Circuits and Systems
Mekelweg 4,
2628 CD Delft
The Netherlands
http://ens.ewi.tudelft.nl/

CAS-2017-04

M.Sc. Thesis

Multi-FPGA Interconnect Simulation

He Zhang B.Sc.

Abstract

The scalable simulation of neuron communication needs a large amount of computing resources. The high throughput of data cause the high requirement of interconnect network. This thesis is aimed at the finding the efficient multi-FPGA connection for the neuron network. First describe the characteristics of the network in terms of the topology, routing and flow control. To find the efficient network structure, analysis of the throughput for the different network with different traffic pattern by considering the hopcount and bandwidth are made. It shows that the multicast in mesh topology has 33% improvement comparing to unicast. Based on the interconnect router architecture, a simulator is built to make a cycle accurate simulation in SystemC and test different traffic pattern by unicast and multicast routing. To break the limitation of FPGA ports, the source synchronous serdes connection is built by using the primitive in the Xilinx FPGA. With the requirement of bandwidth, the possible solution of number of channels and the overhead are anaylsed.

TU Delft

Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Multi-FPGA Interconnect Simulation

Thesis

submitted in partial fulfillment of the
requirements for the degree of

Master of Science

in

Microelectronics

by

He Zhang B.Sc.
born in Beijing, China

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

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 **"Multi-FPGA Interconnect Simulation"** by **He Zhang B.Sc.** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: My Graduation Date

Chairman: _____
dr.ir. TGR van Leuken

Advisor: _____
dr. Carlo Galuzzi

Committee Members: _____
dr.ir. Arjan van Genderen

_____
dr.ir. Amir Zjajo

Abstract

The scalable simulation of neuron communication needs a large amount of computing resources. The high throughput of data cause the high requirement of interconnect network. This thesis is aimed at the finding the efficient multi-FPGA connection for the neuron network. First describe the characteristics of the network in terms of the topology, routing and flow control. To find the efficient network structure, analysis of the throughput for the different network with different traffic pattern by considering the hopcount and bandwidth are made. It shows that the multicast in mesh topology has 33% improvement comparing to unicast. Based on the interconnect router architecture, a simulator is built to make a cycle accurate simulation in SystemC and test different traffic pattern by unicast and multicast routing. To break the limitation of FPGA ports, the source synchronous serdes connection is built by using the primitive in the Xilinx FPGA. With the requirement of bandwidth, the possible solution of number of channels and the overhead are anaylsed.

Acknowledgments

I would like to thank TGR van leuken, Amir Zjajo, Sumeet Kumar, Carlo Galuzzi for their assistance during the writing of this thesis. I would like to thank my parents and friends for the support during this year life. Without them, this would not have been possible.

He Zhang B.Sc.
Delft, The Netherlands
My Graduation Date

Contents

List of Figures

List of Tables

1 Introduction

With technology scaling, larger amount of neuron cells can be implemented on chip, and consequently simulation of brain behaviour could be perform more accurately. The brain simulator needs thousands/millions of neurons computing in parallel which is a challenge for the communication network. Also, the up-to-date technique can not fit such large amount of neurons in one chip which means the interconnection of multiple chips needs to be considered.

1.1 Neurons

The neuron model consist of three parts, dendrite, soma and axon hillock. Inputs from other neurons are received as electrochemical stimuli via the dendrites connection. These stimuli are transferred to the soma, where they are processed and stored as a membrane potential. If this potential reaches a certain threshold, a spike is transferred via the axon to other neurons. The neurons are connected following different patterns such as all-to-all connection, connection between neighbouring cells, or more sophisticated connection schemes based on probability or movement directions.

1.2 Problem Description

Providing highly flexible connectivity is also a major architectural challenge for hardware implementation of reconfigurable neural networks. The neural networks are formed in the different shape of connections, which may influence the performance for the hardware router connection topology. To simulate the behaviour of a highly parallel cognitive system, the communication traffic within the interconnection rises exponentially by the scalability of neurons. The previous work[3] implements a maximum 1188 Hodgkin-Huxley model neurons in one field-programmable gate array (FPGA) which are locally connected to a binary tree network with routers. The FPGA communicates with the PC through Ethernet port. To increase the number of implemented neurons beyond single FPGA, a multi-FPGA system is considered as a solution. This system should have the ability to handle the three type of real-time packets which are the initialization, the internal communication packets and outputs. The solution in this thesis gives a multi-FPGA system that is capable of simulating plenty of neurons in real-time. The design should be configured efficiently for different neuron connection topologies and scales. To speed up implementation on platforms, the design should be flexible enough and supported by formula's to use the optimal amount of critical resources given a certain platform.

1.3 Goals

To design a scalable, low-latency, high throughput network for multi-FPGA neural net emulator.

To develop efficient strategies for multicast communications between spiking neurons' clust ers.

To characterize the network performance across the network size, link design and traffic patterns.

1.4 Contributions

In this thesis, we propose a interconnect network which can be configured with the scale size, flit size, virtual channal unicast/multicast strategy in SystemC. Three traffic patterns, uniform traffic, fully-connected traffic and all-to-one traffic, are generated to test the performance of the network. We also consider the limitation of Xilinx FPGA selectIO port to construct the serdes interface using the IOSerdes primitive of the FPGA chip.

1.5 Outline

Chapter 2 gives a short explanation on what the data flow of the existing neuron system and analysing the requirement for the data rates.

Chapter 3 introduces the interconnection network and serdes interface and discuss the performance with different constraints.

Chapter 4 gives the implementation of the router in SystemC simulation and Serdes interface in VHDL.

Chapter 5 discusses the simulation result.

Background and Model Description

2

Over the years, several neuron network models, such as leaky integrate-and-fire, spiking Model by Izhikevich and HodgkinHuxley model [10], have been proposed and implemented in hardware to simulate the behaviour of the brain or part of it. Due to the high level of parallelism in neuron network and huge calculation for digital simulation, FPGA is considered as one solution to tackle this problem.

2.1 The Neuron Architecture

The calculation architecture is based on the extended version of the Hodgkin-Huxley(HH) model[6] which describes the Inferior Olive Neurons(ION) as a multiple-compartmental cell shown in Figure 2.1. Each compartment holds a certain state potential updated every simulation step. These states are dependent on the simulated chemical currents incoming or outgoing and the coupling effect which is the dendro-dendritic gap-junctional couplings present between the IONs.

The calculations are split into two groups based on the dependency of the NN topology. The first group are the calculations housed by soma and axon (hillock) compartments, which are independent of the neighbouring neurons, while the second group are depended on the coupling effect and consists of the dendrite compartment as shown in Figure 2.2 [3].

2.2 The Communication Architecture

The communication architecture is responsible for initializing and delivering requested packet containing the dendrite or axon potential to the calculation architecture, and communication to an external interface outside of the proposed design. It contains three levels, which are: interface bridge, routing network and cluster controllers as shown in Figure 2.3 [3]. The cluster controller is responsible for relating new values to the calculation architecture when requested and, storing and routing their responses while also storing the responses of other PhyCs. Information between clusters and the interface bridge is passed through a balanced tree network. The two data stream, axon hillock response and dendrite response, are transferred to the top and to all, respectively. The interface bridge is capable of communicating to and from the (virtual) boundary pins of the design and the routing network. At the interface, there will be two kinds of packets transferred, which are initial packet and output packet.

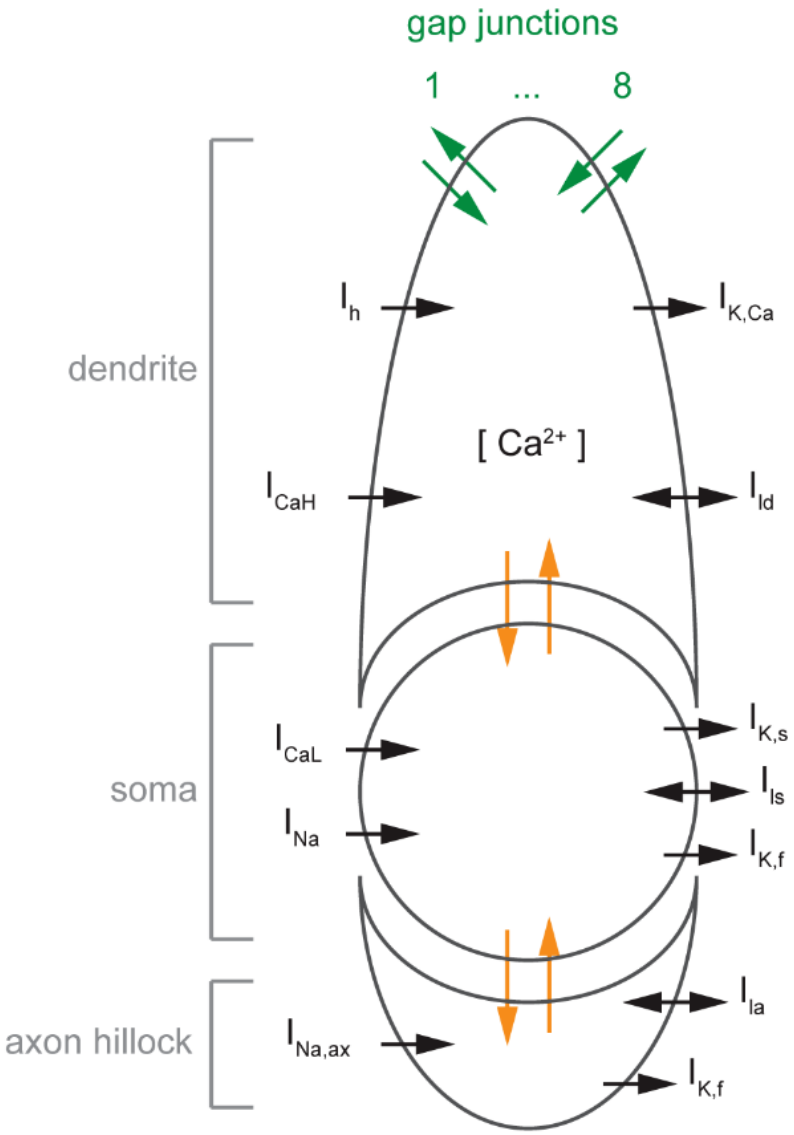

Figure 2.1: Schematic representation of the three-compartmental HH model[6].

2.3 Analysis

In the previous work [3], the physical neurons (PhyCs) are time-division multiplexed to operate for several simulation neurons (SimCs). Each SimC within the design need to calculate and communicate simulated responses to their neighbours and the axon. The calculation and communication are separately considered as shown in Figure 2.4 [3]. All these two operations are carried out within 50μs to simulate real-time of complete Neuron Network.

The initial packet consists of 5 data vectors which are given in Table 2.1. The output packet consists of 3 data vectors which are given in Table 2.2. When Initialization finishes, it returns a ready signal to start the simulation. There is no timing limit for

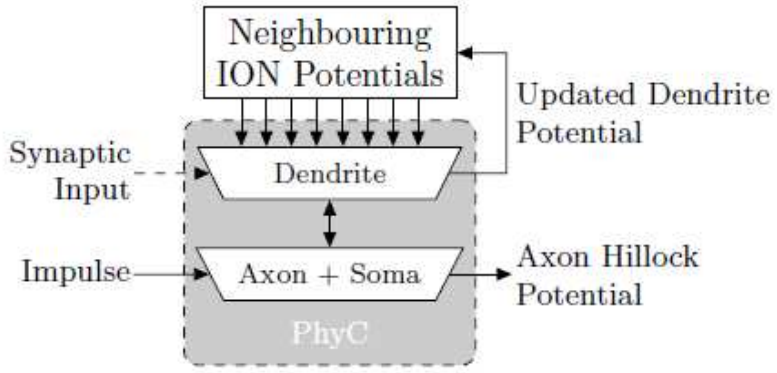

Figure 2.2: Simple unconnected ION model, showing the network dependent (Dendrite) and independent (Axon + Soma) hardware calculation elements[3]. The synapse inputs are not considered in previous work.

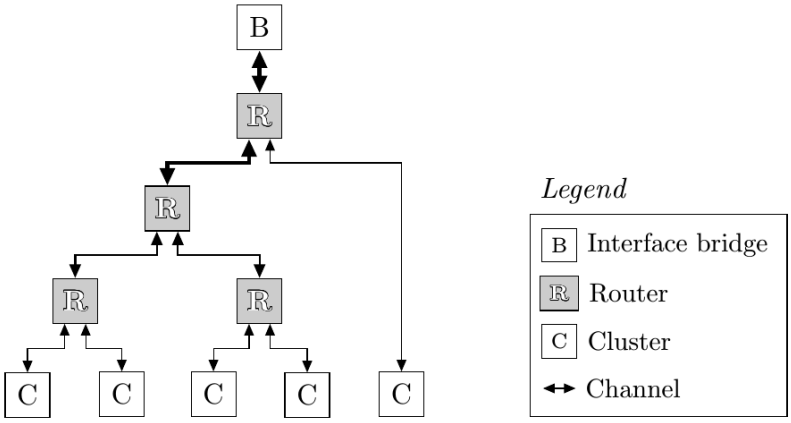

Figure 2.3: Example of a balanced tree network for a 5 cluster implementation.[3]

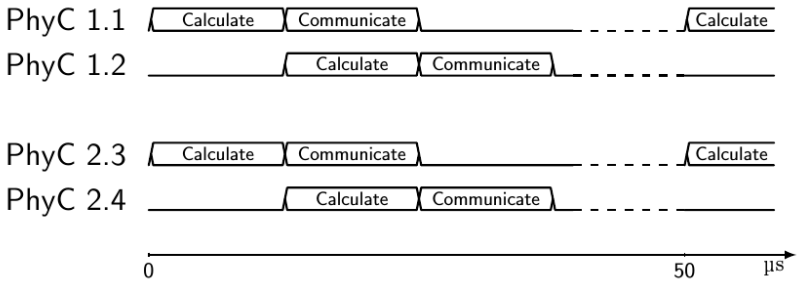

Figure 2.4: Example showing 4 SimCs that are time-shared over 2 PhyCs. Within 50s all calculating and communicating operations for all SimCs should be completed to attain real-time simulated behaviour [3].

the initial section. Real-time inputs are injected before each start signal is triggered. The outputs are generated every 50μs with each neuron having two 64-bit size data and N bits for address and 2 bits for cluster type indicator. The total outputs size is $N \times 2(64 + log_2 N + 2)$ bits which results the average bandwidth of each FPGA is total size divide by every time step which is 50μs, $2 \times 20k \times N \times (64 + log_2 N + 2)$ bit/s. With the scale of 1000 simulated neurons, the total bandwidth will be more than 3 Gbit/s.

Table 2.1: Input init data vector[3]

Used by	Name	Bits
Bridge	$cluster\_init\_clus$	$calc\_bits(\#Clusters\_per\_FPGA - 1)$
PCC	$cluster_i nit_t ype$	3
	$cluster\_init\_adr$	$calc\_bits(\#SimC - 1)$
	$cluster\_init\_adr2$	$calc\_bits(max(\#params - 1, \#Connections))$
	$cluster\_init\_data$	32/64

$calc\_bits(x) = 1 + log_2(max(x, 1))$

Table 2.2: Output data vector[3]

Used by	Name	Bits
Output	$cluster\_out\_type$	2
	$cluster\_out\_adr$	$calc\_bits(\#SimC - 1)$
	$cluster\_out\_data$	32/64

$calc\_bits(x) = 1 + log_2(max(x, 1))$

2.4 Interconnect Network

An interconnect network is the system built for transferring data between the different components. The network system has components such as buffers, channels, switches and controls to deliver data efficiently. The network can be classified into three different levels, the on-chip network, the broad-level or system level and the local or wide network. This thesis is aimed at the second level which is trying to simulate the dataflow within the multi-FPGA system. Bus is a very simple type of network and widely used in the digital systems. However it has the limited speed which can not satisfy the requirement of the modern high speed data transfer rate and is rapidly taken over. The interconnection network is built for balancing the limited resources and high throughput and low latency.

The interconnection network has the following important parameters [5]: the number of nodes, the peak bandwidth of each terminal, the average bandwidth of each node, the required latency the message size or a distribution of message size, the traffic pattern expected, the required quality of service and the required reliability and availability of the interconnection network. The number of ports corresponds to the number

of connected components and also constraint by the port resources. The bandwidth illustrates the data throughput from the network. The peak bandwidth is the maximum data rate and average bandwidth is the average data rate a node require. The message latency is also specified for the network which required the time limitation. Low latency and high bandwidth are often the trade-off requirements for the real network because of the contention for the resources. The length of message size determines the utilization efficiency of the network, where with small message the overhead may larger. The distribution of messages defines the traffic pattern and the underlying network can reduce cost by exploiting the locality of messages. The quality of service balance the allocation when contention emerges. Reliability and availability are the characteristics measuring the robust of the network. Reliability, in most situation, needs to be very high. This is achieved by adding specialized hardware and higher-level software protocol for error detection and correction. The availability is the fraction of time that the network operates correctly.

