementation

The entire analysis was done with MATLAB functions and scripts written for this thesis. The required steps for finding the throughput estimates are described in this section.

The first step in the analysis is to define the type of communication scheme and the number of neurons in each layer. For exemplification purposes, we assume feed forward with 8 input, 10 hidden and 2 output neurons (Figure 4.9a). This scheme can be represented in the form of a *linkList*, a size Lx2 array that contains the link source and destination nodes in each column, which then serves as input for the *createClusterSet.m* function along with the desired cluster size. Assuming this size to be 4, the result will be the data set shown in Table 4.1 and visually represented in Figure 4.9b.

As a side note, in this case the neurons were allocated into clusters based on their index, only for demonstration purposes as this is not an optimal solution - cluster 0 and 1 send messages originating from four distinct sources to three other clusters i.e., none of their neurons have internal communications. The effects of placement rules are investigated in section 4.2.5.3 where neurons are grouped based on their interconnects, ignoring their indexes and position in layers.

Once the neurons are placed in clusters, the analysis parameters can be generated by calling *findLambdaMu.m*. Assuming a single topological layer for all neuron-to-neuron communications, the function outputs for the given example are:

$$\lambda = \begin{pmatrix} 4 \\ 4 \\ 4 \\ 4 \\ 0 \end{pmatrix} \quad \mu = \begin{pmatrix} 0 & 0 & 4 & 4 & 4 \\ 0 & 0 & 4 & 4 & 4 \\ 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad \eta = 0.66$$

The final step is to assign a network topology and use these results as cluster parameters to find the estimated throughputs. Two functions were developed, *solveMultipathRing.m* and *solve2DToroid.m* for estimating traffic through the MPR and torus network respectively. In addition to λ , μ and η these functions take as input the rule structure corresponding to the static routing scheme for each topology. The structure contains 4 matrices, one for each throughput variable associated with cluster zero, formed with the coefficients from the first superposition step (for example (4.4)). The outputs from these two functions are four arrays with N elements containing the ω , ω' , θ and θ' values.

Figure 4.9: The feed forward scheme of size 8-10-2

4.2.5 Traffic estimation

A test neural network for each of the four communication schemes was created, with 20 input, 180 hidden and 10 output neurons, totaling 210 cells. This number was chosen for its large number of divisors that allows the creation of various MPR and 2D torus networks with completely full clusters. Because the method uses cluster level metrics (μ) for the estimation, a larger number of neurons would only result in changes in throughput magnitudes, not traffic behavior, so in order to speed up the computation, this small number of neurons was chosen.

Traffic was estimated for the 2D Torus and the skip 1, 2 and 3 MPR topologies. The number of clusters was varied between 9, the minimum number for which a skip of 3 is possible, and 25, a number based on the available resources in contemporary FPGA (e.g. a medium to top of the range Xilinx Virtex UltraScale [39] has roughly four times as many logic cells than the device used in [12]). When creating the torus network, the number of columns and rows was chosen so as to keep the number of routers without an associated cluster as small as possible. Table 4.2 contains the corresponding network layout for various network sizes.

Table 4.2: Two dimensional torus network layout for various network sizes

N	Layout (row x col)
4	2 x 2
5 - 6	2 x 3
7 - 9	3 x 3
10 - 12	3 x 4
13 - 16	4 x 4
17 - 20	4 x 5
21 - 25	5 x 5

4.2.5.1 Random neuron placement

The 210 neuron cells were randomly distributed among the N clusters for all analyzed topologies and communication schemes 1000 times, after which the throughput estimations were averaged and compared.

Figure 4.10 shows the average maximum throughput in the MPR and Torus. Traffic estimates indicate that the MPR experiences smaller maximum throughputs than the torus for networks of medium sizes. For larger networks, the torus achieves marginally better performance, however, tendencies in the plots suggest that a MPR with larger skip might be a better solution, but this has to be confirmed through further simulations.

Changes in the skip have a significant effect on the maximum throughputs but the relationship is not straightforward. The observed behavior suggests that the skip value has to be chosen depending on network size in order to ensure minimum strain on the network. For example, if the full lateral inhibition communication scheme is used, for networks with less than 16 clusters, a skip of 2 is best, but for larger sizes 3 behaves better.

Of interest is also the change in traffic distribution across the interior and exterior links in the MPR for different skip values. Figure 4.11 and Figure 4.12 contain the maximum counterclockwise throughput values in the two types of links for all four communication schemes. Due to the symmetric nature of the multipath ring, the clockwise traffic is identical. The results indicate that traffic is heavily influenced by changes in the skip. We can observe that as this value increases, data packets are being redirected through the exterior links and that an optimal value can be determined in order to achieve a balance in the network. For example, in this case a skip equal to 2 is recommended for the analyzed size range.

Figure 4.13 shows the average difference between maximum and minimum throughput values in the MPR and 2D torus. If an appropriate skip value is chosen for the multipath ring, a very balanced network can be constructed with far better results than the two-dimensional torus for all tested communication schemes. As with the maximum throughput values, for small network sizes the optimal skip value is 2.

