Design And Implementation of Congestion Aware Router for Network on Chip

PROJECT REPORT

Submitted by

MELVIN T BALAKRISHNAN

in partial fulfillment of the requirements

for the award of the degree of

MASTER OF TECHNOLOGY

Department Of Electrical Engineering

INDIAN INSTITUTE OF TECHNOLOGY MADRAS

JUNE 2021

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY MADRAS

2021

CERTIFICATE

This is to certify that this thesis (or project report) entitled ***"Design And Implementation of Congestion Aware Router for Network on Chip"*** submitted by **MELVIN T BALAKRISHNAN** to the Indian Institute of Technology Madras, for the award of the degree of **Masters of Technology** is a bona fide record of the research work done by him under my supervision. The contents of this thesis (or project report), in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma..

Dr. T. G. Venkatesh
Research Guide
Associate Professor
Department of Electrical Engineering
IIT Madras 600036

Acknowledgment

First and foremost, I would like to express my sincere and deepest gratitude to my guide, **Dr. T G Venkatesh**, Associate Professor, Department of Electrical Engineering, IIT Madras, for providing me an opportunity to work under him. I would like to express my deepest appreciation for his patience, valuable feedbacks, suggestions and motivations.

I convey my sincere gratitude to Vijaya Bhaskar, PhD Scholar, IIT Madras, for all his suggestions and support during the entire course of the project. Throughout the course of the project he offered immense help and provided valuable suggestions which helped me in completing this project.

I would like to extend my appreciation to my friends and family for their help and support in completing my project successfully.

Abstract

From mobile phones to satellites, every electronic device in modern world has an SoC inside them. SoC have multiple high performing IP Modules that need to be interconnected through a reliable, faster and energy efficient network. Networks on Chips (NoC) are the best solution for on chip interconnection challenges of the modern day integrated circuitry. NoC are reliable state of the art on chip packet based communication systems. These sophisticated interconnection networks can maintain low packet latency, high bandwidth, high throughput with minimum area, better energy efficiency and fault tolerance. In this thesis we design a router for NoC using Vivado HLS. These routers are later employed to develop a scalable square mesh NoC capable of doing simulations and estimating different performance metrics like latency, waiting time and total packets handled. The NoC also has provisions to alter other parameters like buffer depth, packet size, packet injection interval and traffic used. Further we propose a simple mechanism to detect local congestion within the network. This congestion metric is used to upgrade XY DOR into a minimal adaptive X/Y routing strategy with very low hardware overhead. The proposed routing method is compared against conventional XY DOR for different parameter variations on Mesh NoC. The results show that the proposed routing method can perform really well with any traffic pattern at medium packet injection rates where the network congestion is low. With transpose traffic peak latency reduction of 60% and 37% increase in packet acceptance is achieved against conventional XY DOR. For shuffle traffic packet acceptance increased by 11% while peak latency remained same and average latency dropped by 6%. Random traffic showed peak latency reduction of 14% against XY DOR.

Contents

List of Figures

List of Tables

ABBREVIATIONS

NoC	Network on Chip
SoC	System on Chip
HLS	High Level Synthesis
VC	Virtual Channel
HoL	Head of Line
DOR	Dimension Order Routing

Chapter 1

Introduction

The modern day integrated circuitry for high performance applications is an integration of many different resources such as CPU, DSP, Intellectual Property (IP) Cores and memory etc. into a single chip. These high performing chips are termed as multiprocessor System on Chip (MPSoC). Due to technology scaling the transistors have become much smaller that their operation is now faster and more power efficient. But the data flow inside the chip has been adversely affected due to smaller and narrower metal wires which has more resistance to the current flow. It is obvious that the processing capabilities and processing power can be increased with numerous high performing modules or Cores integrated in a tight space. But the performance or the speed of processing relies on the speed of inter communication between these modules.

Traditional bus based communications used for the single core chips is outdated simply because the same approach is not applicable for any modern day System on Chip (SoC). The time when computation was expensive and communication cheap is long gone and today we have reached a pinnacle where computation is faster but the movement of data is taking longer duration even inside a chip. So the CPU-Memory speed margin is no more the only concern in modern VLSI industry.

In effect integrating multiple modules into a single chip is easy and vastly employed these days but to squeeze out the maximum performance from the hardware already integrated, an efficient data flow network is necessary inside the chip. With future designs expected to integrate hundreds of processing cores on a single chip, on-chip communication is thus expected to have a significant impact on chip-level performance and energy efficiency. Better alternatives to feed data in and out of each cores have led to the development and implementation of Network on Chips (NoC).