The system structure of multi FPGA system is determined in the early stage of system hardware, including interconnect structure and configuration of FPGA chip system FPGA, system structure should be the choice of logic function, the characteristics of FPGA chip, to achieve a comprehensive consideration of the system, PCB processing level, and other factors.

2.4.1 Network Basics

To meet the performance described above, the network implementation should be considered in the topology, routing and flow control. The key to the efficiency of interconnect network is the sharing of the communication channels by routing the packets between terminals. Network topology can exploit the properties of the networking bandwidth within the limited pins of chips. Routing determines the path of the packet transfer. With proper routing strategy, the length of path and the latency caused by resource contention could be balanced. The utilization of the resource may get a higher rate.

2.4.2 Topology

Selecting the topology of the network is the first step which fits the requirements of packet transfer. It is based on the cost and performance which is evaluated by the measurement such as bisection bandwidth, channel load, and path delay. The topology determines the layout and connections between nodes. The degree of each node determines the hop length of packets to transfer to the destination which also influence the throughput of the network. A mesh network is commonly used topology. A torus topology has less average hop count but with more probability causing deadlock.

The interconnection of multi-FPGA system is the topology of several FPGAs, which defines how the FPGAs connect to each other.

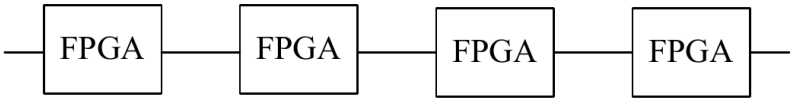

Figure 2.5: source synchronous interface

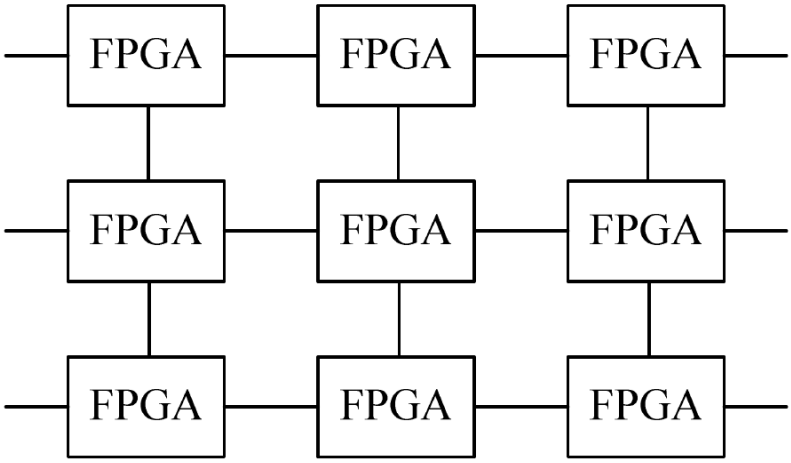

Figure 2.6: mesh topology

2.4.2.1 Linear Arrays Topology

As shown in Figure 2.5, FPGAs are placed in one dimension linear arrays and are connected to the two neighbour FPGAs.

The biggest feature of this structure is the simplicity of structure, low cost, convenient layout. This topology is used mainly in some of the early FPGA system, suitable for some special applications, such as the 32 Xilinx 3090 FPGA multi-FPGA system to run fluctuation algorithm [25].

2.4.2.2 Mesh Topology

There are several connection topology based on the mesh.

The basic structure is also known as n-array 2-mesh illustrated in Figure 2.6. The structure can be considered as two lines in one dimensional array structure based on the extension of one dimensional array, each FPGA in the FPGA system are connected with its nearest neighbors.

Torus topology shown in Figure 2.7 is an improvement on mesh topology where each FPGA connects in a ring in the same line which is also noted as n-array 2-cube.

8-way mesh architecture is shown in Figure 2.8. In the structure of 8 FPGA, each FPGA and their neighbors can be directly communicated, without routing to the other FPGA. The communication efficiency and pin utilization are better than the basic Mesh structure.

The topology of 1-hop is similar with mesh topology as illustrated in Figure 2.9. Comparing with mesh, the FPGA is not only connected to the neighbours but also to

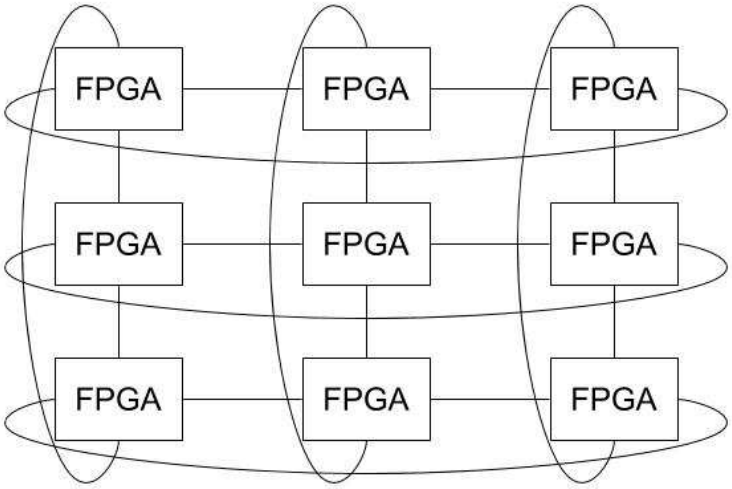

Figure 2.7: torus topology

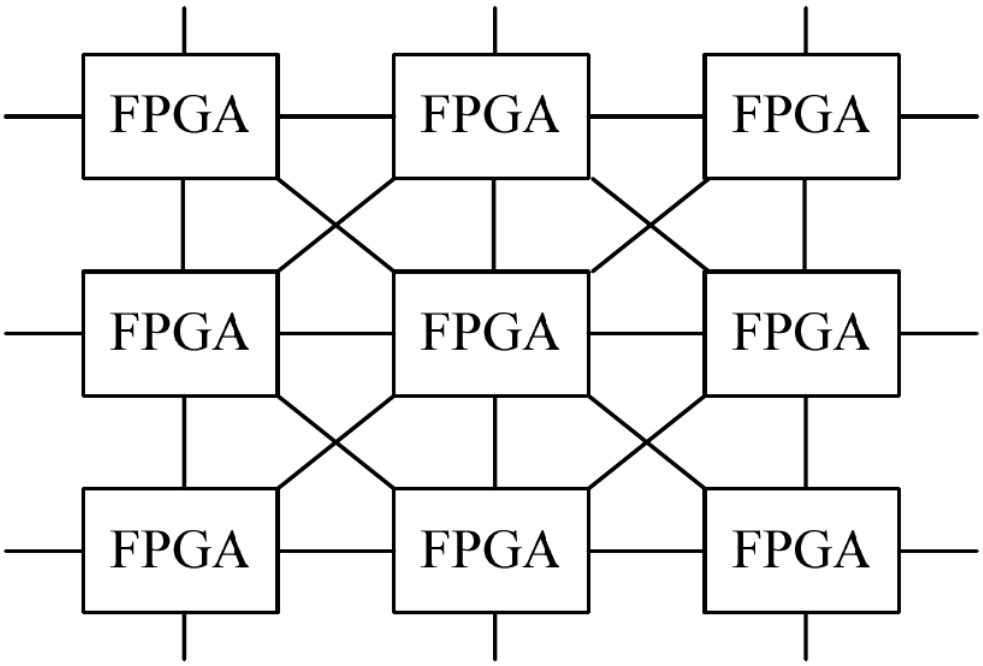

Figure 2.8: torus topology

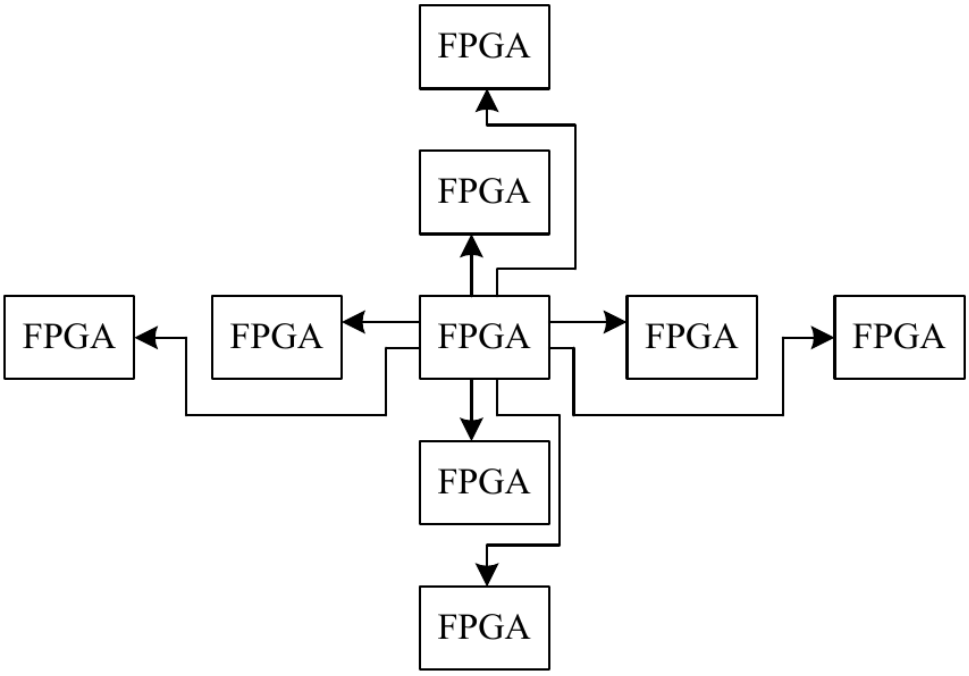

Figure 2.9: torus topology

the one-hop distance FPGAs.

2.4.3 Routing

Routing selects the path from source to the destination in the specific topology. Route strategies balance the traffic load within the channels to obtain the maximum throughput and to keep the path length minimum. The routing algorithms are classified by the selection of the routing path in three aspects: deterministic, oblivious and adaptive.

2.4.3.1 Deterministic Routing

Deterministic routing algorithms choose the same path between the source and destination within the existing possible paths. The underlying topology's path diversity is ignored by these algorithms that the balancing load unevenly distributed. In the cube networks, dimension-order routing is the commonly used algorithm [5]. The packets are routed sequentially in the dimension order which means the packet travels in one direction until it reaches the destination in that dimension. However, the simplicity of algorithm implementation and since it is deadlock-free one reasons for widespread use.

2.4.3.2 Adaptive Algorithm

Adaptive routing algorithms route the packets in the decision based on the information of the network state, which may include the channel load, the queue length of buffers and status of a node or link.

2.4.4 Multicast Routing

Multicast routing is widely used for transferring one packet to several processing nodes which is special for coherence protocol. In the neuron network the neuron state is duplicated to multiple different destination neurons. Using multicast communication could decrease the sent packets and improve the traffic efficiency. In multicast routing deadlock prevent is more difficult because of more complicated ports allocate. In SpiNNaker, the deadlock is examed by counter and the idle packet is first trying to send emergency route channel. If it is out of the threshold of timer, the packet is regard as deadlock and dropped. In Neurogrid the deadlock is prevented by the tree topology and up-down communication protocol. There is no cyclic buffer dependence.

Multicast routing algorithms can be classified as path-based, and tree-based [14]. In unicast-based algorithm, the packets are sent one by one to the multiple destination. This incurs no additional resource but start-up delay. Tree-based multicast routing in wormhole has difficulty in the deadlock avoidance[15].

2.4.5 Flow Control

Flow control determines the allocation of the network resources when packets are transferred. The flow control method is aim to achieve a high utilization of ideal bandwidth and deliver the packets in the low latency with the efficient allocation of resources. It deals with the contention when more than one packets waiting for the same resource. The packets are often split into flow control digit (flit) for the allocation of the channel to keep the overhead for routing and get efficient resource allocation and low block latency.

The simplest flow-control is by dropping or misrouting the contention packets. The dropping packets are re-transmitted in the control of timeout or NACK signal. If the next router send a NACK signal or the ACK signal is out of time, the packets are regarded as contention and new ones are retranmitted. Although the dropping mechanic is simple, it occupies the bandwidth with the dropped packets which decrease the efficiency.

Circuit switching generate the transfer circuit and send packets through it without buffer. The process has four phases. The first phase is to send the request to the destination. If there is no contention, an acknowledge will reply to the source in the second phase. The data and tail flits travel in the third and forth phase respectively. Once the destination get the tail flit, the channels will be deallocated. Though it does not waste resource with dropped packets, the latency and throughput are constrainted by the circuit establishing.

2.4.5.1 Packet-buffer Flow Control

Store-and-forward and cut-through are the flow control methods that allocate buffers and channel bandwidth to packets.

Store-and-forward

In the store-and-forward flow control, each packets are completely receive before it is transferred to the next node. To facilitate this method, the two resources are allocated

before transferring: the transfer channel, and the buffer for the store the packet in the next hop.

Cut-through:

Cut-through flow control overcomes store-and-forward flow control's latency penalty by forwarding a packet as soon as the header is received and resources are acquired, without waiting for the complete packet to be received. The deadlock properties are the same in store-and-forward and virtual cut-through routing strategy[4].

2.4.5.2 Flit-buffer Flow Control

Worm-hole:

Wormhole flow control and cut-through flow control are similar, although the buffers are allocated in the count of flits rather than packets. The packets are splited to three kind of flits, the head flit, the body flit and the tail flit. The head flit requires for the channel allocation of the whole packet. Body and tail flit route along the path of the head flit. The channels are deallocated after the pass of the tail flit and only need a flit size of buffer to route the flit. Comparing to cut-through flow control, worm-hole require less buffer to store the flit rather than the packet which is the main difference of them. However, the conjection may occur for the same reason that only the flit is buffered. This is the trade-off between the cut-through and worm-hole flow control that need to be considered in the design.

2.4.5.3 Virtual-channel

Virtual-channel flow control, which associates several virtual channels (channel state and flit buffers) with a single physical channel, overcomes the blocking problems of wormhole flow control by allowing other packets to use the channel bandwidth that would otherwise be left idle when a packet blocks.

2.4.5.4 Buffer Management and Backpressure

Credit-Based Flow Control:

With credit-based flow control, the upstream router keeps a count of the number of free flit buffers in each virtual channel downstream. Then, each time the upstream router forwards a flit, thus consuming a downstream buffer, it decrements the appropriate count. If the count reaches zero, all of the downstream buffers are full and no further flits can be forwarded until a buffer becomes available. Once the downstream router forwards a flit and frees the associated buffer, it sends a credit to the upstream router, causing a buffer count to be incremented.

2.4.6 Flit and Credit Encoding

There are basically two ways to transfer flits and credits by separate channel or same channel. For separate channel, the bandwidth for flit and credit are different. It will increase the usage of ports which is limited resource in the FPGA implementation. For the implementation of sharing channel, the credit can be added at the back of the flit or multiplexing with flits using phit level protocol [5].

2.4.7 Deadlock Avoidance

Deadlock occurs when a group of packets waiting for others' resources and the dependency between each other generate a ring relation. The absence of cyclic dependencies is a sufficient and necessary and condition for deterministic routing[7]. The deadlock can be avoided by the routing algorithm which has no ring path such as dimension order, west first algorithms[4]. To break the resource dependency, the resources are ranked in different level which eliminate cycles.

In SpiNNaker[18] deadlock is monitor by counter and controlled by dropping mechanism. However dropping mechanism may cause the packet loss which is not allowed for our real time brain machine. To guarantee the deadlock free without dropping, routing strategy need to be proved to deadlock free which is depend on the routing algorithm.

2.4.8 Router Architecture

During design of router architecture area and power constraints are considered. A typical credit-based virtual channel router architecture is shown in Figure 2.10. The major components are buffers consist of virtual channels, route computing component, virtual channel allocation switch allocation and crossbar. Buffers restore the incoming packets. The route compute logic generate the output port direction of the flits. the allocator and arbiter are for virtual channel selection and switch allocation.

2.4.9 Routing Mechanics

2.4.9.1 Table-based

The value of the relation for each pair of inputs is stored in the table and the table is indexed by the inputs. The major advantage of table-based routing is generality. Subject to capacity constraints, a routing table can support any routing relation on any topology. A routing chip that uses table-based routing can be used in different topologies by simply reprogramming the contents of the table. With source routing, all routing decisions for a packet are made entirely in the source terminal by table lookup of a precomputed route. Each source node contains a table of routes, at least one per destination. It is widely used for its simplicity, speed and scalbility in the routing methods for deterministic and oblivious routing. However it is not suitable for the adaptive routing because no information about the traffic is involved. Table-based routing can also be performed by storing the routing table in the routing nodes rather than in the terminals. Node-table routing is more appropriate for adaptive routing algorithms because it enables the use of per-hop network state information to select among several possible next-hops at each stage of the route.