(a) Feed forward

(b) Complete hidden layer

(c) Layered full lateral inhibition

(d) Layered neighbor lateral inhibition

Figure 4.10: Average maximum throughput values

4.2.5.2 Split topological layer

As resulted from the preliminary analysis in section 3.3, the histograms of node output degrees present a pattern that suggests separating the traffic in the topological layer according to the number of neuron connections. Apart from CHL, all other communication schemes have two distinct peaks in their histograms, so a split in the topological layer will be analyzed. The two resulting layers will be referred to as the *Small Degree Topological Layer (SDTL)* and *Large Degree Topological Layer (LDTL)* (Figure 4.14). As the names suggest, SDTL will handle all data packets originating from neurons with an output degree less than a specific threshold, while LDTL the rest.

According to section 3.3, the histogram peaks are located at $c_- = N_I$ and $c_- = N_H$ for FF and $c_- = N_I$ and $c_- = \frac{N_H}{2}$ for LNLI and LFLI. As in the simulations the number of hidden neurons is set to 180 and the number of input neurons is 20, a threshold of anywhere between 21 and 89 would split the traffic between the two layers.

The maximum estimated throughputs are shown in Figure 4.15. The best improvements are seen for feed forward where regardless of network topology, large networks can reach up to 50% reduction in maximum throughput values. This is because there are a lot of neurons with small output degrees and very few with large ones so neither layers are heavily loaded (Figure 3.8a). LFLI which has relatively equal histogram

(a) Feed forward

(b) Complete hidden layer

(c) Layered full lateral inhibition

(d) Layered neighbor lateral inhibition

Figure 4.11: Average throughput for exterior links in the MPR network

peaks (Figure 3.8c) manifests around 33% in maximum traffic reduction.

There is little improvement in the case of a LNLi communication scheme, but this is due to a poor choice in threshold values. The large peaks are located around $c_- \approx 1$ and $c_- = [10, 20]$ (Figure 3.8d) which suggests a threshold value between 2 and 9. A different threshold can be selected for the complete hidden layer scheme within the range of 181 to 189. With these new threshold values, the results are shown in Figure 4.16. Improvements are seen in the case of LNLi over the previous case but the overall traffic reduction is not as significant as for LFLi because the two neuron groups have similar sizes and relatively small degrees.

Because of the presence of a large group of neurons with large output degrees in the complete hidden layer communication scheme (Figure 3.8b) there is very little reduction in traffic even for such a high threshold level.

The multipath ring with skip 1 appears to have the largest reductions in maximum throughput, with skip 3 benefiting the least. However, even in the best case (feed forward scheme on MPR with skip 1) the throughput reduction is sufficiently large only for big networks which makes the idea of splitting the topological layer in multiple parts ineffective as the increase in hardware resources consumption is too large to overcome.

(a) Feed forward

(b) Complete hidden layer

(c) Layered full lateral inhibition

(d) Layered neighbor lateral inhibition

Figure 4.12: Average throughput for interior links in the MPR network

4.2.5.3 Specific neuron placement

A simple neuron placement algorithm was tested as a possible method to reduce the throughput values in the MPR network. The idea behind the algorithm is to group neurons with a large number of output connections together with their neighbors in the same cluster so as to minimize cluster-to-cluster communication. When placing neurons, the following steps are applied:

1. Select the yet unplaced neuron with most outgoing connections.
2. Select a cluster with a vacancy of more than the neuron output degree if available, otherwise select the cluster with most vacancy and place the neuron in it.
3. Select the yet unplaced neighbor with the largest output degree and if possible place it in the cluster, otherwise jump to 2.
4. If there are no more neighbors, skip this step, otherwise jump to 3.
5. If there are neurons that are still unplaced, jump to 1., otherwise the placement is complete.

Figure 4.13: Average difference between maximum and minimum throughput in the MPR and 2D torus network

The effects this placing algorithm has on the average traffic through the network is shown in Figure 4.17. Significant improvements are seen for the feed forward scheme both for MPR and 2D torus. Complete hidden layer manifests little to no changes because of the large number of highly connected hidden neurons that makes rearranging them have almost no effect on cluster-to-cluster communication.

For the other two cases, placing neurons following the suggested rules has negative effects, with average throughput values drastically increasing. This is because the inhibitory neurons communicate with a significant number of neurons so spreading highly connected neurons in different clusters means that these inhibitory cells will send packets to more clusters. A different set of rules, such as placing neurons in clusters according to their type (input, inhibitory etc.) might be better suited for these topologies.

Figure 4.14: The system structure with a split topological layer

4.3 Conclusion

The resulting traffic estimations underline the advantages of the multipath ring network as opposed to the two-dimensional torus. The maximum throughput values are smaller than in the case of the torus and depending on network size, the skip value can be selected to provide an optimal topology. Results indicate that as network size increases, the skipping distance has to be increased both to reduce throughput but also to balance the network. Traffic shaping was also detected, with the skip value having an effect on traffic distribution between the external and internal links.

The same behavior is seen for all four communication schemes, suggesting that the multipath ring topology can support the changes that take place during runtime due to learning.