1.1 Network on Chip (NoC)

In the era of parallel computation which is the soul of modern day SoC, a faster, reliable and efficient data flow network within the silicon is necessary. One way of interconnecting the different modules is by using common buses. But such an architecture is not scalable enough for the multi core architecture within modern day SoC. Network on Chip (NoC) is the viable alternative which is reliable, faster and efficient paradigm for on chip communication[1]. With more nodes in the system, metrics like communication bandwidth, memory bandwidth, processing throughput etc. of the MPSoC increases. Thus, router based networks are the favorable on chip interconnection architecture for the modern multi-core processors[2].

NoC consists of a set of nodes that will be interconnected to other nodes in the network. These interconnection pattern give rise to the topology of the network. Every node in a topology has its own processor, local memory, and interface modules[3]. The functionality of the nodes differ based on the application of the chip like vector processors, graphics processors, DSP processors, I/O processors etc. Intellectual Property (IP) Cores is the generic term used for these components.

Essentially NoCs are router based On Chip communication networks that use packets for data transfer. Each packet is a group of smaller units called flits[4]. The basic building blocks of NoC are routers, IP cores and network interface. Routers

at every node are connected to the neighbour node via on-chip local wiring called links. Thus the design of NoC comprises of the selection of a topology, routing algorithm, flow control mechanism, router architecture and link specifications[5]. The architecture of NoC is explained in the subsequent sections.

1.2 Topology

Topology refers to the physical layout and interconnection scheme between the nodes in a network. There are various topologies in literature like mesh, torus, binary tree, ring, butterfly etc. The performance of a network depends on the topology chosen[6]. For instance, the Figure 1.1 shows a mesh topology. The red squares are the routers

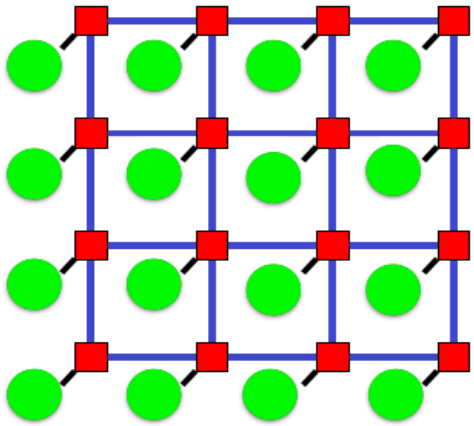

Figure 1.1: Mesh Topology

and the green circles represent the IP Core for each node. This is a 4x4 mesh NoC with 16 routers and IP cores. The router interconnection links are shown in blue lines. Links are the actual metal wiring in the network.

The transfer of a packet from one node to other is called a hop. A message is delivered from the source to the destination node by making several hops across the shared channels or links. Once a topology is selected, there could be different

possible paths (sequences of nodes and links) that a message could take through the network to reach its destination. This refers to the path diversity of a topology[4]. A topology with a high path diversity indicates the possibility of transmitting packets through multiple shortest paths. Path diversity gives the routing algorithm more flexibility to send a packet under congestion.

1.3 Router

The most important component within the node is the router. These are intelligent units capable of communication with other nodes. As a result, direct interconnection networks are also called router-based networks. Each router has interfaces to connect to other routers and the local IP Core. Links are used for interconnecting two routers. Usually, the links are unidirectional so a pair of such links are used for interconnecting two neighboring routers to ensure bidirectional communication. The dedicated routers enables overlapped computation and communication within the network releasing this pressure from the IP core[7].

Design of router is also equally important for an NoC. Routers are equipped with different units like FIFO Buffers, VC allocator, route computation unit, arbiter, crossbar switch, switch allocator etc.[8][9].

1.3.1 FIFO Buffers

FIFO buffers are used to store packets or more precisely the flits. The router can have buffer at either input or output or both sides. But these come at the cost of hardware overhead. The most commonly used architecture is the input buffered router where input side alone has the buffers[10].

The organization of these buffers can be performed using Virtual Channels (VC) which is the most common architecture of today[11]. In virtual channel based routers, multiple virtual-channels can be linked to one physical channel. The allo-

cation of VC is done independently by the Virtual Channel Allocator[12]. Because of this, virtual channels can isolate buffers from transmission units. This results in less network congestion and improves both throughput and latency. For adaptive routing, they could provide deadlock freedom by allowing blocked packets to be bypassed.