2.4.9.2 Algorithm-routing

Instead of storing the routing relation in a table, we can compute it using an algorithm. For speed, this algorithm is usually implemented as a combinational logic circuit. Such algorithmic routing sacrifices the generality of table-based routing. The algorithm is specific to one topology and to one routing strategy on that topology. However,

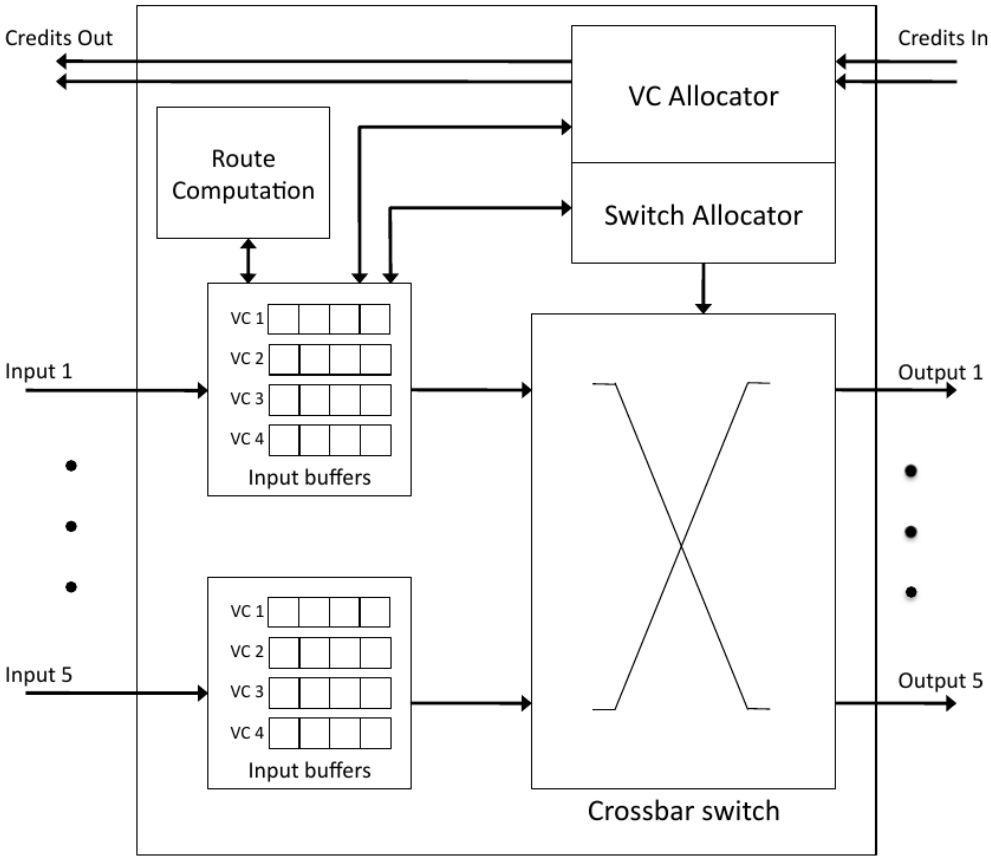

Figure 2.10: basic router architecture

algorithmic routing is often more efficient in terms of both area and speed than table-based routing.

2.4.10 Performance of Interconnect Network

Performance of an interconnect network is described primarily by average latency of a packet, which is defined as the time from when the first bit of the packet arrives at the source terminal to when the last bit of the packet arrives at the destination terminal, as a function of offered traffic, the average amount of traffic (bits/s) generated by each source terminal of the network. To explore topology, routing, and flow control in the early design stages, some incremental approach are used. To find the lower bound on the average latency, the zero-load latency is analyzed through a network[5]. The zero-load assumption is that the resources are allocated to one packet. Under this assumption, the average latency of a packet is its hop latency and serialization latency. The average hop count in different network with the actual routing algorithm gives a tighter bound because the average hop count is no less than the minimal hop count. Finally, the flow control employed by the network can further reduce the performance over the bounds given by the topology and the routing.

A similar approach gives a set of upper bounds on the throughput of the network. While each source offers a particular amount of traffic to the network, the throughput, or accepted traffic, is the rate that traffic (bits/s) is delivered to the destination terminals. In the random routing, half of the traffic transfer through the bisection of the network. This gives bisection bandwidth a upper bound on the throughput which is under the assumption of balanced traffic. A paticular routing algorithm may not have the perfect balance. The throughput of routing algorithm θ cannot get to the bisection bandwidth and never exceed it. Finally, if our flow control results in idle channels due to resource dependencies, the saturation throughput of the network s can be significantly less than the bound of θ.

2.5 Interface

In order to transmit a large number of bits in parallel mode, Serializer / deserializer (SerDes) combination circuits are usually used for compensation in the high-speed limited I / O system. Typically, the serializer circuit converts the parallel n-bit data to a faster 1 bit signal. The encoded SerDes circuit in some popular applications such as DVI / HDMI, serial ATA, USB 3.0, Gigabit Ethernet and PCI Express.

2.5.1 SerDes

The Serializer/Deserializer is a Time-Division Multiplexing(TDM), point-to-point communication technology. The low-speed parallel signal is transferred into a high-speed serial signal at the transmitter. Through the transmission medium (cable or copper Line), the high-speed serial signal at the receiving end is re-converted into low-speed parallel signal. This point-to-point serial communication technology can effectively solve inter-symbol interference, signal crosstalk, DC drift and PCB routing problems in effect making full use of the transmission medium channel capacity, reducing the required transmission medium and the number of pins and chip area, realize Low-cost medium-long distance high-speed communication.

2.5.2 Asynchronous Implementation

The 8b/10b SerDes has characteristic of high transfer rate, long transfer distance, good anti-noise performance, easy to error detection and other advantages which makes it a study hot spot and has been widely used. An encoder, a serializer and a transmitter form the sender circuit. A receiver, a clock generating circuit and a clock recovery circuit form the receiver circuit. Encoders and decoders complete coding and decoding functions, of which 8b / 10b and 64b / 66b are the most commonly used coding schemes. The serializer and deserializer are responsible for parallel to serial and serial to parallel conversion. The clock generation circuit is usually implemented by a phase-locked loop (PLL). The deserializer requires a clock data recovery circuit (CDR) to provide a recovered clock and a retiming time of the data. The clock recovery circuit is usually achieved by the phase-locked loop. The transmitter and receiver perform differential signal transmission and reception, where LVDS and CML are the most commonly used

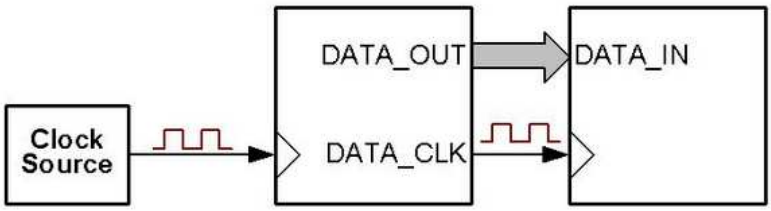

Figure 2.11: source synchronous interface

two differential signal level standards. In addition to these conventional modules, there exists several additional used for testing, built-in bit error rate test, pre-increase the circuit and so on.

2.5.3 Synchronous Implementation

Parallel clock SerDes is often used to convert data, addresses, and control parallel buses into serial data. The SerDes interface does not serialize the address, data and control bus signal into a serial signal, but serialize them respectively into serial address signal, serial data signal and serial control signal. These serial signals will be sent to the receiver together with the clock signal. The receiver will sample the serial data with the received clock signal, and the multiplexer will convert the serial data into parallel signals. While transmitting the data signal, the SerDes of this structure also needs to send a clock signal in parallel to receive the synchronization signal at the receiving end. The structure is shown in Figure 2.11.

Compared with parallel bus, parallel clock SerDes greatly reduces the number of signal transmission lines, reduces system power consumption and signal crosstalk, and drives longer cables. Moreover, parallel clock SerDes can carry multiple serial signal which can effectively reduce the transmission rate of a serial channel. This extend the traditional bus transmission distance to the range of several meters. Thus, parallel clock SerDes is often used for stackable Ethernet switches, interconnection between racks and racks. However, the parallel clock SerDes needs parallel transmission of a clock signal, which makes the transmission distance limited. In the design of the system, it requires careful consideration due to the effect of clock skew caused by the channel of the system, so as to avoid the timing problems in the system.

2.6 Introduction of FPGA Basics

FPGA consists of the input output block (IOB), the configurable logic block (CLB) and the interconnect [25].

2.6.1 Input Output Unit

Programmable input / output unit is the interface part between the FPGA and the external circuit. It performs the drive and matching requirements for the input /

output signals under different electrical characteristics and Provide input buffer, output driver, interface level conversion, impedance matching and delay control function. The FPGA I/O is classified by group and each group can support different I/O standards independently. Through the flexible configuration of the software, it can be adapted to different electrical standards and I/O physical characteristics, adjust the size of the drive current, change the upper and lower resistance. At present, I/O port frequency is continuously increasing, and some high-end FPGA through DDR register technology can support data rates up to 2Gbps.The external input signal can be input to the FPGA through the buffer of the I/O block (IOB) module, and can also be input directly to the FPGA interior.

In order to facilitate the management and adapt to a variety of electrical standards, FPGA IOB is divided into several groups (bank). Each bank interface standard is determined by the interface output supply voltage (VCCO) and can only have a VCCO, but different bank VCCO may be different. Only ports with the same electrical standard can be connected together. The same VCCO voltage is the basic condition of the interface standard.

2.6.2 Configurable Logic Block

CLB consists of multiple identical Slice and additional logic. Each CLB module can be used not only for combinational logic, temporal logic, but also for distributed RAM and distributed ROM.

2.6.3 Configurable Interconnection

Programmable internal interconnect resource connects blocks of programmable logic blocks or input / output blocks to form specific functions. Users can programmatically determine the functionality of each unit and their interconnections to achieve the required logic functions.

2.6.4 Xilinx select IO resource[24]

All 7 series FPGA support a variety of standard interfaces and SelectIO drivers and receivers. There are two kinds of bank resources high performance (HP) and high range (HR) both contains inputs, outputs, and 3 state drivers. HP bank has separate IDELAY and ODELAY modules. The HR bank has an extra ODELAY module than the HP bank. The additional resources Input serial-to-parallel converters (ISERDESE2) and output parallel-to-serial converters (OSERDESE2) support very fast I/O data rates. It allows the internal logic to run up to 8 times slower than the I/O. The ISERDESE2 primitive in the 7 series FPGA is a dedicated serial-to-parallel converter with specific clock and logic characteristics designed to facilitate the implementation of high-speed source synchronization applications. ISERDESE2 avoids the extra timing complexity encountered when designing deserializer in the FPGA schema. ISERDESE2 deserializer can achieve high speed data transmission without matching the input data frequency with the FPGA schema. The converter has single data rate (SDR) and double data rate (DDR) mode. In SDR mode, the serial parallel converter can produces 2-, 3-, 4-,

5-, 6-, 7- or 8- bit parallel word. In DDR mode, it supports 4-, 6-, 8- bit parallel mode in one ISERDESE2 and 10-, 14- wide word when using two cascaded ISERDESE2. The maximum speed can achieve 1200 Mbps in the DDR mode[23].

2.7 Other Implementations

2.7.1 Neurogrid

The Neurogrid uses neurons in with analogue model in subthreshold and communicate spikes in digital signal. The Neurogrid chip has four main components, a Neuron array, a receiver and transmitter, a RAM and a Router[16]. The silicon neurons are organized in a mesh array which are controlled by receiver and transmitter to go through spikes. The transmitter is 256×256 size and receiver, which selects shared-synapses to activate or disables a neuron, is 2048×256 size in Neuroncore. The massive connectivity data is saved in RAM for reconfigure. Neurogrid chips communicate in a tree hierarchy topology where each node connects to one parent and two children. There is a router built in each chip which has 4 directions to parent, two children and itself[16]. Deadlock for multicast is eliminated by the up-down, point-to-point routing strategy. Each spike is first passed up to a top node and then transported to the destination node. The physical implementation has 256×256 silicon neurons with each one has a soma, a dendrite and shared synapses circuits. The consumption of Neurongrid is 2.7w with total 983040 neurons on one chip. A limitation of Neurongrid is lack of synapses plasticity.

2.7.2 SpiNNaker

SpiNNaker is a massively parallel computer that is suitable for computational neuroscience modeling of large-scale spiking neural networks in biological real time[8]. The architecture of SpiNNaker is of 2 dimensional node array which consists of 18 ARM968 cores and network-on-chip circuit for communication. Each node also has 96kB of local memory and 128MB shared memory. The data type used is 32bit integer. Each ARM core can model hundred neurons and 10^3 synapse connections with the peak speed of 10 million connection per second. The connectivity mechanism is divided into several stages by function, each of which is pipelined. The first stage identifies packet errors to ensure the correctness of the data flow on the Communication NoC. The second stage is all of these units operate in lock-step as a single pipelined routing stage. The final stage is responsible for adapting the output route to any congestion or faults in the subsequent asynchronous interconnection. multicast packet-routing strategy through a three-valued associative lookup table based on address event representation (AER). The routing algorithm they used is called neighbor exploring routing (NER) [19] which generate tree-based routing path on the closet route segment. The transport of spike is achieved by the router within globally asynchronous locally synchronous (GALS) communication system. The router has the direction of six neighbor and local. The routing table has 1024 entries for each node and is kept in content addressable memory (CAM). In processing, one of 18 cores is selected as operating system and remaining 16 cores are for application the last one is for redundancy fault-tolerance. The system NoC

shares chip resources i.e. on chip system RAM and ROM, Ethernet media-independent interface controller[20]. 48 SpiNNaker chips connect in hexagonal topology and three FPGAs to implement SpiNNlinks for inter-board communication channels on PCB. The connections between chips on the PCB are direct wired connections using a self-timed 2-of-7 non-return-to-zero protocol to transmit 4 bits symbols with two wire transitions and one wire transition for the asynchronize acknowledge response. Eight asynchronous channels are controlled by a SpiNNlinks into a serial 3-Gbps I/O channel using SATA cables for board-to-board interconnect. On-chip communication has the bandwidth of 5.0 Gb per second and off-chip link bandwidth is 250Mb/s. Router can receive a 5.3 Gb input data per second. Each chip has 360 mW Idle power consumption. For 1000 neurons simulation, it consumes 12 nJ/ms per neuron and 4 nJ for connection[8].

2.7.3 IBM TrueNorth

TrueNorth is a large scale neurosynaptic architecture developed by IBM in the DARPA SyNAPSE program. The TrueNorth core[17] is inspired by the organization and function of biological brain which has four main elements neurons, axons, dendrites and synapses. The TrueNorth architecture is a network of neurosynaptic cores which can generate scalable and flexible spikes. The neurosynaptic core in crossbar topology is the key block which connects axons to neurons. Each core has 256 neurons as outputs and 256 axons as inputs with the complete connection that is 256 by 256 programmable synapses. The output of synapses are coded into digit and fed to the digital integrate-and-fire neuron model which has 23 configurable parameters. Some characteristic of synapses connection, the neuron leakage and threshold are generated by digital pseudo-random sources. To scale up, an upper layer of architecture is used which consists of 4096 routers. These routers with five ports (four directions and local) are arranged in a two-dimensional mesh that transport spikes from neurons to any axons in any neurosynaptic core using the point-to-point router strategy. The algorithm is routing the dimension-order and deadlock-free. Axonal delay is implemented at the target. A merge-spilt structure is used during communication of interchip. At four edges of the mesh, spikes are split to enter the chip from shared link and merged to transport within chips. The physical chip of TrueNorth is in fully digital function which has 1 million neurons and 256 million synapses. Because of the architecture and 28nm process technology, the power density is 20mW per cm^2 which is substantially lower than conventional central processing unit and graphic processing unit.

2.7.4 CxQuad and ROLLS

A convolution neural network system has been developed in University of Zurich and ETH Zurich[9]. The system is implemented using full-custom neuromorphic devices, which can classify visual inputs to outputs without additional computing or memory resources. The system includes a Dynamic Vision Sensor(DVS), a PCB with 9 "cxQuad" multi-neuron chips, and a Reconfigurable On-Line Learning Spiking (ROLLS) neuromorphic processor. The DVS is used to report the location where the scene contrast of a pixel has changed. The cxQuad chip is a multi-core device which contains analog neuron and synapse circuits and asynchronous routing fabric. Each cxQuad chip has

1k neurons and 64k synapses distributed among 4 cores. The synapse block comprises a linear integrator circuit which integrates input events from 64 12-bit programmable Content Addressable Memory (CAM) cells. Output events generated by the neurons can be routed to the same core, to other cores on the same chip, as well as to cores on different chips. 4k 12-bit Static Random Access Memory (SRAM) at different levels are used to store the routing information. The classification layer is implemented by the ROLLS chip, which comprises 256 neurons and 133,120 synapses. The neuron circuits are the same as in the cxQuad chip, while the synapse circuits have three different types, each of which has different functions in the learning process.