Overall, the multipath ring network is better suited for the Topological layer, because of its traffic characteristics and its symmetric and simple physical implementation.

The possibility of splitting the topological layer into two network levels that handle traffic originating from neurons with large and small number of output connections, respectively, has also been analyzed. Based on output degree histograms, different threshold values were set for the four communication schemes that resulted in various reductions in maximum throughput values in the two layers. However, the noticed improvements were not enough to overcome the extra hardware resource consumption and therefore the system architecture is better being kept at only one topological layer.

It is worth noting that all the analyses done so far do not take into account conges-

(a) Feed forward

(b) Complete hidden layer

(c) Layered full lateral inhibition

(d) Layered neighbor lateral inhibition

Figure 4.15: The average maximum throughput values in the MPR and 2D torus network in the case of a split topological layer

(a) Layered neighbor lateral inhibition

(b) Complete hidden layer

Figure 4.16: The average maximum throughput values in the MPR and 2D torus network in the case of a split topological layer with different threshold values for LNLI and CHL

tion in the network because of the infinite bandwidth assumption from section 4.2.3.1. This allows superposition to be used for solving the system of equations that describes

(a) Feed forward

(b) Complete hidden layer

(c) Layered full lateral inhibition

(d) Layered neighbor lateral inhibition

Figure 4.17: The effect of the placing algorithm on average throughput in the MPR and 2D torus network

traffic patterns in the network and as a consequence, at the moment, the method is unsuited for any congestion aware analysis.

Energy-delay product

The high performance, power consumption and reliability requirements in current technology have changed the methodology when it comes to *network on chip (NoC)* design. More focus is allocated to estimating the final power consumption even in earlier development stages through the use of system level modeling. Two main components are considered when making these estimations: *latency* or the delay one packet encounters and the *energy* the said packet consumes when traversing the network.

The energy can be divided in two components: dynamic energy which captures the effect of contention in the network and leakage energy which is the buffer static energy consumption. [43] For current process technologies, the dynamic energy component dominates and is generically expressed as

$$E_{dynamic} = 0.5\alpha CV_{DD}^2 \quad (5.1)$$

where C is the total switching capacitance, V_{DD} the supply voltage and α the switching activity. If this value is multiplied by the system clock frequency f_{CLK} one can determine the dynamic power consumption.

Various methods have been developed to provide estimations of the power/energy consumption of a system. In [44] the authors describe a power and performance interconnect simulator called Orion that uses power models based on detailed estimations of gate and wire capacitance and switching activities. These parametrized power models were created for complex network components such as arbiters, FIFO buffers and SRAM blocks starting from other existing simple component models. However, this method requires the use of low level architectural descriptions to achieve high precision in its estimations.

Another estimation method is presented in [45] where VHDL descriptions of the system are used to simulate the network behavior and power consumption patterns. The authors assume one specific type of routing (wormhole) and injection rate distribution (self-similar) for their analysis and focus on a handful of network topologies among which are the mesh, torus and fat tree.

In both of these methods, as in most cases, estimating power and energy consumption of a system requires low level modeling of the architecture followed by lengthy simulations that then provide the results. A much faster approach, albeit not as exact, would be to not rely on simulations but on traffic patterns through the network. Such a case is presented in [46] where the *communication probability distribution (CPD)* is used, which represents the probability that a packet will travel a certain distance through the NoC. Once traffic patterns are generated based on specific rules for the network topology, the CPD is calculated and then used as input for a simple mathematical model that gives the energy estimate.

This simulation-free method of finding energy estimates is well suited for comparing

the multipath ring with other commonly used topologies because so far there are no hardware implementations for the multipath ring and developing them for the sole purpose of estimating power consumption is not feasible or useful (implementations could easily change throughout the design process). Also, the method is fast and can make use of the traffic estimations determined in the previous chapter.

The energy-delay-product [47] will be used as an objective comparison criterion between the multipath ring, mesh, binary tree and their improved variants, inverse clustering and inverse clustering with mesh that is designed for low power applications.

5.1 Mathematical derivation

The EDP estimate for a data packet sent from cluster i to cluster j is defined as:

$$EDP_{i-j} = E_{i-j} * D_{i-j} \quad (5.2)$$

where E_{i-j} is the average energy required to send the packet and D_{i-j} is the delay of said packet.[48] These two variables are calculated as shown in (5.3) where E_{router} and E_{link} are the average energies consumed in the router and link segment as the packet passes through and N_{router} and N_{link} are the number of routers and links in the entire path. The delay is influenced by three elements: D_{router} , the average delay through a router from one network port to another, D_{clust} which is the delay from a cluster to its dedicated router and D_{link} , the delay through a link segment. Figure 5.1 shows a schematic representation of all the energy and delay components in a path from cluster i to j .

$$\begin{aligned} E_{i-j} &= N_{router} * E_{router} + \sum E_{link} \\ D_{i-j} &= (N_{link} - 2) * D_{link} + 2 * D_{clust} + N_{router} * D_{router} \end{aligned} \quad (5.3)$$

After implementing a design for the network components, the delay parameters are easily extracted from synthesis reports, while the energy consumed is estimated using Xilinx XPE [49]. For this analysis, however, all parameters are set equal to 1, but in order to distinguish among topologies, E_{link} is considered dependent on the link segment's position (and length) and has different values for each analyzed topologies as shown in the following sections.