1.3.2 Switch Allocation

Switch Allocator is a shared resource within the router. It is possible that multiple packets in different buffers requesting for the same output port. Under such conditions, the Switch allocator uses priority based arbitration scheme to allot one of the output ports to one input buffer unit while the other packets are kept waiting. There are different arbiter implementations like fixed priority, matrix and round robin arbiters[13].
In fixed priority arbiters, the priority for different input ports are preassigned to a fixed value. This causes packets in low priority buffers to wait when a higher priority port is continuously asserted. So it is completely unfair.
The round robin arbiter will keep the priority cycling. If a certain port is granted during a cycle then in the next cycle the priority for this port will be the least. The implementation of round robin arbiter is simple and due to its strong fairness round robin arbiters are most commonly used[5].
Matrix arbiters also has strong fairness based on the principle of the least-recently-served policy. But their implementation is more complex.

1.3.3 Route Computation

Route computation units are equipped with routing algorithms that determine how a packet should be routed in the network. A more detailed explanation of routing methods is in the next section.

1.4 Routing

Routing determines which of the possible paths a packet actually takes in the network. The primary objective of router is to route the packets through the network and so the routing unit is inside the router. A good routing algorithm chooses the shortest path between a source destination and then send the packets via this route traversing different nodes. After the necessary number of hops, each packet will reach its destination from the source node. Another important feature that a routing algorithm should have is its ability to balance the traffic on the network. A more balanced traffic will reduce congestion in the network and thereby improves the performance of the NoC.

1.4.1 Classification of Routing

The routing method can be classified as per the location of routing decision as source routing and distributed routing. In source routing, the route for a packet is precomputed at the injection node and encoded to the head flit while in distributed routing, each packet carries the source and destination addresses for route computation in all intermediate nodes[14].
Based on adaptability, the routing can be classified into deterministic, oblivious and adaptive routing. In deterministic routing packets always take the same path between a source-destination pair. In oblivious routing the packets can take multiple paths between source-destination pair but it will be irrespective of the network congestion. Adaptive routing has the freedom to reroute packets through non-minimal paths as per the congestion on the network.
Adaptive routing algorithms can be classified as minimal routing where routing is through the shortest paths and non-minimal routing which allows longer paths. A minimal fully adaptive routing can route packets along any shortest path adaptively while a minimal partially adaptive routing cannot route packets along every shortest path[15].

1.4.2 Challenges of Routing

Certain challenges associated with routing are deadlock, livelock and starvation.
Livelock occurs when packet travel around their destination node, but unable to reach them due to the links being busy. Livelock as the name suggests has movement of data and therefore power wastage. It will occur in non-minimal adaptive routing where packets have the freedom to follow non-minimal paths.
Deadlock occurs when packets gets blocked inside the network because of multiple requests on a shared resource and none of them being granted. Deadlocks can be routing induced deadlock or packet dependent deadlock.
Starvation occurs when packets have priorities. The packets with low priorities never reaching their destination.

Dimension-ordered routing is the simple method to avoid deadlock while using deterministic routing. Being a non prioritized minimal routing, it also ensures livelock and stagnation free routing. In a 2D mesh topology, this routing is called XY routing where packets traverse the X direction first until it reaches the X-coordinate of the destination node and then Y direction is traversed until it reaches the destination

1.5 Flow Control

Another important functionality of the router is Flow control. Flow Control dictates which messages get access to particular network resources at a given time. Thus the flow control mechanism shall have a strong fairness in allocating a shared resource to multiple requests. A flow control strategy with strong fairness will allocate the shared resources with an equal priority to all the requesting components. This minimizes the delay or the overall latency and also avoids idling of resources reducing wastage of hardware cycles. Each router is equipped with various modules that collectively ensures all the above said operating features[16].
Flow control techniques are classified into packet based and flit based flow control.