Experiment shows the system can be trained to recognize visual shapes. The system is first configured to have 8 different pools of neurons trained to perform binary classification of 8 different visual patterns. Then visual stimuli were flashed on a monitor and sensed by the DVS. The DVS outputs were then sent to the cxQuad board using a direct cable connection, and convolution and pooling operation took place on the cxQuad board in parallel and in real-time. After a few milliseconds, the results of the convolution network started being transmitted to the ROLLS chip for classification. After the first spikes received, the pool of 32 neurons that was trained to recognize the input stimulus presented response with an average firing activity higher than all other pools.

2.7.5 BrainScaleS

BrainScaleS is an analogue circuit system with high-speed and large-scale for neuromorphic computing[21]. The neuron model in BrainScaleS is the exponential integrate-and-fire model called AdExp-model. Each neuron implemented in 180 nm CMOS technology costs an area of 150 by 10 μm^2. Internal currents are controlled from 100 nA to 1 μA that makes the operating point stay upon deep sub-threshold region which prevents the strong noise associated with it. A structure called Analog Network Core is built to mimic the synapses' characteristic connecting neurons. In the middle of ANC is a group of dendrite membrane circuits which are programmable by configuring 23 independence parameters. Each den-mem connect to 224 synapses whose weight is digital value of current generated by DAC. Spike-timing dependent plasticity (STDP) and short-term depression and facilitation are the plasticity levels used in the synapse circuit. A single wafer contains 4$\hat{1}$0 synapses and up to 180k neurons. The communication between ANCs is implemented by a specific asynchronous serial event communication protocol which has 2Gb/s bandwidth. In the channel, low-voltage differential signal decreases the power consumption. To reduce the number of connections, eight chips containing one ANC consist one reticle. An additional metal layer put upon along reticles. The communication between wafers is handled via 1 or 10 Gbit Ethernet links.

2.8 Conclusion

In this section, a brief description of HH model is given. By using an extended version of the HH, a neuron can be split into 3 different compartments. Each compartment models the behaviour of a certain area of the ION (Dendrite, Some and Axon Hillock).

Through work done by [3], the PhyC resources are time shared over several SimCs to calculate multiple ION responses. The generated responses are transferred in the router network and fed to the PhyCs for the next step. By doing this the most 1188 SimCs are simulated on a Virtex7 device.

To scale the number of neurons, a robust interconnect network between FPGAs is considered. The technology of interconnect network and SerDes interface are depicted which are used in this design. The implementation will be focus in chapter 4.

3 Network Analysis

This chapter presents the analysis of the network specification and characteristics.

Due to the concurrent processing of each neuron model, a large amount of data is generated at each simulation step. These data are transferred in two directions, to the outside of the system and to the destination neuron, which cause the high throughput traffic. The traffic is not determined for the real-time configuration of neuron connections. The connection matrix also has influences on the traffic. This came with a challenge of building the robust high-throughput communication network.

3.1 Network Specification and Constraints

The network design starts with a set of specifications that describe the requirement for the neuron network model. As illustrated in the chapter 2, the specification is shown in the following.

The bandwidth requirement is specified by the data throughput. In the Jan's work [3], the message consists of a 32/64 bits data, an address covering all the neurons and 2 bits of data type which indicate the usage of data value. So the message size is $64 + log_2 N + 2$. The average bandwidth of each FPGA is $40000 \times N \times (64 + log_2 N + 2)$ bit/s. Assuming the number of neuron N is 1000, then the total bandwidth will be more than 3 Gbit/s. The peak bandwidth is achieved when the communication is in cycle accurate, this means every cycle $64 + log_2 N + 2$ bit of data and address need to be transferred. The peak bandwitdh will be 7.6Gbps if we assume 10ns clock frequency. The message latency here is defined as the cycles it takes to transfer the message from the source node to the destination node. The message latency has the relation with the calculation cycles that the total operating cycles should be within 5000 (for 10ns clock period). As depicted in chapter two, $latency + M * cal\_latency < 5000$. The lower the transfer latency is, the larger amount of the neurons can be simulated.

3.2 Topology Analysis

Providing highly flexible connectivity is a major architectural challenge for hardware implementation of reconfigurable neural networks. The neural network are formed in different shape of connections which may influence the performance for the hardware router connection topology. To examine the relation between the allocation between neural network and hardware resources, the analysis is based on the several neuron distribution and routing typologies with three packet-based communication methods (unicast, multicast and broadcast).

Table 3.1: Specification for the interconnect network

parameter	value
input ports	5
output ports	5
peak bandwidth	7.6Gbps
average bandwidth	3Gbps
message latency	¡5000
message size	$64 + log_2 N + 2$ bit
traffic pattern	neighbor connected
No. of FPGAs	¡ 25

3.2.1 Performance and Cost Metrics

Throughput depends on the parallelism in the architecture and the interconnect bandwidth [1]. In order to compare different architectures that provide the same throughput and end-to-end delay, an Effective Bandwidth as the actual communication bandwidth or throughput is defined as the following [1] :

$$BWeff = \frac{\sum num\_link}{TotalHopCount} f \times U$$

Where *num_link* is number of links between each node; *TotalhopCount* is the average total distance traversed by each packet (going to all destinations), measured in the number of hops; f is the operation frequency and U is the utilization of the architecture [22]. We assume that the router works in the realistic condition which transfer the all input data . All circuits works in the same clock frequency. The cost is compared by the wiring area and memory size.

$$LinkWidth = W \sum num\_link$$

$$MemSize = num\_router * num\_neuron$$

The cost-performance ratio R is:

$$R = \frac{BWeff}{LinkWidth * MemSize}$$

3.2.2 Mesh Topology

A n-ary mesh comprises n processors and n routers arranged in a $\sqrt{n} \times \sqrt{n}$ mesh. The total number of links in 2D-mesh is

$$LinkWidth_{mesh} = 2\sqrt{n}(\sqrt{n} - 1)$$

The routing table is stored in each router with the size of n and n^2 in total.

$$MemSize_{mesh} = n^2$$

3.2.3 Tree Topology

The binary tree topology has n neurons at the leaves and n-1 routers compose the branches. The fat tree has the same topology but the links increase to the root that the number of links in each level remains the same of n. The total number of links in binary tree is

$$LinkWidth_{tree} = 2(n-1)$$

and in the fat tree is

$$LinkWidth_{fatTree} = nlog_2n$$

The routing table is stored in each router with the size of n and $n(n-1)$ in total.

$$MemSize_{tree} = (n-1)n$$

3.3 Different Traffic Pattern Analysis

3.3.1 Hopfield Neural Network

Hopfield network is the network with complete connection to other neurons. It is the upper bound of the condition that the network get the maximum traffic. Here we emulate Hopfield network on 2D-mesh and tree topology. The average distance between two nodes in 2D-mesh and tree is $\frac{2}{3}\sqrt{n}$ and $[2n(log_2n-1)+2]/(n-1)$ respectively. The total hop count for one neuron packet is sum of one neuron to the others:

$$TotalHopCount^{mesh}_{unicast} = (n-1)\frac{2}{3}\sqrt{n}$$

$$TotalHopCount^{tree}_{unicast} = 2n(log_2n-1)+2$$

so the throughput is

$$BWeff^{mesh}_{unicast} = \frac{3}{\sqrt{n}+1} = O(n^{-\frac{1}{2}})$$

$$BWeff^{tree}_{unicast} = \frac{n-1}{n(log_2n-1)+1} = O(\frac{1}{log_2n})$$

$$BWeff^{fatTree}_{unicast} = \frac{nlog_2n}{2n(log_2n-1)+2} = O(1)$$

Because of the completely connection of hopfield network the multicast and boardcast is the same. The total hop count of broadcast in mesh topology is the number of edges in spanning tree which is $n-1$ and in tree topology is $2(n-1)$. Based on the equation of throughput,

$$BWeff^{mesh}_{broadcast} = \frac{2\sqrt{n}}{\sqrt{n}+1} = O(1)$$

$$BWeff^{tree}_{broadcast} = O(1)$$

$$BWeff^{fatTree}_{broadcast} = \frac{nlog_2n}{2(n-1)} = O(log_2n)$$

Table 3.2: complexity of different topology implementing Hopfield network

	TotalNumLink	TotalHopCount	ThroughPut
Mesh unicast	O(n)	O(nˆ1.5)	O(nˆ-0.5)
Mesh multicast	O(n)	O(n)	O(1)
Mesh broadcast	O(n)	O(n)	O(1)
Binary tree unicast	O(n)	O($n\log_2 n$)	O($\log_2 n^-1$)
Binary tree multicast	O(n)	O(n)	O(1)
Binary tree broadcast	O(n)	O(n)	O(1)
Fat tree unicast	O($n\log_2 n$)	O($n\log_2 n$)	O(1)
Fat tree multicast	O($n\log_2 n$)	O(n)	O($\log_2 n$)
Fat tree broadcast	O($n\log_2 n$)	O(n)	O($\log_2 n$)

The cost metric is defined here as wire area multiply memory size. For the Hopfield fully connected network the routing table is only generated in the unicast condition which means the memory size is 1 in broadcast. The cost for each condition are as following:

$$Cost^{mesh}_{unicast} = 2\sqrt{n}(\sqrt{n}-1) * n^2 = O(n^3)$$
$$Cost^{mesh}_{broadcast} = 2\sqrt{n}(\sqrt{n}-1) = O(n)$$
$$Cost^{tree}_{unicast} = 2(n-1) * (n-1)n = O(n^3)$$
$$Cost^{tree}_{broadcast} = 2(n-1) = O(n)$$
$$Cost^{fatTree}_{unicast} = nlog_2 n * (n-1)n = O(n^3 log_2 n)$$
$$Cost^{fatTree}_{broadcast} = nlog_2 n = O(nlog_2 n)$$

3.3.2 RNDC Neural Network

In realistic scenarios, neurons are in random exponential local connectivity [13]. The Randomly Connected NN (RNDC) is the probability of connection between neuron a to b in the definition

$$p(a,b) = \frac{C}{2\pi\lambda^2} e^{-D(a,b)/\lambda}$$

where D(a,b) is the Euclidean distance between a and b, λ is a spatial connectivity constant reflecting the average traversing distance. Here we map the neurons to the mesh topology by the distance. The average distance is

$$avgDist = \frac{C}{2\pi\lambda^2} \int \sqrt{D} e^{-D/\lambda} dD = 2\lambda C$$

where $D = \sqrt{x^2 + y^2}$

The unicast total average hop count is

$$TotalHopCount^{mesh,RNDC}_{unicast} = 2\lambda C^2$$

$$BWeff^{mesh,RNDC}_{unicast} = \frac{\sqrt{n}(\sqrt{n}-1)}{\lambda C^2} = O(n^{-\frac{1}{3}})$$

where $\lambda \approx \sqrt[3]{n}, C \approx \sqrt{n}$

For the multicast, the total number of hops is approximated by the average distance then traversing one hop to the C connected ones.

$$TotalHopCount_{multicast}^{mesh,RNDC} = (2\lambda + 1)C$$

$$BWeff_{multicast}^{mesh,RNDC} = \frac{2\sqrt{n}(\sqrt{n}-1)}{(2\lambda+1)C} = O(n^{\frac{1}{6}})$$

Broadcast has the same hop count comparing to fully connected.

$$BWeff_{broadcast}^{mesh} = \frac{2\sqrt{n}}{\sqrt{n}+1} = O(1)$$

To simulate RNDC network in tree topology, the distance is calculated as

$$D(n) = log_2 n$$

where n is the number of neuron.

The average hop count for unicast is

$$avgDist() = \frac{C}{2\pi\lambda^2} * (\frac{2}{e})^{logn} \frac{logn - 1}{2e^{-\frac{2}{\lambda}} - 1} = O((\frac{2}{e})^{logn} logn)$$

$$TotalHopCount_{unicast}^{tree,RNDC} = O((\frac{2}{e})^{logn} n^{\frac{1}{3}} logn)$$

$$BWeff_{unicast}^{tree,RNDC} = \frac{2(n-1)}{TotalHC} = O(n^{\frac{2}{3}}(\frac{e}{2})^{logn} logn)$$

$$BWeff_{unicast}^{fatTree,RNDC} = \frac{nlogn}{TotalHC} = O(n^{\frac{2}{3}}(\frac{e}{2})^{logn})$$

The average hop count for multicast is first to average hop count of unicast and then to the C neuron in the branches

$$TotalHopCount_{multicast}^{tree,RNDC}$$

$$= avgDist(n) + C * avgDist(C)$$

$$= O(n^{\frac{1}{2}} logn)$$

$$BWeff_{multicast}^{tree,RNDC} = \frac{2(n-1)}{O(n^{\frac{1}{2}} logn)} = O(\frac{n^{\frac{1}{2}}}{logn})$$

$$BWeff_{multicast}^{fatTree,RNDC} = \frac{nlogn}{O(n^{\frac{1}{2}} logn)} = O(n^{\frac{1}{2}})$$

The broadcast performance is the same as Hopfield connection.

3.4 Conclusion

In this chapter, the specification of required network is calculated related to the given HH model. The scaled neuron network with 1000 simulated neuron needs a 7.6Gbps peak bandwitdh. A brief analysis of the bandwidth and throughput is discussed according to three common topology, mesh, tree and fat tree and three communication strategy, unicast, multicast and boardcast. The result shows that both for the Hopfield and RNDC connection, multicast in mesh topology is an efficient solution.

4 Implementation Approach

The design of interconnect architecture and simulator is in the SystemC language. The Serdes interface is written in VHDL. The aim of this section is to introduce the design philosophy used to implement the multi-FPGA interconnection structure for the existing HH model FPGA implementation.

4.1 General Model Description

The network is in mesh topology which is generate on the Jan's module, shown in Figure 4.1. Each top router has a dual port connecting to the neighbour router for full duplex communication. The structure is shown in Figure 4.2.

4.2 Routing

In the simulator, unicast and multicast routing are both supported with different allocator structure. The point-to-point unicast routing algorithm uses the dimension

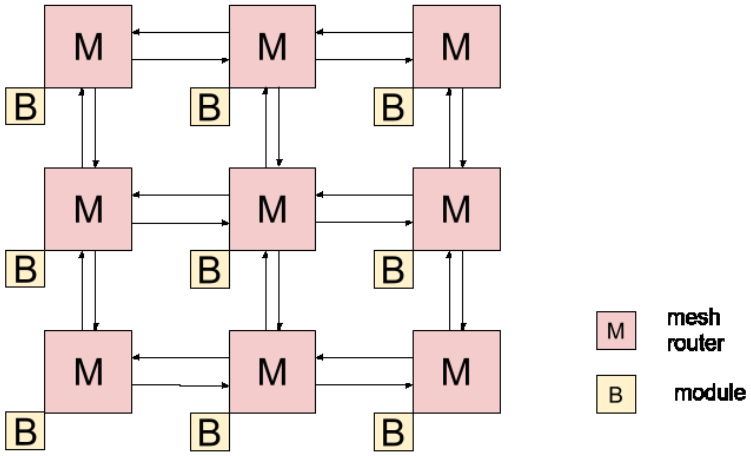

Figure 4.1: Example of a configuration of 3×3 mesh.