There is one important effect that must be underlined, which is the assumption that all E_{router} and D_{router} parameters are equal. This translates to the ideal case where there is no congestion and the router can immediately transfer the incoming packet to the corresponding output port, in other words, traffic through the network is neglected. For the purposes of the desired comparison between network topologies this is not detrimental as it idealizes the physical limitations and accounts for the topological advantages alone that each architecture presents. However, one can still use an average measured/estimated value for these parameters to make the EDP modeling more accurate.

Equation (5.2) is based on the original definition of the energy-delay product given in [47], but other works such as [50] have assigned more weight to the delay parameter by squaring or even cubing it. This method yields more accurate power estimations for

Figure 5.1: The energy and delay components that make up the path from cluster i to j

performance oriented systems where a reduction in supply voltage produces a reduction with approximately the cube root of the power reduction in operating frequency (and as a consequence increased latency)[51]. For thoroughness, (5.4) will also be used in the energy-delay product estimation.

$$EDP_{i-j} = E_{i-j} * D_{i-j}^3 \quad (5.4)$$

In all derivations, the index difference between cluster i and j defined as (5.5) will be used.

$$\Delta = |i - j| \quad (5.5)$$

5.1.1 Multipath ring

For the MPR, the interior links are considered to be longer than the exterior ones and as a consequence, the values for E_{link} are different and given by (5.6), where σ is the skip value.

$$E_{link} = \begin{cases} 1 & , if \text{ external link} \\ \sigma + 1 & , if \text{ internal link} \end{cases} \quad (5.6)$$

The equations from (5.7) have been determined and are used to compute N_{router} , N_{link} and $\sum E_{link}$ for every path between i and j .

$$\begin{aligned} N_{router} &= \begin{cases} \lfloor \frac{\Delta}{\sigma+1} \rfloor + \Delta \text{mod}(\sigma+1) + 1 & , \Delta \leq \frac{N}{2} \\ \lfloor \frac{N-\Delta}{\sigma+1} \rfloor + (N-\Delta) \text{mod}(\sigma+1) + 1 & , \Delta > \frac{N}{2} \end{cases} \\ N_{link} &= N_{router} + 1 \\ \sum E_{link} &= \begin{cases} \lfloor \frac{\Delta}{\sigma+1} \rfloor * (\sigma+1) + \Delta \text{mod}(\sigma+1) + 1 & , \Delta \leq \frac{N}{2} \\ \lfloor \frac{N-\Delta}{\sigma+1} \rfloor * (\sigma+1) + (N-\Delta) \text{mod}(\sigma+1) + 1 & , \Delta > \frac{N}{2} \end{cases} \end{aligned} \quad (5.7)$$

Figure 5.2: Comparison between binary tree and inverse clustering networks of size 8. The numbers on the links indicate proportionality factors between each corresponding E_{link}

5.1.2 Mesh

In this case, a square network is assumed, with the horizontal links physically shorter than the vertical ones by a factor of 4. For this reason, E_{link} is defined as:

$$E_{link} = \begin{cases} 1 & , if \text{ horizontal link} \\ 4 & , if \text{ vertical link} \end{cases} \quad (5.8)$$

which gives:

$$\begin{aligned} N_{router} &= \lfloor \frac{\Delta}{\sqrt{N}} \rfloor + \Delta \bmod \sqrt{N} + 1 \\ N_{link} &= N_{router} + 1 \\ \sum E_{link} &= \lfloor \frac{\Delta}{\sqrt{N}} \rfloor * 4 + \Delta \bmod \sqrt{N} + 1 \end{aligned} \quad (5.9)$$

5.1.3 Tree and Inverse clustering

A small binary tree and inverse clustering network are shown in Figure 5.2. The inverse clustering network is described in [52] and is essentially a rearrangement of the binary tree so that clusters that are farther away from each other are connected with shorter paths. This has the effect of reducing the delay and energy through these paths because long distance traveling packets no longer need to pass through a large number of routers and in the case of CMOS IC implementations, can be routed through lower metal layers.

The assumed static routing scheme for both networks always chooses the path with smallest number of hops for sending a data packet from cluster i to j .

Figure 5.2 also shows the proportionality factors for E_{link} . For the binary tree, the lower the level, the shorter the links are assumed to be, a trait which is reversed for the inverse clustering. Under these assumptions, (5.10) are the equations used for calculating all necessary parameters for the binary tree.

Figure 5.3: The inverse clustering with mesh network topology and the associated E_{link} proportionality factors. The inset shows the same network arranged as a mesh with the connections in the first inverse cluster layer visible.

$$\begin{aligned}
 N_{router} &= 2 * \lfloor \log_2 \Delta \rfloor + 1 \\
 N_{link} &= N_{router} + 1 \\
 \sum E_{link} &= 2^{\lfloor \log_2 \Delta \rfloor + 2} - 2
 \end{aligned} \tag{5.10}$$

Because of the connection rearrangements in the inverse clustering topology, finding an exact solution for the three parameters is difficult. A work around was used where the cluster indexes were temporarily changed to a binary tree but with the E_{link} proportionality factors from the normal inverse clustering, the parameters computed in a similar fashion as with (5.10) and then the indexes changed back to their original values.