Packet based flow control includes store-and-forward flow control where each node will forward a packet to the subsequent node only after the entire packet is received. This causes long delays for each hop and so delay critical networks does not employ this method. Another packet based flow control is virtual cut through flow control where packets can proceed to the next node before the entire packet is received. But this forwarding can happen only if sufficient buffer space to hold a complete packet is available in the downstream router.
Flit based flow control method includes wormhole flow control where flits can proceed to the next router before an entire packet is received. Unlike virtual cut through flow control, here a flit is forwarded from the current node if sufficient buffer space is available in the downstream router for this flit instead of packet. As longer packets could be occupying buffers of different routers, credit flow is necessary between the routers for implementing this method[17]. The buffer depth necessary for flit-based flow control is less than packet-based techniques. Due to faster operation and less buffering requirements, wormhole flow control is predominantly adopted for building NoC.

1.6 Motivation

With the multi core processing becoming more common, the efficiency of parallel processing can only be ensured by assuring the availability of all the required input data for every IP Core simultaneously. The single bus based communication approach cannot ensure an efficient parallel computation and a dedicated one to one connection increases the wiring overhead considerably. So NoC is proposed as a viable solution for this increasing complexity of on-chip communication problems. The advantages of NoC includes energy efficiency, reliability, scalability of bandwidth and distributed routing.

1.7 Outline Of Report

The remainder of this report is organized as follows:

Chapter 2 gives a brief overview to the parameters and metrics for an NoC. Also it discusses the past and recent works done in the domain of NoC.

Chapter 3 explains the design of NoC router which is used for all subsequent works. All the internal functions of the router and the packet structure used are covered in this chapter. The Vivado HLS synthesis results for the router block are also included.

Chapter 4 covers the design of a Scalable Square Mesh NoC using C++. It has the record of all simulation work done using the designed simulator. The various performance metrics used and the corresponding curves are also given.

Chapter 5 is a proposal for a new Simple Congestion Aware X/Y Routing Strategy. The concept and the simulation results are explained here.

Chapter 6 summarizes the works and concludes this project report. It also gives an insight into the future possibilities of this project.

Chapter 2

Literature Review

2.1 Literature Review

The topology is the major designing aspect of NoC. Once the topology is fixed the performance of NoC can be scaled by optimizing the router functions. Reconfigurable topologies can give better performance for different applications using the same architecture[18].

Huge NoCs need numerous routers which comes at high hardware costs so alternatives like router less NoCs are suggested[7]. Buffer less router configurations with scheduled routing is another alternative[19]. Energy efficient Bufferless routing could enable in order delivery using worm hole flow control and reduce large buffering requirements at receiver side[20]. Buffered router configurations commonly used can cause more latency while approximate buffer less routers could give better throughput[21]. Wireless network-on-chip (WiNoC) has short-range, radio-hub nodes which facilitate faster packet transfer when the source destination pair are far apart[22].

The router functions can be executed in pipeline increasing throughput and to make the NoC more faster. However reduction of pipeline stages is favorable with single cycle routers gaining popularity[23]. Optimized Virtual channels implementations can route flits in a single cycle and help overcome limitations of buffer space[24].

Route computation module within the router is responsible for decoding packets and deciding the output port for a packet[15]. Source routing requires larger header flit size for storing the routed path which results in lower bandwidth utilization. A detailed exploration of source routing shows low overhead for smaller NoCs[25]. However distributed routing remains the widely employed routing strategy. Adaptive routing algorithms gives lower latency and better throughput over deterministic routing but the power consumed by such routing are also on the higher side[26]. So energy efficiency of non minimal adaptive routing is a concern. Fault tolerant routing schemes are necessary with high integration density, as the chances for faulty links are high. Fault Tolerant routing algorithms are workaround methods for this case[27]. The effect of routing algorithms on WiNoC are studied[28].

For higher performance, adaptive routing with wormhole flow control is the best combination. To get the best performance gain from an adaptive routing the congestion in the network is an important parameter. Congestion awareness along with Fault Tolerance and process variation awareness on an adaptive routing for asynchronous NoC is shown in[29]. Congestion prediction algorithms could be used for a more balanced traffic distribution to improve the throughput and speed[30]. Low cost congestion detection method using Buffer Occupancy Value is implemented in[31]. The local congestion as well as global congestion information are equally relevant, a compromise between locally and globally congestion aware routing is shown in [32]. While deterministic routing is incapable of traffic awareness, they could be modified for traffic awareness and hotspot detection[33].

NoC traffic patterns are temporal in nature and so neural network based analysis and study of these temporal patterns are explored to develop congestion prediction algorithms based on buffer occupancy[34]. Certain routing methods works well

with some traffic patterns while for other traffic patterns the performance could be worse[35]. An architecture where the routing strategy is changed as per the local traffic pattern called traffic-robust routing algorithm is proposed in[36]. They have the advantage of better performance under any traffic pattern. CONNECT, is an NoC IP generator to produce synthesizable NoC implementations using Verilog[10].