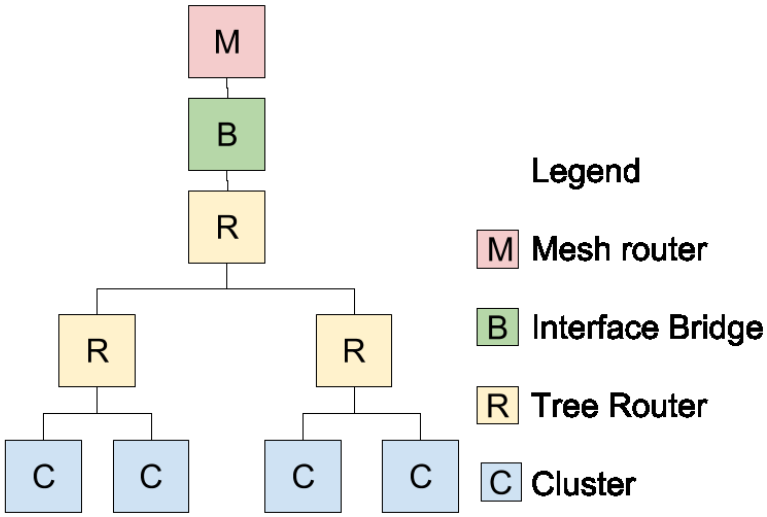

Figure 4.2: Example of a configuration of 4 clusters.

order routing algorithm by the computing at the input. The multicast routing algorithm we used is depicted in Algorithm 1 is tree based on the dimension order routing as mentioned in [19]. The routing path are the collection of all the path in the dimension order for unicast with only one route path for each node. The multicast DOR routing is also deadlock free. The routing destinations are pre-computed and stored in the routing table which is checked at the input buffer.

```
Source address collection S;
Mapping Destination address collection D(S);
Route = φ
for each s in S do
    for each d in D(s) do
        path = DOR(s,d);
        for each link l in path do
            if l is not in Route then
                add(l,route);
            else
            end
        end
    end
end
return Route
```

Algorithm 1: Multicast route generation with DOR

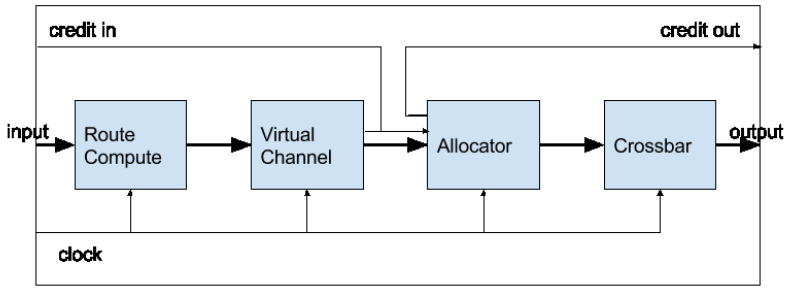

Figure 4.3: Block diagram of router structure.

4.3 Flow Control

There are two kinds of packets in the design, the initial packet which contains type, cell address, cluster address, parameter value and data value and the output packet which contains type, cell address and data value.

Both on/off signal and credit based flow control are supported in the simulator. For on/off signal back pressure the base line for the on/off of buffer need to be calculated. The latency between two routers is three router cycles which is consider in the lower boundary of the off signal generated under the vacancy input buffer size.

4.4 Router Structure

The router structure is based on the traditional one which is depicted in the previous chapter. This structure has mainly five parts input buffer, switch allocator, vitrual channel allocater, crossbar switch and processing unit interface. The block diagram is shown in Figure 4.3.

4.4.1 Virtual Channel

Virtual channels are used to buffer the packets which are contention for the same physical channel and increase the throughput. More virtual channels also need larger size of FIFOs and complex arbiter and allocator to route the packets.

4.4.2 Buffer

By far, the router architecture is in 5 port with an arbiter and a crossbar switch which is shown in Figure 2.10. Routing table is checked at the buffer write stage. A mesh topology connection is implemented with the XY routing algorithm. The flowchart for the arbiter is shown in Figure 4.4. Each input port is connected to a FIFO the size of which has less influence on the packet delay comparing to network load [12].

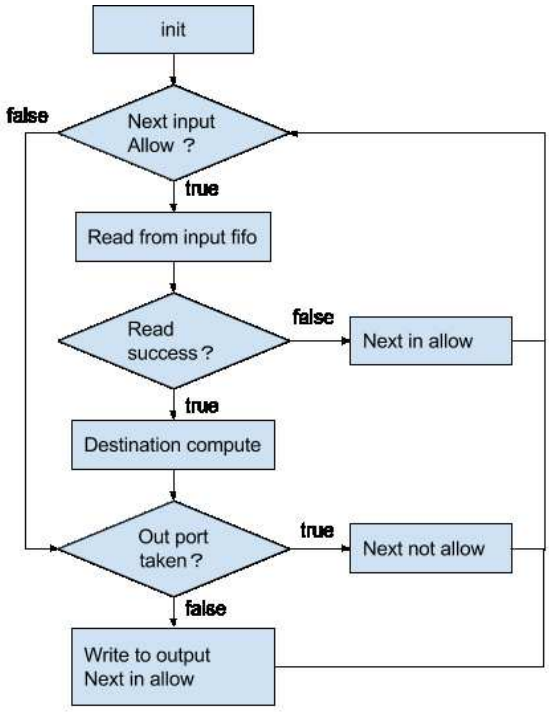

Figure 4.4: The flowchart of allocator

4.4.3 Credit Backpressure

The backpressure signal can be transfer within separate physical channel or combined with flits. However, there is trade-off between transfer latency and port resource. For separate channel, the credit can be transfer within one cycle latency in parallel. But it will cost 4 cycles to transfer multiplexing channel with data by serial. The additional latency in the back pressure transmission decrease the throughput of the network or need more virtual channels to multiplex using the physical channels.

4.4.4 Channel Allocator

Channel allocator is based on round-robin sequence. Each head flit is asked for one output channel and the body or tail flit keep the same virtual channel as the head.

4.4.5 Switch Allocator

The switch allocator allocate the output port for the flits which has taken the virtual channel. The allocator also use round-robin stretegy to arbitration when the contention occur in one output.

4.4.6 Crossbar Switch

The crossbar switch is the structure for the dataflow from input to output. It is controlled by the allocation decision.

4.4.7 Packet Model

There are different methods of connecting components in a system. Network on Chip (NoC) is one interconnection approach that promises higher flexibility and performance than classical interconnect approaches. The NoC network could get the most usage of the interconnection for the concurrently processing neurons. The signals contained in each of packet are illustrated in the following Table 4.1. The design in this thesis has very specific characteristics. As the message is only concluded of address and axon or dendrite potential data which are in the same format and width, the packets are in the same data width. To make the network suitable for the pseudo-random connection matrix, the destination is determined by lookup in the routing table generated at the initial section. The data in the packet is the voltage for axon or dendrite which has 32 bit width for float type and 64 bit width for double type. The value of type indicate which type the data stand for. *adr* is source address of the packet for checking lookup table. *dir* is specified in the multicast condition for output identify. *h_t* signal has three values for flit flow control. 01 stands for head, 11 stands for body flit and 10 for the tail flit. *vc_num* is for virtual channel distrubution during tranfer which can be allocated to different virtual channel at the input port. *tmp_vc* is used in the arbitration and allocation stage for multicast routing where a packet from input can set to multiple output port each cycle.

Table 4.1: data and signal contained in packet model

name	width
data	32/64
type	2
address	logN
direction	5
flit type	2
virtual channel No.	2

4.4.8 Router-node Interface

The interface between router and cluster bridge is implemented by three channels *cluster_in* , *cluster_init* and *cluster_out*. This is expend from the Jan's previous work[3]. The packets transfer within the routers are in unique form. The in/out packets and init packet are integrated to one type of packets and distinct at the output to inter the node.

4.5 Simulator

The interconnection network is a design under test in the simulator. A source and a sink are connected at the interface port of each node of the network to be test.

4.5.1 Test Traffic

At the source node, the test pattern is generated and push into a infinite source queue which is regard as a FIFO. The output is connect to the network input buffer and is controlled by the signal which could indicate the buffer state. When the buffer is vacancy, one test data is push out of the queue and injected into the network at each clock cycle. The test traffic can be set as uniform distribution, one direction or fully connected which is the worst case traffic the network tested.

The uniform distribution traffic can be regard as an average conditon for the traffic model which is a test condition in other simulators[2]. In the uniform distribution of test traffic, the destination of the data is generated randomly in the uniform distribution at each cycle. The bi-section bandwidth influence the throughput of this uniform condition where half of the traffic cross the bisection.

One direction test is specified for the axon output condition. In the model system[3], the output for axon is the only received at the one port and no need to transfer to the other node. The data flow of this condition purely unbalance and different with the others. The channel usage is only for the direction to the output and the bottleneck is focus on it. The traffic throughput is determined by the bandwidth of the interface to the outside.

The fully connected traffic is the most contention can occur in the total condition. This traffic pattern can occur in the Hopfield network where one neuron connected to the others.

4.5.2 Infinite Source Queue

The packets are generated at the source node according to packet length, a specified traffic pattern. An infinite depth queue connects the source generator and each node of the network. These source queues are not included in the network being tested, but implement to isolate the traffic processes from the network itself[5]. An input packet measurement process is facilitate before the packet injected which counts injected packets and measures the start time of each packet.

4.5.3 Receiver

The data are received at the sink and stamped with the end time of the transfer. The connection within network and receiver is unlimited bandwidth which transfer the packet with no latency and contention.

4.5.4 Data for simulator

To make a simulation of network, packets are injection into the tested structure and record the data listed blowTable 4.2. The start_time and end_time are stamp at the

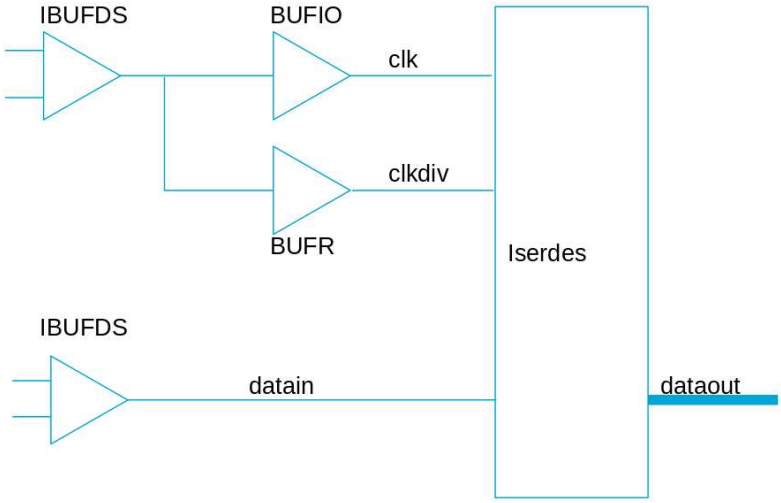

Figure 4.5: Schematic of iserdes.

source and sink module when injecting and pushing out of the network. The difference between them indicates the latency of eack packet tranfer. The distance between source and destination of each packet is recorded as hop_count in the model which has the trade-off with node degree when choosing the topology. The number of packets received by each node is also recorded which statistic for the traffic distribution for the routers.

Table 4.2: data contained in packet model for simulation

name	type
start_time	sc_time
end_time	sc_time
hop_count	int

4.6 FPGA serdes interface

The interface implemented in this thesis is using the source synchronous serdes connection. The schematic of interface is shown in the Figure 4.5 for input serdes and Figure 4.6 for output serdes.

The differential data interface mainly consists of two parts: data receiving module and data sending module. The data receiving module first converts differential signal of the data, the clock and the frame signal to a single end signal using the IBUFDS primitive. Then using the converted single-end clock for the clock input to the ISERDES primitive using BUFIO and BUFR and generate parallel clock with frequency division. The ISERDES primitive transfer serial data to parallel.

The serial data is processed by serial and conversion through the ISERDES module. When the ISERDESE2 interface type is NETWORKING, the latency is two CLKDIV

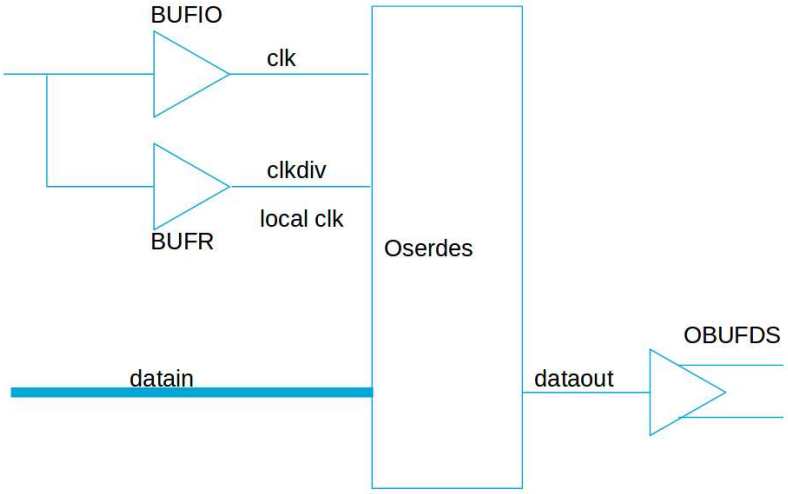

Figure 4.6: Schematic of oserdes.

cycles.

At the output port, first serialize the parallel data by OSERDES module and transfer the single end signal to differential signals to send. Latency is defined as a period of time between the following two events: (a) when the rising edge of local clock the data at inputs into the OSERDESE2, and (b) when the first bit of the serial stream appears at serial output. The latency values are shown in Table 4.3 [24].

Table 4.3: OSERDES latency values[24]

data$_rate$	data$_width$	latency
SDR	2:1	1 CLK cycle
	3:1	2 CLK cycle
	4:1	3 CLK cycle
	5:1	4 CLK cycle
	6:1	5 CLK cycle
	7:1	6 CLK cycle
	8:1	7 CLK cycle
DDR	4:1	2 CLK cycle
	6:1	3 CLK cycle
	8:1	4 CLK cycle
	10:1	5 CLK cycle
	14:1	5 CLK cycle

Because the input divided clock and local clock are not the same source, the input data is isolated by the FIFO which has the write clock of deserialized input clock and read clock of the local.

The acceptable bandwidth for the LVDS recevier is 1250Mb/s [23]. To achieve the 7.5Gb/s data rate it should have at least 6 LVDS ports per channel. Considering the

source synchoronous interface, another port is specified for the source clock.

4.7 Overhead

There are two level of overhead, the port overhead and frame overhead. In the interface level, the additional links are established for clock forward and credit back pressure for the buffers. As calculated in the previous, the 6 LVDS ports for data transfer one LVDS for source synchronous clock. The overhead is one over seven. The number of credit backpressure signal is equal to the number of virtual channels Nv which has the width of $logNv$. The additional signal is backpressure signal, the number of virtual channel. The total frame is containing data, source address and type of data. The overhead is calculated by $(Nv + logNv)/(Nv + logNv + data + adr + type)$. For the virtual channel number of 4, the overhead of frame is 6/85, nearly 7%.

4.8 Conclusion

The chapter describes the implementation approach for the router structure and the synchronous serdes interface in the Xilinx FPGA. The simulator has different traffic pattern to test the network performance.

5 Performance Analysis

5.1 Measures of Interconnection Network Performance

5.1.1 Throughput

Throughput is measured by counting the packets arriving at each output. For uniform traffic, we know on average that half of the traffic, N/2 packets, must cross the Bc bisection channels. The best throughput occurs when these packets are distributed evenly across the bisection channels. The ideal throughput is defined as the input bandwidth when the bottleneck channel is saturated [5].

5.1.2 Latency

Latency is measured by subtracting the start time from the finish time for each packet. The saturation is observed by the packet transfer latency. When it has a large latency, it can be regarded as being in the saturation. The ideal latency is dominated by the dimension. The average minimum distance hopcount H is [11]

$$H = \begin{cases} 2k/3 & k \quad is \quad even, \\ 2k/3 - 2/3k & k \quad is \quad odd. \end{cases} \tag{5.1}$$

where k is the size of the array.

The latency can be separated to the router latency T_r and interface latency T_i which are 3 and 4 cycles respectively in the design. The ideal latency which can also be regarded to zero-load latency is $T_r * H + T_i * (H - 1)$

5.2 Experiment Setting

The performance is tested with two kinds of traffic pattern, random uniform traffic and complete connection traffic. The uniform traffic is tested for the average performance which the network occurs where the destination of packets are randomly generated. The complete connection traffic pattern is that every packet need to transfer to the all other nodes which is the worst case the network can face.

5.3 Performance Simulation

The performance of the network is analyzed in the uniform traffic distribution. To find the maximum throughput, the injection rate is changed under the configuration of 4 virtual channels per input port. The configuration is shown in Table 5.1. The result is shown in the Figure 5.1. As the injection rate goes up, the throughput increases until

the injection gets to 45%. The latency has a burst increment when the injection rate is greater than 45 which means the network get to the saturation.

The number of virtual channels has big influence on the total performance which is show in the Figure 5.2. The configuration is shown in Table 5.1. The maximum throughput increase from 12 to 26 flits per cycle with the increment of virtual channels. When the number of virtual channel is greater than 4, the improvement of performance is not obvious. The relation of flit size with throughput is shown in Figure 5.3. It doesn't have obvious influence on the throughput. However when the flit size is 1, the throughput decreased, because the credit back pressure needs one cycle latency of propagate which introduce a cycle idle in the router pipeline.

Table 5.1: configuration of network with test of injection rate

communication mode	unicast
number of nodes	16
number of virtual channels	4
traffic pattern	uniform

Table 5.2: configuration of network with test of number of virtual channels

communication mode	unicast
number of nodes	16
injection rate	100
traffic pattern	uniform

The injection rate is calculated as $packets/total\_cycles$. The input packets are in the completed connected condition, which means every simulated neuron in one FPGA generate data to the all others. This reflects the worse case of the connection and the maximum throughput need to achieve.