5.1.4 Inverse clustering with mesh

Inverse clustering with mesh is a hybrid architecture proposed in [52] that, as the name suggests, uses a mesh to connect neighboring clusters and the inverse clustering for longer paths. Figure 5.3 shows an example of such a topology for a network of 16 clusters. The corresponding E_{link} proportionality factors are also indicated on the links. Finding an exact equation for computing the three parameters of interest is difficult because of the hybrid nature of the architecture and so, for the simulations, all parameters were manually calculated by analyzing every path between cluster i and j and choosing the shortest one. These values are presented in Appendix A.

5.2 EDP estimation

Figure 5.4a shows the energy-delay product estimations for a network of size 32 arranged as a function of Manhattan distance according to (5.2). For small distances, the multipath ring has the smallest estimated EDP for all skip values. Tree and inverse clustering with mesh have an almost identical EDP, with the latter behaving marginally worse. As the distance increases, MPR with skip 1 becomes the best choice out of the three MPR topologies, with mesh becoming slightly more advantageous for large distances. Nevertheless, MPR with skip 2 and 3, which have almost the same EDP, have a smaller maximum distance (network diameter) than all the other topologies and remain the attractive solution.

Figure 5.4b presents the EDP estimates using (5.4). The same characteristics are visible, but now the mesh topology has a greater advantage over the multipath ring with skip 1 for large distances. Nevertheless, the MPR topologies remain attractive for small to medium distances due to their very low EDP and small diameters.

Because data locality is assumed to be exploited in the configuration of the system, the advantages that the mesh topology exhibits for large distances have very low probabilities to manifest, which leaves the multipath ring as the topology of choice.

The effect traffic has on the global EDP (the value corresponding to the entire network, not just one message) can also be studied. In section 4.2.5 throughputs were estimated for the multipath ring after 1000 random neuron placements, estimations that can be used in the traffic dependent global EDP calculations. Figure 5.5 shows the estimated EDP according to (5.2) as a function of Δ , the index difference, for a multipath ring topology of size 32. The symmetric nature of the architecture can be seen in the shape of the EDP. The peak represents the worst case when $\Delta = (N - 1)/2$ i.e., when the source and destination cluster are on opposite sides of the network.

According to the definition, packets traveling through the exterior links of the MPR are counted by ω and ω' and all of them have the EPR value corresponding to $\Delta = 1$. The interior traffic expressed by θ and θ' depends on the skip value σ and has an EDP corresponding to $\Delta = \sigma + 1$. Following these rules, the global EDP can be computed for every one of the 1000 iterations with:

$$EDP = \sum_{i=0}^{N-1} (\omega_i + \omega'_i) * EDP_1 + \sum_{i=0}^{N-1} (\theta_i + \theta'_i) * EDP_{\sigma+1} \quad (5.11)$$

where N is the number of clusters. As the smallest and largest networks for which traffic was estimated had a size of 9 and 25 respectively, the following results will be for these two cases. There is no error introduced in the estimation by using a EDP developed for $N = 32$ in place of 9 or 25 because the only values that intervene in (5.11) are for Δ less than 4 (the maximum skip is 3) which are the same for all cases. The average of the 1000 global EDP estimates are shown in Figure 5.6, with the error bars representing one standard deviation.

Results indicate that the communication scheme, in essence traffic distribution, has a powerful effect on the energy-delay product regardless of network size, but more prominent for a large number of small sized clusters because the number of intra cluster

(a) Normal delay weight

(b) Cubed delay weight

Figure 5.4: EDP estimations for different network topologies of 32 clusters vs. Manhattan distance for different delay weights

Figure 5.5: Energy-delay product as a function of index difference in the multipath ring of size 32

connections is small and there is more strain on the network. In this case, EDP can vary with up to 50% for MPR with skip 3 between LNLi and CHL.

The estimates for large cluster sizes indicate that there exists an optimum skip value for minimizing EDP, which confirms the results from Section 4.2.5 which would recommend a skip of 2 for networks of 9 clusters. However, this is not also true for CHL and not enough measurements are available to confirm whether or not this is due to insufficient placement iterations or if it is a characteristic of the tightly connected scheme. In the case of a 25 cluster network, the small number of neurons allocated to each cluster results in no such pattern being visible.

(a) A network of size 25

(b) A network of size 9

Figure 5.6: Global energy-delay products for all communication schemes with traffic resulting from random neuron placements

Conclusion and future work

6.1 Conclusion

After multiple analyses focused on traffic and energy consumption, the multipath ring (MPR) topology was validated as an efficient dataflow architecture for neuron-to-neuron communication in a clustered spiking neuron network simulator to be implemented on FPGA. When compared with the two-dimensional torus, an architecture commonly employed for multi core platforms, better throughput capacity was found along with the possibility of traffic shaping, or redirecting through specific links.