Instead of monitoring the buffer occupancy of adjacent routers, the outgoing traffic based Traffic allocation adaptive routing can be implemented[37]. An intermittent XY routing where XY and YX routing are used alternatively at each subsequent nodes is proposed[38]. XY adaptive routing using context aware agent which tries to select the least congested path for routing is proposed in [39]. A centralized adaptive routing mechanism called Adaptive Toggle Dimension Order Routing (ATDOR), an extension of O1TURN model using central control mechanism is explained[40]. Effect of congestion aware adaptive XY-YX routing using pseudo random generator on Tianjic2 chip is studied in[41].

2.2 Vivado HLS

Vivado HLS is a High Level Synthesis (HLS) engine made by Xilinx. It can take in behavioural codes and algorithms written in C, C++ or SystemC and it can generate an RTL (Register Transfer Level) block with this behaviour. VHDL, Verilog and SystemC RTL descriptions are possible to be generated using Vivado HLS[42].

We begin with a C or C++ functional code. The code is the one that will be implemented as an RTL, so it is verified using a test-bench file written in the same language. Once the behaviour is verifed and found correct, we can proceed with the synthesis option to build the RTL HDL-based model for our code. Using the co-simulation feature, Vivado HLS generates test-vectors on its own using the user designed C/C++ level test-bench code. The co-simulation is used for the functional verification of the RTL generated[43]. After successful synthesis and co-simulation, the generated RTL shall be exported as an IP core. This IP can then be

imported using Vivado design suite to verify the functionality using simulations[44]. Additionally a bit-stream generation is also possible to burn this RTL into an FPGA for testing the same on hardware.

For a successful synthesis, hardware directives can be inserted into the C/C++ code. These directives indicate the Vivado HLS how the generated RTL should perform on a hardware. The synthesis report generated by the tool will show the estimated hardware utilization and the latency involved[45]. The hardware utilization is expressed as the total number of LUTs, BRAM, DSP48, Multiplexer, Registers etc used for the RTL block. The latency report shows the estimated number of hardware cycles required to complete the functionality of the entire module. The minimization of hardware utilization and latency is the key designing objective. The use of directives helps in achieving this goal to a larger extent[46]. More about these directives can be found in the user guide for Vivado HLS[47].

In HLS, designs can be started from scratch, or using a preexisting code written for a conventional CPU. There are additional libraries available within the VIvado HLS for hardware specific functionalities like shift registers, bit level variable assignment for counters etc. These libraries come in handy for the reduction of hardware and latency of the generated RTL. The use of these in built libraries as well as directives of Vivado HLS are promoted to synthesize a more optimal design. Another feature of this tool is the debugger mode analysis where more incite to each function and object inside the top module is provided. By using the analysis perspective the debugging is easier and more effective. The root function or method that cause the substandard design can be identified using the perspective available with Vivado HLS. So making the root corrections will bring out substantial improvements in the design.

Chapter 3

Design of Router

Routers are the brain for any NoC. Routers are used on a network for directing the traffic or packets from the source to the destination. It coordinates the data flow which is very crucial in communication networks. Routers are intelligent devices that receive incoming data packets, inspect their destination and figure out the best path for the data to move from source to destination. A router has a collection of registers, switches, function units, and control logic that collectively implement this functionality. Although many router architecture exists today, the router designed and used for this project is an input buffered 2 stage router. The architecture of the router is explained below.

3.1 Router Architecture

Input-queued routers is used as the architecture of choice as shown in Figure 3.1[11]. This is a generic architecture for five port input-queued NoC routers. As seen from figure only the input side has the buffers to store the packets. Any packet that

cannot be forwarded immediately is temporarily held in FIFO buffers connected to the input ports of the router until they can proceed to the next hop. With VC, buffers are allocated as per the VC ID in the head flit. Advantages of VCs include deadlock freedom and bypassing for Head-of-Line (HoL) blocking. Without VC, each incoming packet will be queued to the FIFO buffers of the corresponding input port.