The performance is also tested with all-to-all connection pattern for the worst case test. The whole transfer latency is recorded which is the interval of the first packet being injected to the queue and the last packet being transferred to the destination node.

For unicast and multicast routing, the test result of total transfer cycles is shown in Figure 5.4 and average node throughput is shown in Figure 5.5. The traffic distribution is shown in Figure 5.6 and Figure 5.7.

In the dimension of 2 by 2 or 3 by 3, the unicast routing and multicast routing have almost the same performance to transfer all the injected packets. When the size increases, the throughput of unicast routing decrease dramatically from 0.96 to 0.53. Multicast is more efficient in the large size of network. In the size of 5 by 5 network for unicast and multicast routing, the total cycles of transferring is 4527 compare to 2913 which is 33 percent better performance. For the multicast routing, which is more efficient for transfer same packets to different destinations, the simulation result for

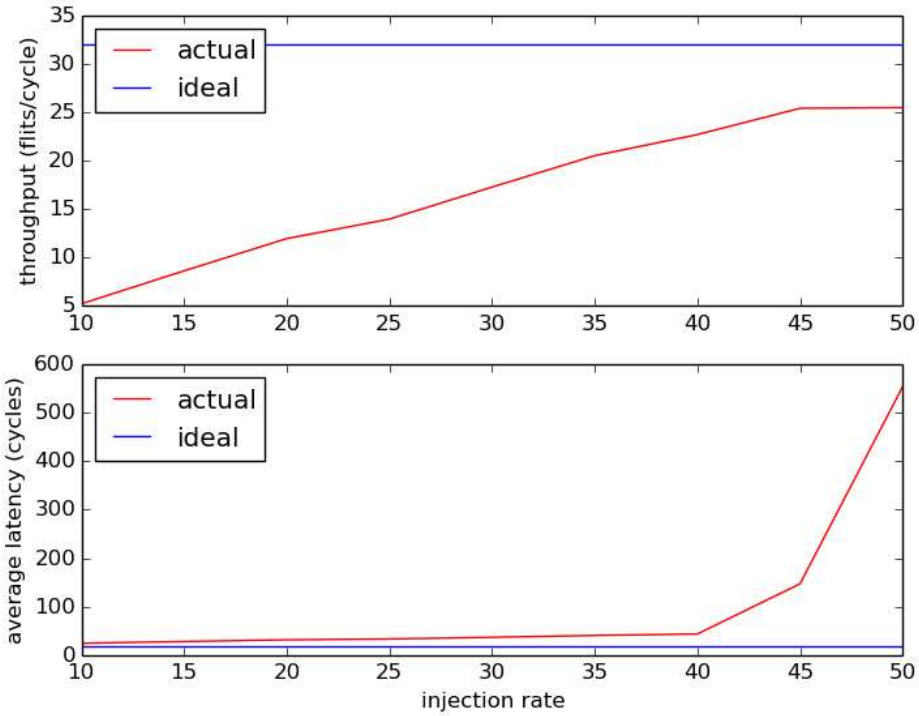

Figure 5.1: The throughput and latency with different injection rate

fully connection communication is shown in Figure **??**. The latency has liner relation with the mesh dimension. The traffic distribution is shown in Figure 5.7. The test is for the all-to-all connected traffic in 4by4 mesh topology.

5.4 Traffic Simulation Under Jan's Configuration

As depicted in the previous chapter, the neuron model has the neighbor connection of 8. Which means, each neuron is connected to the other 8 neuron next to it in the mesh topology. This neuron connection topology is suitable for the mesh interconnection.

We analyzed the effect of transfer latency between routers on the total transmission latency. The simulated network has total of 400 neurons and implemented in 16 PhyCs which are equally separated with four FPGAS. Each FPGA is configured in 2 clusters with 2 PhyCs and has 25 slices which is 100 simulated neurons. The FPGAs models are set in mesh topology with the dimension order routing algorithm. The number of packets pass through each router at the scale of 2×2, 3×3 and 4×4 is shown in Figure 5.8, Figure 5.9 and Figure 5.10, respectively. The result shows unequal distribution which is caused by the output data need to be transferred to the only port communicated with computer. This port is set at the router 1 in this assumption. The most packets are the data to the output not the dendrite data communicate to the other neurons. Figure 5.11 shows the number of packets of dendrite which is communicate

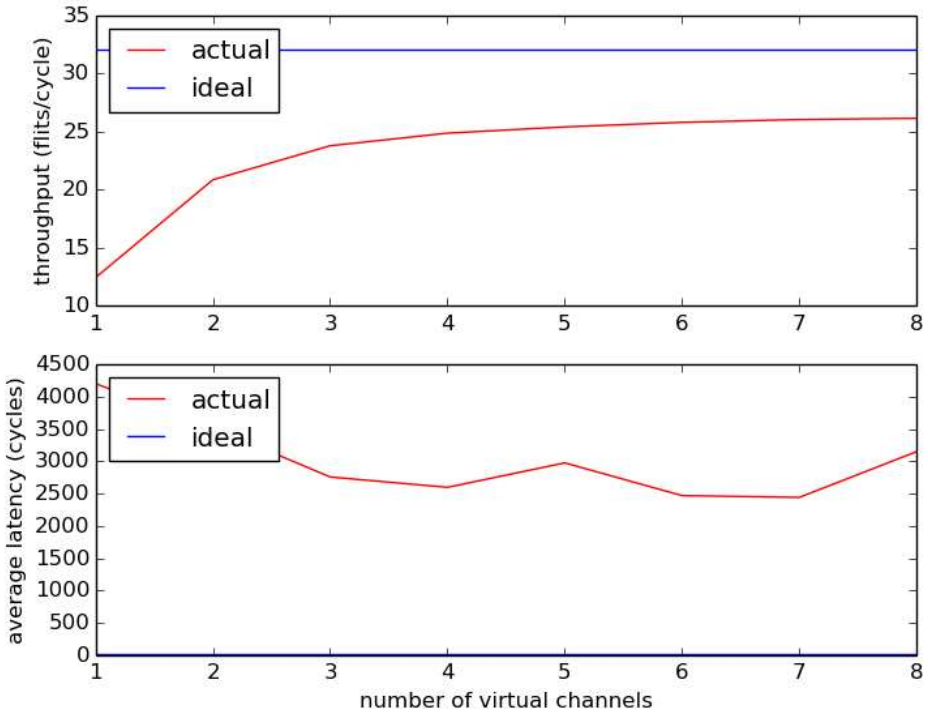

Figure 5.2: The throughput and latency with different number of virtual channels per input

with other neurons for neighboring ION potential. The axon packets are shown in Figure 5.12.

The results are shown in Figure 5.13 and Figure 5.14. As the latency is increased for 4 cycles to 16 cycles, the maximum latency is increased from 44 to 3064. The distribution of the 4 cycles condition shows linear characteristic which has the mean of 25. But for the 16 cycles condition, the most is around 200 which is 8 times of the previous. The average latency is 252 which means the congestion occurs heavier because of the lower speed. The latency increased by the number of routers, which is shown in Figure 5.15 and Figure 5.16. The average transfer latency are 35 and 422 for the size of 9 FPGAs and 16 FPGAs. The maximum latency is 100 and 3124 respectively.

To reduce the traffic caused by the axon data, two network are used for axon and dendrite separately. The network for dendrite is built in mesh topology with the double direction channel. The network for axon is built in tree topology with one direction connect. The routers are set to only transfer the internal packets for dendrite which are shown in Figure 5.17 and Figure 5.11. The maximum packets transferred per router is reduced to 438 and the latency is 18 with a linearly distribution.

With a different connection of SimC, a 3-d connection, and 12 neighbours per each SimC, the router traffic increases and average latency increases from 25 to 60 with the maximum 44 to 294

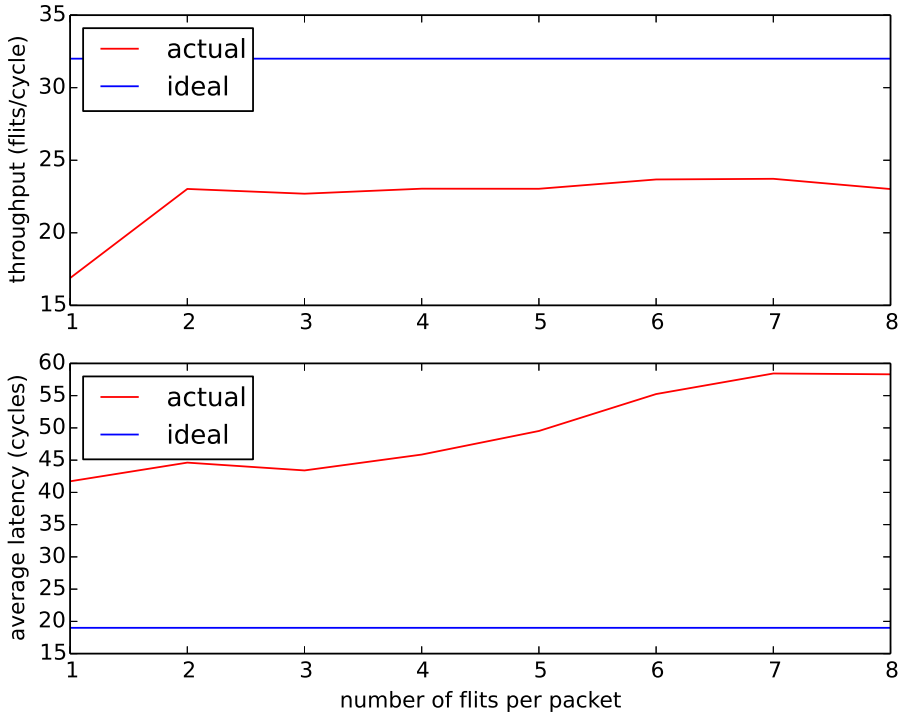

Figure 5.3: The throughput and latency with different flit size

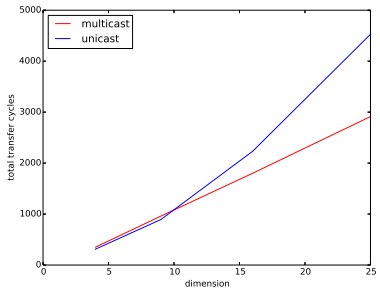

Figure 5.4: total transfer cycles

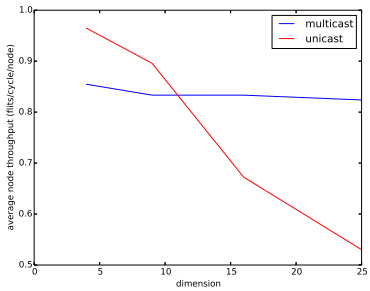

Figure 5.5: average node throughput

5.5 Serdes Interface

The interface behavioural is simulated for iserdes and oserdes respectively. The iserdes behaviour testbench result is shown in Figure 5.19. The oserdes behavioural testbench result is shown in Figure 5.18. The simulation for the whole interface which connect oserdes to iserdes is shown in Figure 5.20. The total latency from oserdes to iserdes is 4 local cycles maximum which is depend on the alignment of deserialization in the 10 to 1 serdes connection.

The result of bitslip control for signal alignment is shown in Figure 5.21. With one

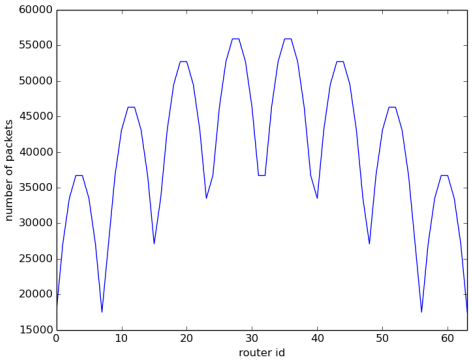

Figure 5.6: The unicast traffic distribution

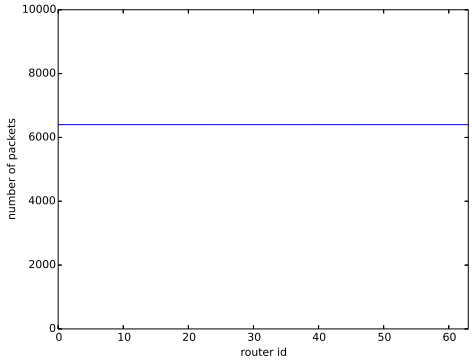

Figure 5.7: The multicast traffic distribution

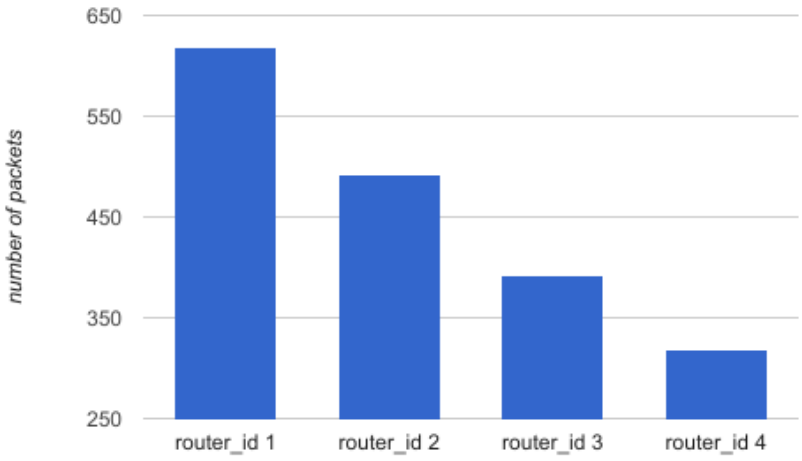

Figure 5.8: The number of packets pass through each router within every simulation step

cycle of inserted signal, the serdes output shift one bit.

5.6 Performance Simulation With Serdes Latency

5.6.1 Influence of the Interface Latency

Considering the 4 cycle serdes latency, the throughput is decreased because of the idle cycle generated during the transferring. The test pattern is set to transfer all packets generated for the all-to-all connection. The injection rate is set to 100 which means every cycle a packet is injected into the input queue.

The comparison result of total cycles to transfer the test pattern is shown in Fig-

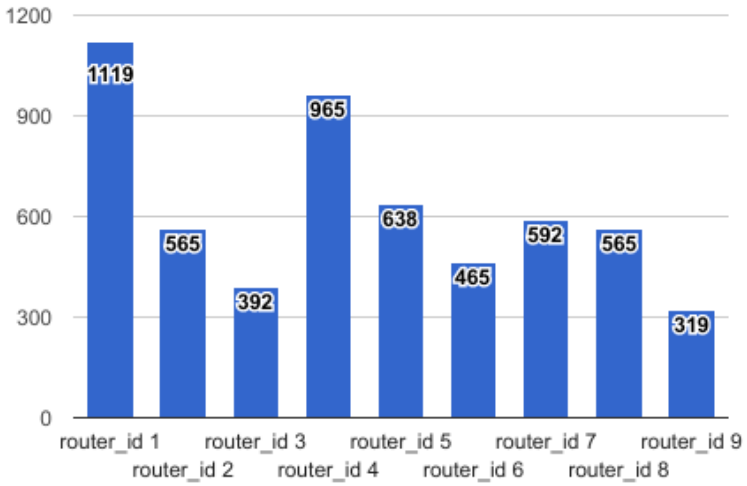

Figure 5.9: The number of packets pass through each router within every simulation step

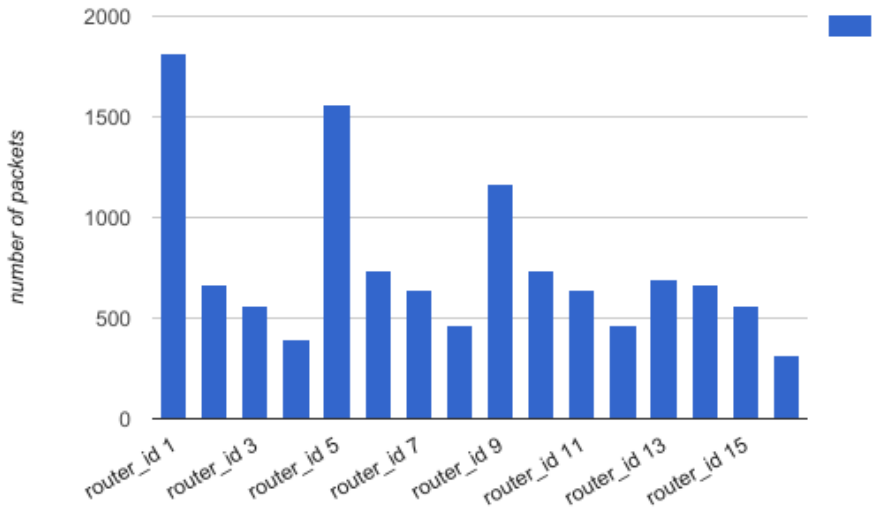

Figure 5.10: The number of packets pass through each router within every simulation step

ure 5.22. The total cycles that needed to receive all packets at the output of each node is recorded. With different size of dimension, the total transfer cycles goes linearly from 513 (2×2 mesh) to 4379 (5×5 mesh) with the interface latency and from 351 (2×2 mesh) to 2913 (5×5 mesh) without interface latency. The average transmission latency