By configuring MPR parameters such as skipping distance, the topology can be adapted for specific network sizes and to maintain very good traffic balance. The MPR was proven capable of handling different neuron communication schemes as well which makes it suitable for supporting the continuous changes in traffic patterns due to learning processes in the neural network.

As a method for characterizing the energy efficiency of the multipath ring network topology, the energy-delay product was estimated and compared with other network types such as the low power inverse clustering with mesh. The observed values indicate an energy efficient topology, comparable with its peers, that has the added benefit of small network diameter and physical implementation simplicity.

Strain on the network was reduced using two physically separate communication layers, one for the neuron-to-neuron data, for which the MPR was suggested, and one for configuration and input/output data. For the latter, a binary tree topology was adapted, with a channel routing protocol for downstream traffic. In the case of upstream traffic, no routing decisions have to be made apart from arbitration. Overall, the routers can be kept simple and do not require routing tables.

A possible further reduction in network throughput by separating the neuron-to-neuron communication layer into two distinct networks that handle neurons with a large degree of connectivity separately from those with small degrees was investigated. The results indicate a reduction in maximum throughput values that is heavily dependent on neuron-to-neuron communication schemes and network size. Up to 50% smaller throughputs can be achieved for feed forward schemes distributed among large number of clusters but for other topologies the improvements are minor or nonexistent. The conclusion is that the benefits of splitting the neuron-to-neuron traffic are not enough to justify the extra FPGA resources that are consumed.

6.2 Future work suggestions

The next step in the analysis of the proposed heterogeneous neuronal array system topology is to implement the design in SystemC and achieve cycle accurate simulations. The throughput values can then be determined for different types of traffic patterns and other routing schemes, including dynamic variants. Afterwards, a transition to synthesizable SystemC [53] and implementation on FPGA can be made.

An intelligent neuron placement algorithm can be devised to allow optimum configurations to be used that reduce overall network load either during runtime (actively reassigning neurons to clusters) or at least at the starting state of the system. This thesis has shown that such an algorithm has to be sensitive to the communication scheme between the neurons as the effects can vary greatly.

Traffic estimations revealed that by changing the skip value in the multipath ring, traffic could be redistributed between the exterior and interior links. An interesting path would be to investigate if this characteristic can be exploited as a method for adapting to changes in traffic patterns by reconfiguring the network topology during runtime with a minimum number of router and link changes.

Bibliography

- [1] W. Gerstner and W. M. Kistler, *Spiking Neuron Models*. Cambridge University press, 2002.
- [2] W. Teka, T. Marinov, and F. Santamaria, “Neuronal spike timing adaptation described with a fractional leaky integrate-and-fire model,” *PLoS Computational Biology*, vol. 10, p. e1003526, Mar 2014.
- [3] E. M. Izhikevich, “Simple model of spiking neurons,” *IEEE Transactions on Neural Networks*, vol. 14, pp. 1569–1572, Nov 2003.
- [4] A. L. Hodgkin and A. F. Huxley, “A quantitative description of membrane current and its application to conduction and excitation in nerve,” *Journal of Physiology*, vol. 117, no. 4, pp. 500–544, 1952.
- [5] R. Wang, T. J. Hamilton, J. Tapson, and A. van Schaik, “An fpga design framework for large-scale spiking neural networks,” in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 457–460, June 2014.
- [6] U. Farooq, Z. Marrakchi, and H. Mehrez, *Tree-based Heterogeneous FPGA Architectures*. Springer, 2012.
- [7] M. Ambroise, T. Levi, Y. Bornat, and S. Saighi, “Biorealistic spiking neural network on fpga,” in *2013 47th Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–6, March 2013.
- [8] A. Cassidy, A. G. Andreou, and J. Georgiou, “Design of a one million neuron single fpga neuromorphic system for real-time multimodal scene analysis,” in *2011 45th Annual Conference on Information Sciences and Systems*, pp. 1–6, March 2011.
- [9] S. W. Moore, P. J. Fox, S. J. T. Marsh, A. T. Markettos, and A. Mujumdar, “Bluehive - a field-programable custom computing machine for extreme-scale real-time neural network simulation,” in *2012 IEEE 20th International Symposium on Field-Programmable Custom Computing Machines*, pp. 133–140, April 2012.
- [10] R. Miikkulainen, “Neuroevolution,” in *Encyclopedia of Machine Learning* (C. Sammut and G. I. Webb, eds.), pp. 716–720, Boston, MA: Springer US, 2010.
- [11] G. J. Christiaanse, A. Zjajo, C. Galuzzi, and R. van Leuken, “A real-time hybrid neuron network for highly parallel cognitive systems,” in *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 792–795, Aug 2016.
- [12] J. Hofmann, A. Zjajo, C. Galuzzi, and R. van Leuken, “Multi-chip dataflow architecture for massive scale biophysically accurate neuron simulation,” in *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 5829–5832, Aug 2016.