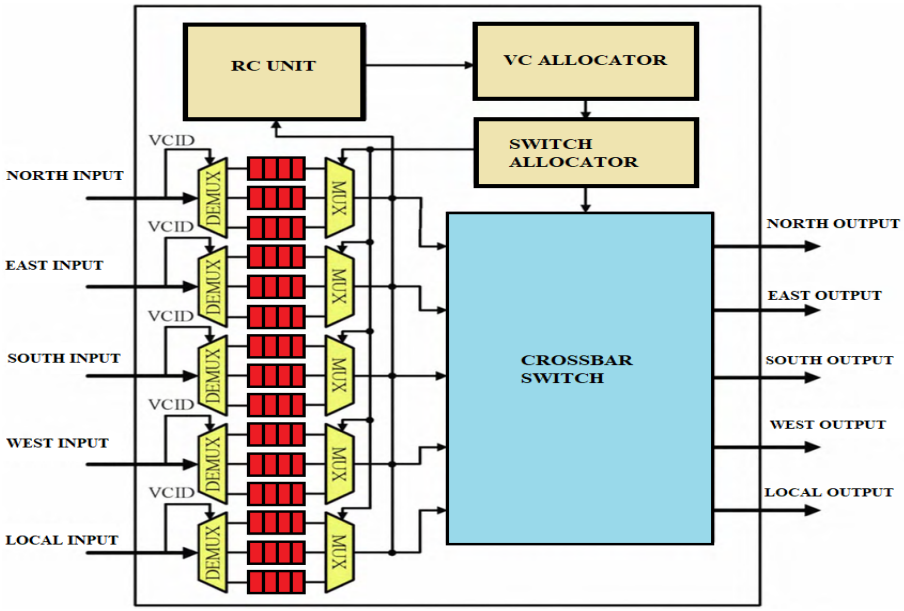

Figure 3.1: Input Buffered Router Architecture

The important parameters of FIFO buffers is the FIFO depth and width. FIFO Depth indicates the number of flits a buffer can store and the FIFO Width indicates the bit size of the buffers. The FIFO Width should match the bit size of the flits. For this design bit width for the flits and buffers are chosen as 32 bits.

The ports indicated in the figure makes connection to the adjacent routers using the four directional ports and the local port is used for making connection with the local IP core of each node. These five ports enables bidirectional communication possible by using two set of unidirectional wiring for either side. These five ports of the router are arranged and indexed as shown in Figure 3.2

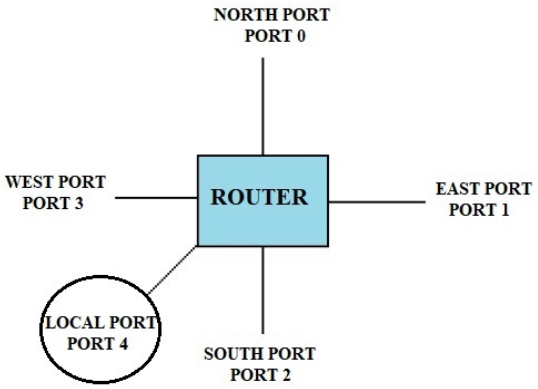

Figure 3.2: Router Port Arrangement

As explained earlier, a routerś primary function is to forward each flit that arrives on one of its input ports to the appropriate output port. To this end, the internal blocks shown in Figure 3.1 has to execute their corresponding functions as below to achieve the router functionality. These operations are generally termed the router pipeline stages[5].

- **Route Computation (RC):** The RC block decodes the head flit of each packet to identify its destination node id and compares it with the current router id and determines to which output port the corresponding packet is to be routed. Route computation also needs to be done only for the head flit of each packet.

- **Virtual Channel (VC) Allocation:** Once route computation is completed, the packet requests an output VC from the VC Allocator. After successful VC Allocation, each flit is forwarded to this VC. VC allocation only needs to be performed for head flits, as the assigned VC is inherited by the subsequent body and tail flits of the same packet.

- **Switch Allocation:** The switch allocator unit is uses a crossbar switch and connects input buffers to the output ports. If multiple packets request for the

same output port, the switch allocation serves them as a queue using arbiters. Thus Switch Allocator unit resolves conflicts between packets in different input buffers destined for the same output port.

- **Switch Traversal:** After receiving a grant or ready signal from the switch allocator, the last stage of pipeline where the flit traverse through the crossbar switch to arrive at its destination output and continue traversing through the network.

3.2 Specifications for Router Design

Table 3.1 shows the specifications for design of the router using Vivado HLS.