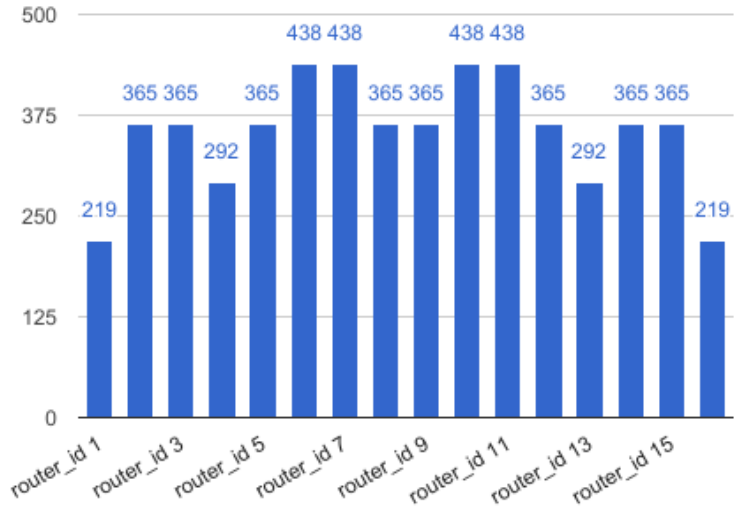

Figure 5.11: 16 FPGA routers' traffic

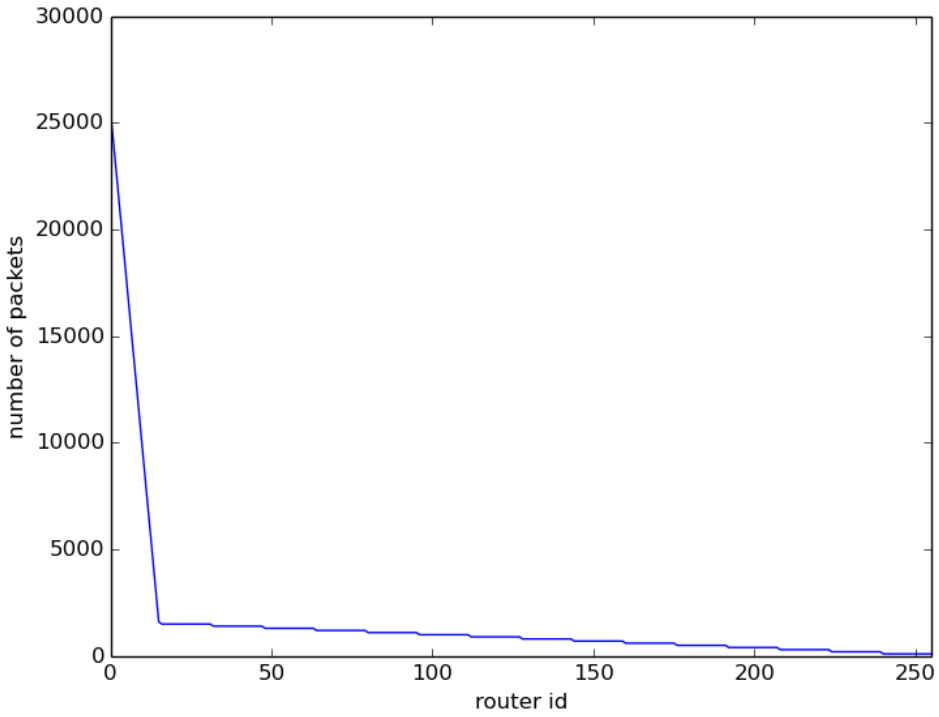

Figure 5.12: The distribution of axon packets

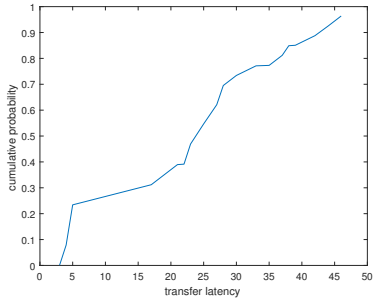

Figure 5.13: 4 cycles latency per hop

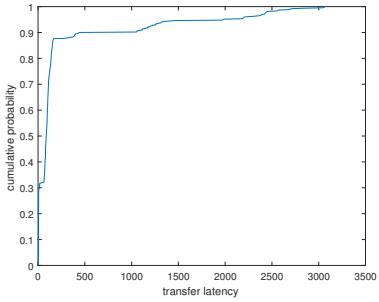

Figure 5.14: 16 cycles latency per hop

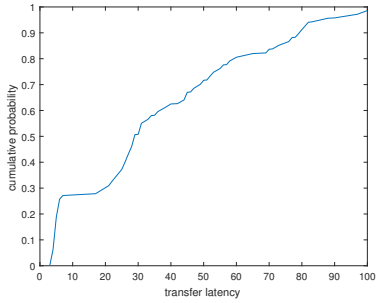

Figure 5.15: 9 FPGA routers

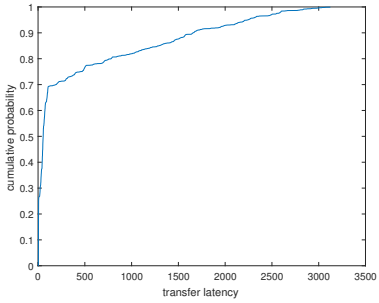

Figure 5.16: 16 FPGA routers

of each packet is shown in Figure 5.23. With the dimension increased, there is a liner relation with the latency. With different size of dimension, the average latency goes linearly from 209.03 (2×2 mesh) to 2035.36 (5×5 mesh) with the interface latency and from 125.94 (2×2 mesh) to 1390.02 (5×5 mesh) without interface latency. The average throughput is for each node is shown in the Figure 5.24. With the dimension increased more contention occurred, consequently the average throughput decreases. For the no interface latency condition, the average node throughput keeps around 0.85 to 0.8 flits per cycle per node. When the interface latency is 4, the average throughput drops to around 0.6 to 0.55. The result shows that which the same router resource the interface latency has negative influence and decrease the 30 percent on throughput performance

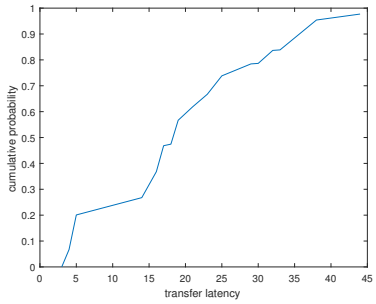

Figure 5.17: 16 FPGA routers without axon

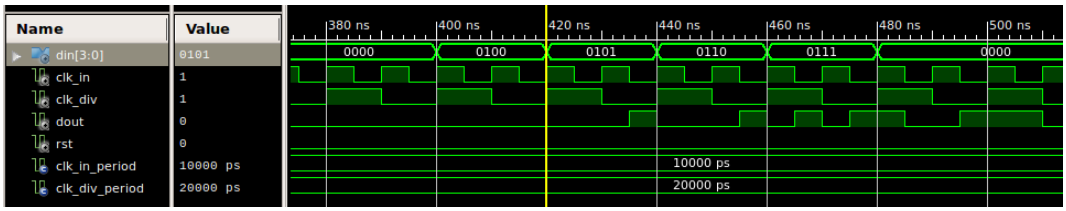

Figure 5.18: The simulation of oserdes

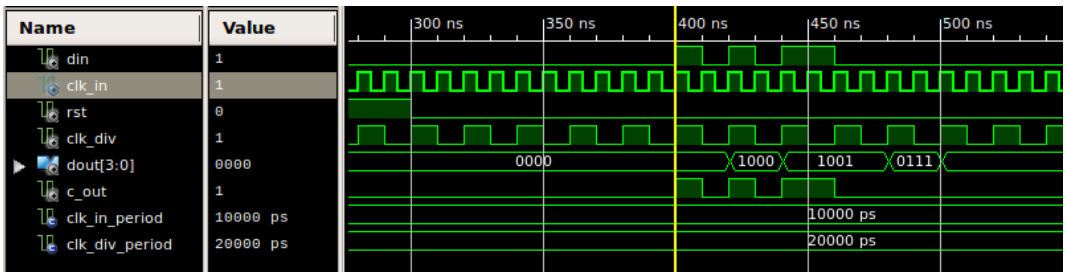

Figure 5.19: The simulation of iserdes

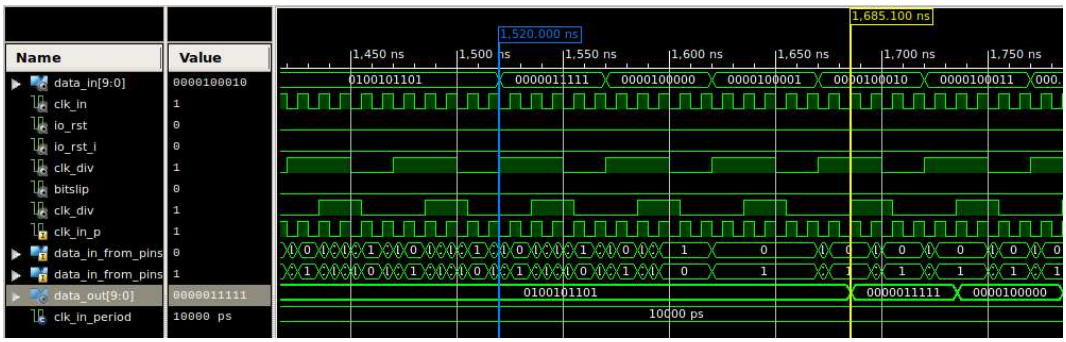

Figure 5.20: The simulation of iserdes connect to oserdes

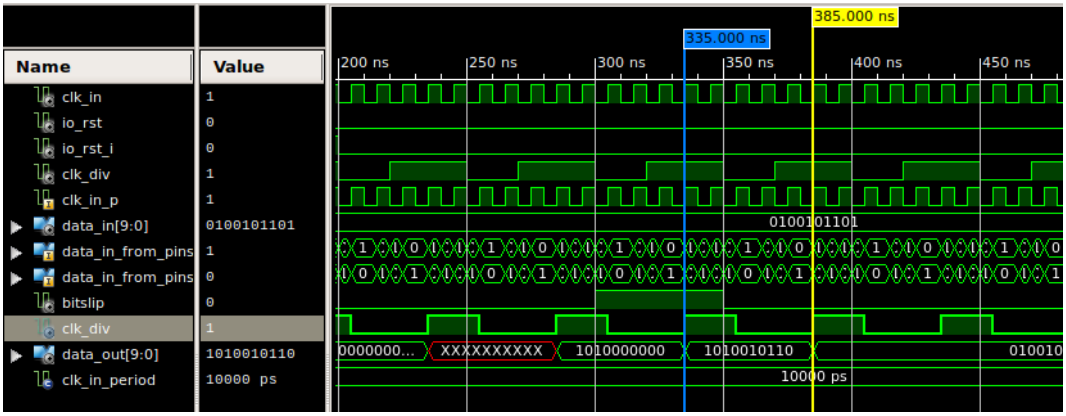

Figure 5.21: The simulation of bitslip

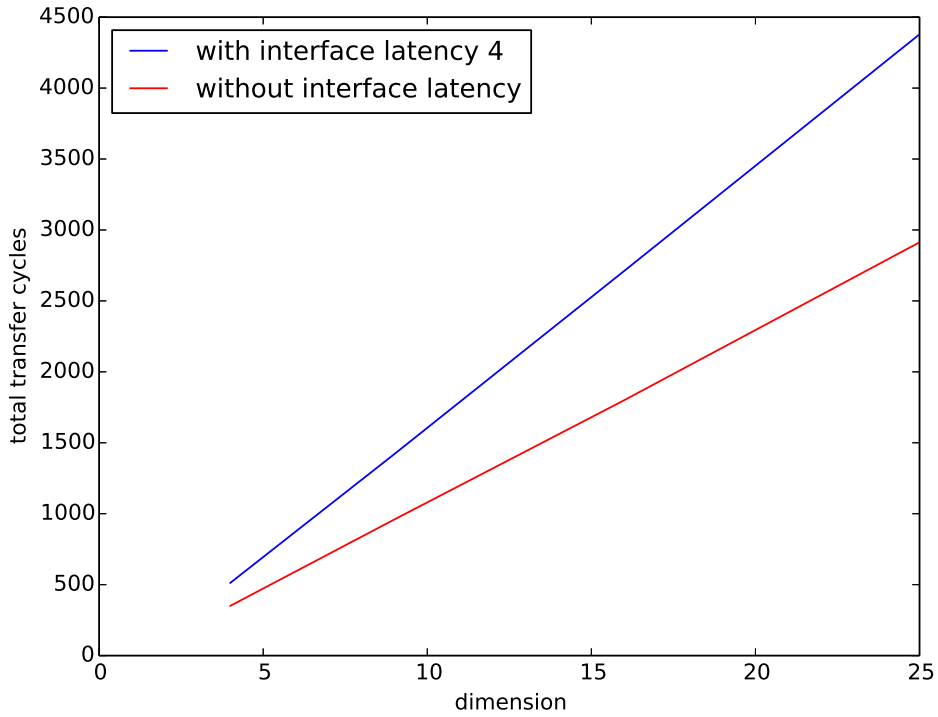

Figure 5.22: The simulation with interface latency

and increase 46 percent of total transfer cycles.

5.6.2 Virtual Channel

To decrease the additional idle generated by the interface latency, more virtual channels can be contained to fully use physical channel. The test is set with the variation of virtual channels of each input port. The test pattern is the all-to-all connection. The injection rate is set to 100 to test for the maximum throughput.

The test result of total cycles and improvement rate are shown in Figure 5.25. The improvement rate is calculated by the decreased cycles over the increasing number of virtual channels. With different number of virtual channels (VC), the total transfer cycles is dramatically reduced from 5631 (2 VC) to 1638 (10 VC) which is shown in blue line. The improvement (in red line) is obvious in the small number of VC but not so much when the channels are enough. The improvement gets to 1.1 from the 2 VC to 4 VC, but from 8 VC to 10 VC only 0.07 improvement in total transfer cycles. Figure 5.26 shows the average transfer latency in the variation of VCs. With the number of VC increased, there is decrease in the latency which is from 2619 (2 VC) to 168 (10 VC). The average throughput is for each node is shown in the Figure 5.27. The throughput per node goes up linearly from 2 VC to 6 VC. With further incresement of VC the improvement rate slow down. The result shows that with the more virtual channel resources, the performance improves. The improvement is obvious for the less

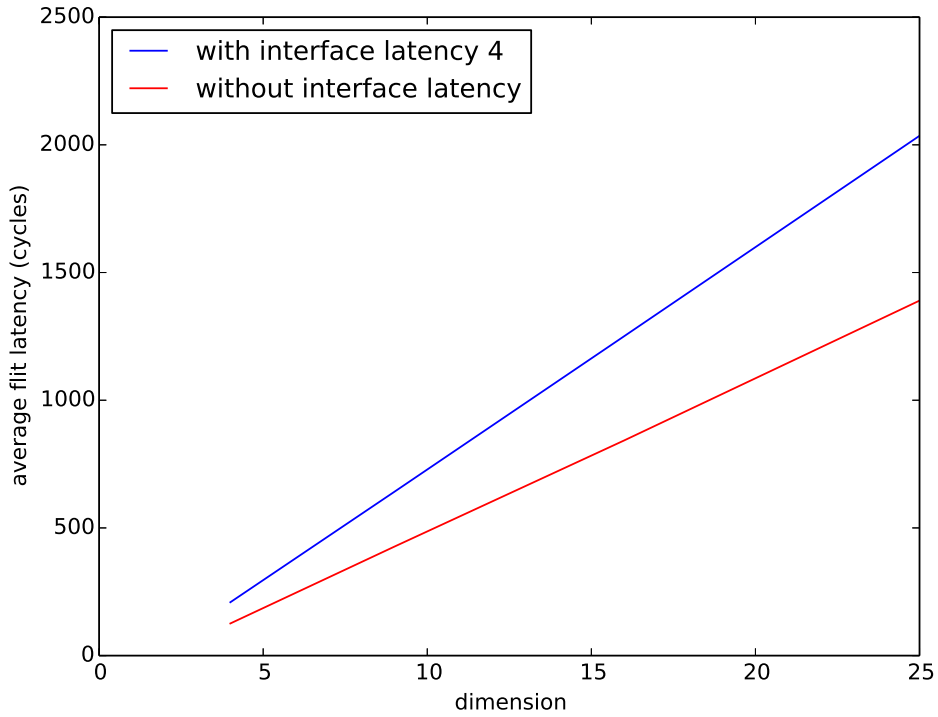

Figure 5.23: The average latency

resources.