- [13] G. J. Christiaan, “Real-time hybrid neuron network for highly parallel cognitive systems.” MSc Thesis Delft University of Technology, The Netherlands, 2016.
- [14] J. D. Gruijl, P. Bazzigaluppi, M. T. G. de Jeu, and C. I. de Zeeuw, “Climbing fiber burst size and olivary sub-threshold oscillations in a network setting,” *PLoS Computing Biology*, vol. 12, p. e1002814, December 2012.
- [15] E. M. Izhikevich, “Which model to use for cortical spiking neurons?,” *IEEE Transactions on Neural Networks*, vol. 15, pp. 1063–1070, Sept 2004.
- [16] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, “The spinnaker project,” *Proceedings of the IEEE*, vol. 102, pp. 652–665, May 2014.
- [17] 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*, vol. 62, pp. 2454–2467, Dec 2013.
- [18] B. V. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. R. Chandrasekaran, J. M. Bussat, R. Alvarez-Icaza, J. V. Arthur, P. A. Merolla, and K. Boahen, “Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations,” *Proceedings of the IEEE*, vol. 102, pp. 699–716, May 2014.
- [19] P. Merolla, J. Arthur, R. Alvarez, J. M. Bussat, and K. Boahen, “A multicast tree router for multichip neuromorphic systems,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, pp. 820–833, March 2014.
- [20] 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, 2014.
- [21] M. Beuler, A. Tchaptchet, W. Bonath, S. Postnova, and H. A. Braun, “Real-time simulations of synchronization in a conductance-based neuronal network with a digital fpga hardware-core,” in *Artificial Neural Networks and Machine Learning – ICANN 2012: 22nd International Conference on Artificial Neural Networks, Lausanne, Switzerland, September 11-14, 2012, Proceedings, Part I* (A. E. P. Villa, W. Duch, P. Érdi, F. Masulli, and G. Palm, eds.), pp. 97–104, Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.
- [22] H. A. Braun, M. T. Huber, M. Dewald, K. Schfer, and K. Voigt, “Computer simulations of neuronal signal transduction: The role of nonlinear dynamics and noise,” *International Journal of Bifurcation and Chaos*, vol. 08, no. 05, pp. 881–889, 1998.
- [23] M. Beuler, A. Krum, W. Bonath, and H. Hillmer, “Nepton processor for real-time computation of conductance-based neuronal networks,” in *2017 Euromicro Conference on Digital System Design (DSD)*, pp. 78–85, Aug 2017.

- [24] H. Shayani, P. J. Bentley, and A. M. Tyrrell, “Hardware implementation of a bio-plausible neuron model for evolution and growth of spiking neural networks on fpga,” in *2008 NASA/ESA Conference on Adaptive Hardware and Systems*, pp. 236–243, June 2008.
- [25] K. L. Rice, M. A. Bhuiyan, T. M. Taha, C. N. Vutsinas, and M. C. Smith, “Fpga implementation of izhikevich spiking neural networks for character recognition,” in *2009 International Conference on Reconfigurable Computing and FPGAs*, pp. 451–456, Dec 2009.
- [26] D. Neil and S. C. Liu, “Minitaur, an event-driven fpga-based spiking network accelerator,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, pp. 2621–2628, Dec 2014.
- [27] E. Cota, A. de Morais Amory, and M. S. Lubaszewski, *Reliability, Availability and Serviceability of Networks-on-Chip*. Springer, 2012.
- [28] J. Duato, S. Yalamanchili, and N. Lionel, *Interconnection Networks: An Engineering Approach*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002.
- [29] J.-S. Shen and P.-A. Hsiung, *Dynamic Reconfigurable Network-on-Chip Design: Innovations for Computational Processing and Communication*. Hershey, PA: Information Science Reference - Imprint of: IGI Publishing, 2010.
- [30] T. Pionteck, C. Albrecht, and R. Koch, “A dynamically reconfigurable packet-switched network-on-chip,” in *Proceedings of the Design Automation Test in Europe Conference*, vol. 1, pp. 8 pp.–, March 2006.
- [31] M. B. Stensgaard and J. Spars, “Renoc: A network-on-chip architecture with reconfigurable topology,” in *Second ACM/IEEE International Symposium on Networks-on-Chip (nocs 2008)*, pp. 55–64, April 2008.
- [32] E. Fiesler, “Neural network topologies,” 1996.
- [33] R. Miikkulainen, “Topology of a neural network,” in *Encyclopedia of Machine Learning* (C. Sammut and G. I. Webb, eds.), pp. 988–989, Boston, MA: Springer US, 2010.
- [34] J. J. Hopfield, “Neural networks and physical systems with emergent collective computational abilities,” *Proceedings of the National Academy of Sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.
- [35] H. K. Hartline, H. G. Wagner, and F. Ratliff, “Inhibition in the eye of limulus,” *The journal of general physiology*, vol. 5, no. 39, pp. 651–673, 1956.
- [36] D. Vainbrand and R. Ginosar, “Network-on-chip architectures for neural networks,” in *2010 Fourth ACM/IEEE International Symposium on Networks-on-Chip*, pp. 135–144, May 2010.

- [37] R. Coultrip, R. Granger, and G. Lynch, "A cortical model of winner-take-all competition via lateral inhibition," *Neural Networks*, vol. 5, no. 1, pp. 47 – 54, 1992.
- [38] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.
- [39] Xilinx. <http://www.xilinx.com>.
- [40] "21. underdetermined systems," in *Accuracy and Stability of Numerical Algorithms*, pp. 407–414.
- [41] E. J. Ientilucci. <http://www.cis.rit.edu/~ejipci/Reports/svd.pdf>.
- [42] C. Lawson and R. Hanson, *Solving Least Squares Problems*. Society for Industrial and Applied Mathematics, 1995.
- [43] C. Nicopoulos, V. narayanan, and C. R. Das, *Network-on-chip Architectures*. Springer, 2010.
- [44] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik, "Orion: a power-performance simulator for interconnection networks," in *35th Annual IEEE/ACM International Symposium on Microarchitecture, 2002. (MICRO-35). Proceedings.*, pp. 294–305, 2002.
- [45] P. P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh, "Performance evaluation and design trade-offs for network-on-chip interconnect architectures," *IEEE Transactions on Computers*, vol. 54, pp. 1025–1040, Aug 2005.
- [46] G. B. Bezerra, S. Forrest, M. Moses, A. Davis, and P. Zarkesh-Ha, "Modeling noc traffic locality and energy consumption with rent's communication probability distribution," in *Proceedings of the 12th ACM/IEEE International Workshop on System Level Interconnect Prediction, SLIP '10*, (New York, NY, USA), pp. 3–8, ACM, 2010.
- [47] M. Horowitz, T. Indermaur, and R. Gonzalez, "Low-power digital design," in *Proceedings of 1994 IEEE Symposium on Low Power Electronics*, pp. 8–11, Oct 1994.
- [48] M. H. Ghadiry, M. Nadi, and D. Rahmati, "New approach to calculate energy on noc," in *2008 International Conference on Computer and Communication Engineering*, pp. 1098–1104, May 2008.
- [49] X. XPE. <https://www.xilinx.com/products/technology/power/xpe.html>.
- [50] D. M. Brooks, P. Bose, S. E. Schuster, H. Jacobson, P. N. Kudva, A. Buyuktosunoglu, J. Wellman, V. Zyuban, M. Gupta, and P. W. Cook, "Power-aware microarchitecture: design and modeling challenges for next-generation microprocessors," *IEEE Micro*, vol. 20, pp. 26–44, Nov 2000.
- [51] M. J. Flynn, P. Hung, and K. W. Rudd, "Deep submicron microprocessor design issues," *IEEE Micro*, vol. 19, pp. 11–22, Jul 1999.

- [52] V. George, H. Zhang, and J. Rabaey, “The design of a low energy fpga,” in *Proceedings. 1999 International Symposium on Low Power Electronics and Design (Cat. No.99TH8477)*, pp. 188–193, Aug 1999.
- [53] Accellera. <http://www.eda.org/activities/working-groups/systemc-synthesis>.

Parameter value table for Inverse clustering with mesh network

This appendix contains the table with values for N_{router} , N_{link} and ΣE_{link} used to estimate the EDP in a 32 cluster network. The * symbol indicates that the corresponding route only passes through the inverse clustering section. Routes with ** pass through both inverse clustering and mesh sections, while those without any symbol only pass through the mesh.

Table A.1: Parameter values for Inverse clustering with mesh network of size 32

Δ	N_{router}	N_{link}	ΣE_{link}
1	2	3	3
2	3	4	4
3	4	5	5
4	5	6	6
5	6	7	7
6	2	3	6
7	3	4	7
8	4	5	8
9	5	6	9
10	6	7	10
11	7	8	11
12**	3	4	10
13	4	5	11
14**	4	6	36
15**	3	5	35
16*	1	2	32
17**	3	5	35
18**	4	5	36
19**	5	7	37
20**	6	8	38
21**	4	6	38
22**	3	5	38
23**	4	6	39
24*	3	4	48
25**	5	7	51
26**	6	8	52
27**	7	9	53
28*	5	6	56
29**	7	9	59
30*	7	8	60
31*	9	10	62

B.1 Rule structure

The *rule_struct* variables used in *solveMultipathRing.m* and *solve2DToroid.m* are sets of matrices of coefficients based on the first superposition step for cluster C0. For example, considering (B.1) we have the ω_0 corresponding matrix shown in (B.2):

$$\left\{ \begin{array}{l} \omega_0 = \mu_{0,1} + \mu_{0,2} + \mu_{0,3} \\ \omega_1 = \mu_{0,2} + \mu_{0,3} \\ \omega_2 = \mu_{0,3} \\ \omega_4 = \mu_{0,5} + \mu_{0,6} + \mu_{0,7} \\ \omega_5 = \mu_{0,6} + \mu_{0,7} \\ \omega_6 = \mu_{0,7} \end{array} \right. \quad (\text{B.1})$$

$$\omega_0 = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 1 & 2 & 3 & 0 \\ 2 & 3 & 0 & 0 \\ 4 & 5 & 6 & 7 \\ 5 & 6 & 7 & 0 \\ 6 & 7 & 0 & 0 \end{bmatrix} \quad (\text{B.2})$$