5.7 Conclusion

In this chapter the performance are tested with different configuration of the network. The results show that the multicast communication has at most 33% improvement comparing with unicast communication. The latency has liner relation with the dimension which proofs the analyse in the chapter 4.

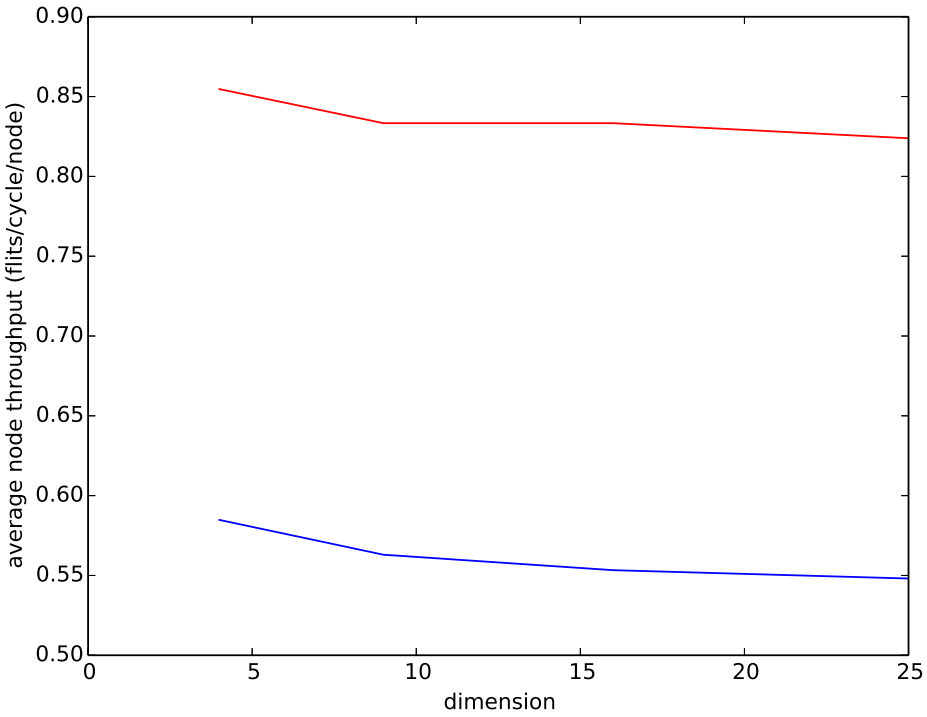

Figure 5.24: The average throughput

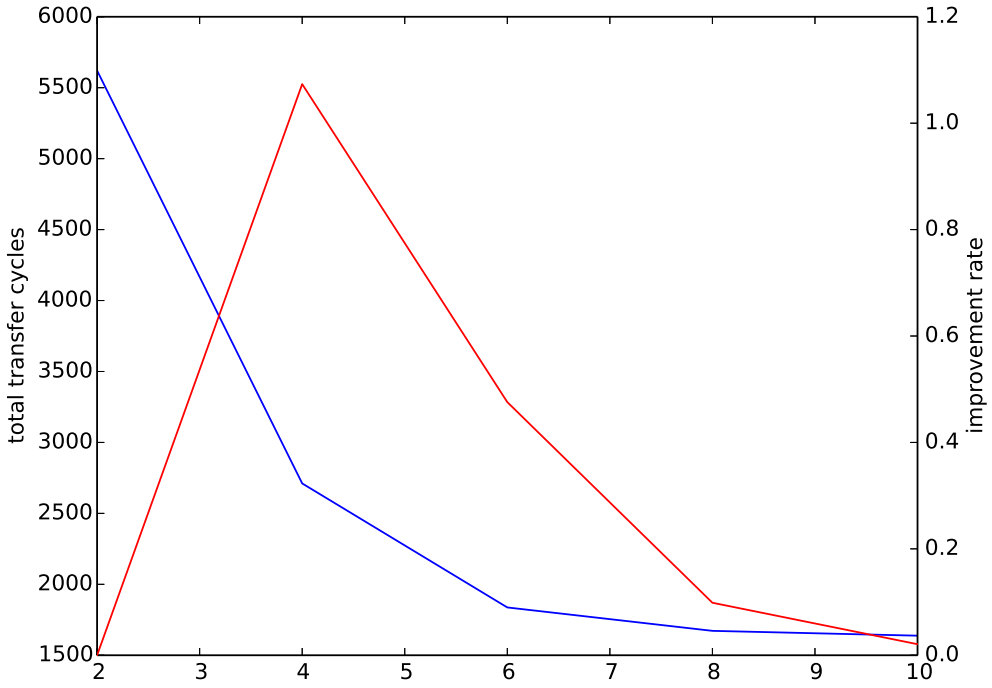

Figure 5.25: The simulation with interface latency

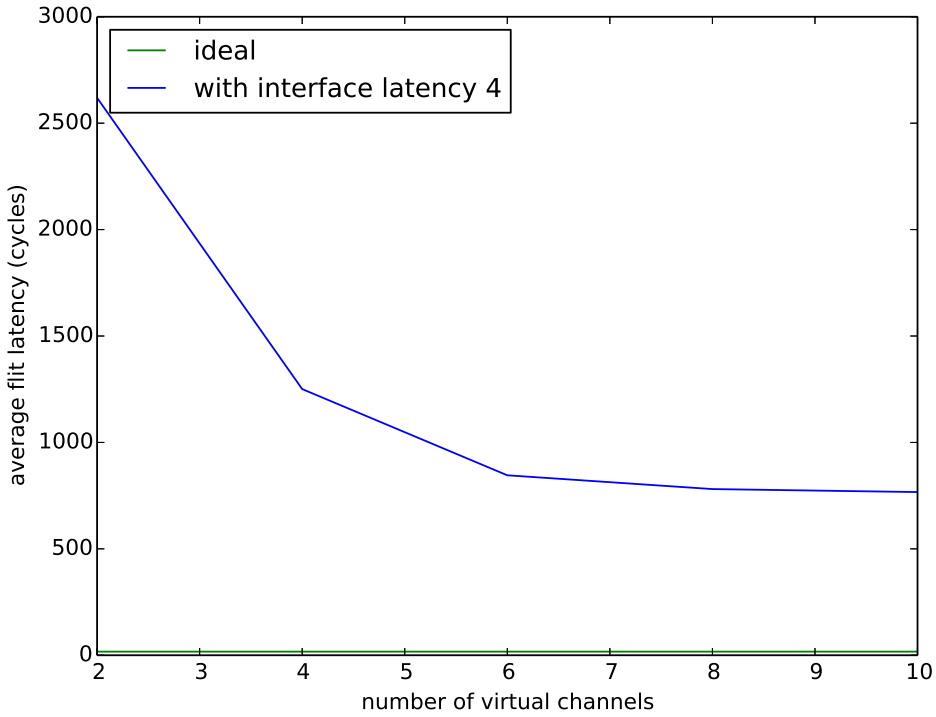

Figure 5.26: The average latency

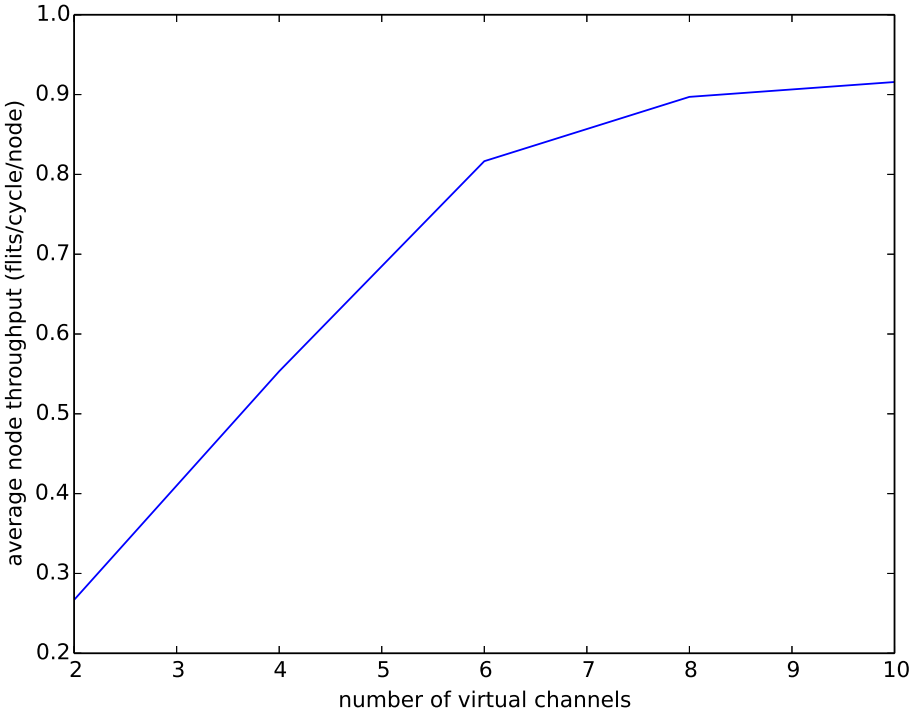

Figure 5.27: The average throughput

6 Conclusion and Future Work

6.1 Conclusion

In thesis, we implement a cycle accurate interconnect network simulator platform to mimic the packet transfer with the different nodes of neuron network generated by Jan[3]. The simulator can be configured with different size, flits number and virtual channels. Different test patterns are required to test the network within different condition.

To find an efficient solution, the simulator compares unicast and multicast routing and the result shows that multicast routing has linear time complexity with the increase of the number of nodes and has most 33% performance improvement in the mesh size of 5×5. The number of 4 virtual channels have most efficiency among others.

6.2 Future Work

This section provides ideas for further research interests.

Synthesize and implement on FPGA: The system is work in the cycle accurate SystemC but need to transmit to synthesizable SystemC to implement on FPGAs.

Other topology: Different topology can be implemented such as torus, small world. The network topology has obvious influence on the total performance. Other topology which is specified for the neural network connection could also be tested which may have better performance.

Deadlock mechanism The difficulty is to deal with the deadlock which occurs at the ring dependency of resource. The deadlock control mechanism can be discussed to get the better service quality. If the recovery mechanisms is included, more routing algorithms can be implemented.

Bibliography

[1] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny. Cost considerations in network on chip. *INTEGRATION, the VLSI journal*, 38(1):19–42, 2004.

[2] V. Catania, A. Mineo, S. Monteleone, M. Palesi, and D. Patti. Noxim: An open, extensible and cycle-accurate network on chip simulator. In *Application-specific Systems, Architectures and Processors (ASAP), 2015 IEEE 26th International Conference on*, pages 162–163. IEEE, 2015.

[3] G. J. Christiaanse, A. Zjajo, C. Galuzzi, and R. van Leuken. A real-time hybrid neuron network for highly parallel cognitive systems. In *Engineering in Medicine and Biology Society (EMBC), 2016 IEEE 38th Annual International Conference of the*, pages 792–795. IEEE, 2016.

[4] W. J. Dally and C. L. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. 1988.

[5] W. J. Dally and B. P. Towles. *Principles and practices of interconnection networks*. Elsevier, 2004.

[6] J. R. De Gruijl, P. Bazzigaluppi, M. T. de Jeu, and C. I. De Zeeuw. Climbing fiber burst size and olivary sub-threshold oscillations in a network setting. *PLoS computational biology*, 8(12):e1002814, 2012.

[7] J. Duato. A necessary and sufficient condition for deadlock-free adaptive routing in wormhole networks. *IEEE Trans. Parallel Distrib. Syst.*, 6(10): 1055–1067, Oct. 1995. ISSN 1045-9219. doi: 10.1109/71.473515. URL http://dx.doi.org/10.1109/71.473515.

[8] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana. The spinnaker project. *Proceedings of the IEEE*, 102(5):652–665, 2014.

[9] G. Indiveri, F. Corradi, and N. Qiao. Neuromorphic architectures for spiking deep neural networks. In *Electron Devices Meeting (IEDM), 2015 IEEE International*, pages 4–2. IEEE, 2015.

[10] E. M. Izhikevich. Which model to use for cortical spiking neurons? *IEEE transactions on neural networks*, 15(5):1063–1070, 2004.

[11] N. E. Jerger and L.-S. Peh. On-chip networks. *Synthesis Lectures on Computer Architecture*, 4(1):1–141, 2009.

[12] S. Kumar, A. Jantsch, J.-P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, and A. Hemani. A network on chip architecture and design methodology. In *VLSI, 2002. Proceedings. IEEE Computer Society Annual Symposium on*, pages 117–124. IEEE, 2002.

[13] R. Legenstein and W. Maass. Edge of chaos and prediction of computational performance for neural circuit models. *Neural Networks*, 20(3):323–334, 2007.

[14] R. Libeskind-Hadas, D. Mazzoni, and R. Rajagopalan. Tree-based multicasting in wormhole-routed irregular topologies. In *Parallel Processing Symposium, 1998. IPPS/SPDP 1998. Proceedings of the First Merged International... and Symposium on Parallel and Distributed Processing 1998*, pages 244–249. IEEE, 1998.

[15] M. P. Malumbres, J. Duato, and J. Torrellas. An efficient implementation of tree-based multicast routing for distributed shared-memory multiprocessors. In *Parallel and Distributed Processing, 1996., Eighth IEEE Symposium on*, pages 186–189. IEEE, 1996.

[16] 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*, 61(3):820–833, 2014.

[17] 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, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014.

[18] J. Navaridas, M. Luján, J. Miguel-Alonso, L. A. Plana, and S. Furber. Understanding the interconnection network of spinnaker. In *Proceedings of the 23rd international conference on Supercomputing*, pages 286–295. ACM, 2009.

[19] J. Navaridas, M. Luján, L. A. Plana, S. Temple, and S. B. Furber. Spinnaker: Enhanced multicast routing. *Parallel Computing*, 45:49–66, 2015.

[20] E. Painkras, L. A. Plana, J. Garside, S. Temple, F. Galluppi, C. Patterson, D. R. Lester, A. D. Brown, and S. B. Furber. Spinnaker: A 1-w 18-core system-on-chip for massively-parallel neural network simulation. *IEEE Journal of Solid-State Circuits*, 48(8):1943–1953, 2013.

[21] J. Schemmel, D. Briiderle, A. Griibl, M. Hock, K. Meier, and S. Millner. A wafer-scale neuromorphic hardware system for large-scale neural modeling. In *Circuits and systems (ISCAS), proceedings of 2010 IEEE international symposium on*, pages 1947–1950. IEEE, 2010.

[22] D. Vainbrand and R. Ginosar. Network-on-chip architectures for neural networks. In *Networks-on-Chip (NOCS), 2010 Fourth ACM/IEEE International Symposium on*, pages 135–144. IEEE, 2010.

[23] Xilinx. *Artix-7 FPGAs Data Sheet: DC and Switching Characteristics*. .

[24] Xilinx. *7 Series FPGAs SelectIO Resource User Guide (UG471)*. .

[25] C. Zhang. *Research on Key Issues and Applications of Multi-FPGA Systems*. PhD thesis, Chongqing university, 2011.

Code Examples

A

A.1 packet model

```
1  class packet{

       sc_uint<3> init_type;
       mod_prec data;
       sc_uint<2> type;
6
       sc_uint<CALC_LOG2(NUMBER_OF_CELLS)> adr;
       sc_uint<CALC_LOG2(DEF_MAX( (NUMBER_OF_STATES-1) , (
           MAX_NUMBER_OF_CONNECTIONS)))> adr2;

       sc_uint<BIT_NUM_OF_PORTS> dir;
11     //direction for multicast
       sc_uint<BIT_NUM_OF_DEST> dest;
       //destination address
       sc_uint < CALC_LOG2 (NUMBER_OF_CLUSTERS - 1) > cluster_init_clus;
       //initialization
16     sc_uint<3> h_t;
       //identifier for header body or tail flit
       uint vc_num;
       //virtual channel number
       uint tmp_vc[NUMBER_OF_PORTS];
21     //temporary virtual channel number when arbitration for multicast
   #ifdef SIM_CYCLE
       sc_time start_time;
       //start time when packet is generated
       sc_time end_time;
26     //end time when packet gets to the destination
       int hop_count;
       //number of routers the packet transfered
   }
```