No of In Ports	5
No of Out Ports	5
Input Buffer Depth	8*
No of VC	2
Bit Width of Buffer	32**
Flow Control Scheme	Store and Forward***
Routing Algorithm	XY Dimension Ordered Routing
Switch Allocation Arbitration Scheme	Round Robin

Table 3.1: Design Specification of Router

*Buffer depth can be changed by defining the FIFO_DEPTH parameter in para.h file
**For all subsequent work, the flit size and buffer width are fixed at 32 bits
***For all subsequent work, store and forward flow control is employed.

As explained before, in Store and forward flow control, the router waits until an entire packet is received at the input buffer before beginning the subsequent functions like route computation, switch allocation and switch traversal.

3.2.1 XY Dimension Ordered Routing

Dimension-ordered routing which is an example of a deterministic routing algorithm is implemented within the router. In a 2-D topology such as the mesh topology, XY dimension-ordered routing sends packets along the X-dimension first, followed by the Y dimension[48]. Due to its simplicity, DOR XY routing is the most commonly used routing algorithm for NoC. The generic algorithm for implementation of XY DOR is as below.

Algorithm 1: XY Dimension Ordered Routing

```
if router_id = dest_id then
    Proceed to LOCAL PORT[4];
else if router_col < dest_col then
    Proceed Right or EAST[1];
else if router_col > dest_col then
    Proceed Left or WEST[3];
else if router_row < dest_row then
    Proceed Up or NORTH[0];
else
    Proceed Down or SOUTH[2];
```

3.2.2 Round Robin Arbiter for Switch Allocation

For switch Allocation, a round-robin arbiter is used. Here the arbiter take the entire set of requests for an output port and grant one of the input port as per the current priority. In the next cycle the input port already granted will have the least priority while the input port next to that will have the highest priority. This way in a round robin fashion the grant signals are generated with strong fairness[13]. The complexity of implementing a round robin arbiter is also quite low[49].

3.3 Structure of Packet

The packets used for all subsequent works has the following generic structure[50]. The flit size is 32bit and every packet has a head flit and a tail flit. When the packet size is more than 2 flits, then body flits are inserted between the head and tail flits. For major portion of this work, the packet size is 4 flits with one Head flit, two Body flits and a Tail flit. The packet size can be changed from the global parameters header file **para.h** using the parameter PACKET_SIZE
The classification of flits are based on the top 2 MSB values i.e. bits 31,30 (as LSB starts at index 0) as in Table 3.2. The head flit stores destination node id and source node id for route computation and switch traversal purposes. The structure of head flit is shown in Table 3.3. For the body flit and tail flit, bits from 29 to 0 are assigned the destination node id value instead of 0 so that body and tail flits for different packets can be identified separately. Vivado HLS has an inbuilt **range()** function for the addressing a range of bits[47].

Bit 31,30	Flit Type
11	Head Flit
10	Body Flit
01	Tail Flit

Table 3.2: Flit Classification

Bits	31,30	29, ..., 22	21, ..., 8	7, ..., 0
Notation	Flit Type	Destination ID	Unused bits	Source ID

Table 3.3: Head Flit Structure

3.4 Vivado HLS Synthesis of Router

The structure and operation of router is defined using C++ programming language in Xilinx Vivado-HLS[49]. The major blocks of the router like Input Buffers, VC Allocator, Switch Allocator and Arbiter are defined using individual C++ class. The objects of these classes are declared within the main class, **router_5port.** The top function, **router_top()** creates a static router_5port object and executes all the router functions explained in Section 3.1. The entire behavioural operation of router top function is depicted using flowchart in Figure 3.3. The top function also creates the following RTL interfaces for each 5 directions as mentioned in Figure 3.2:

1. **In Port:** 32 bit input ports for reading incoming flits from 5 directions.
2. **In Req:** 1 bit (Boolean) input request ports are used to indicate a request is coming for the corresponding input port.
3. **In Ready:** 1 bit input ready ports to acknowledge the incoming input request and to indicate if the input ports are ready to read flits through the corresponding input port.
4. **Out Port:** 32 bit output ports for sending flits to 5 directions as per the RC operation.
5. **Out Req:** 1 bit output request ports to send a request to the adjacent router or local IP core in order to initiate packet transfer.
6. **In Ready:**1 bit output ready ports checks if the adjacent router/IP core is ready to read flits that will be send through the corresponding output port.

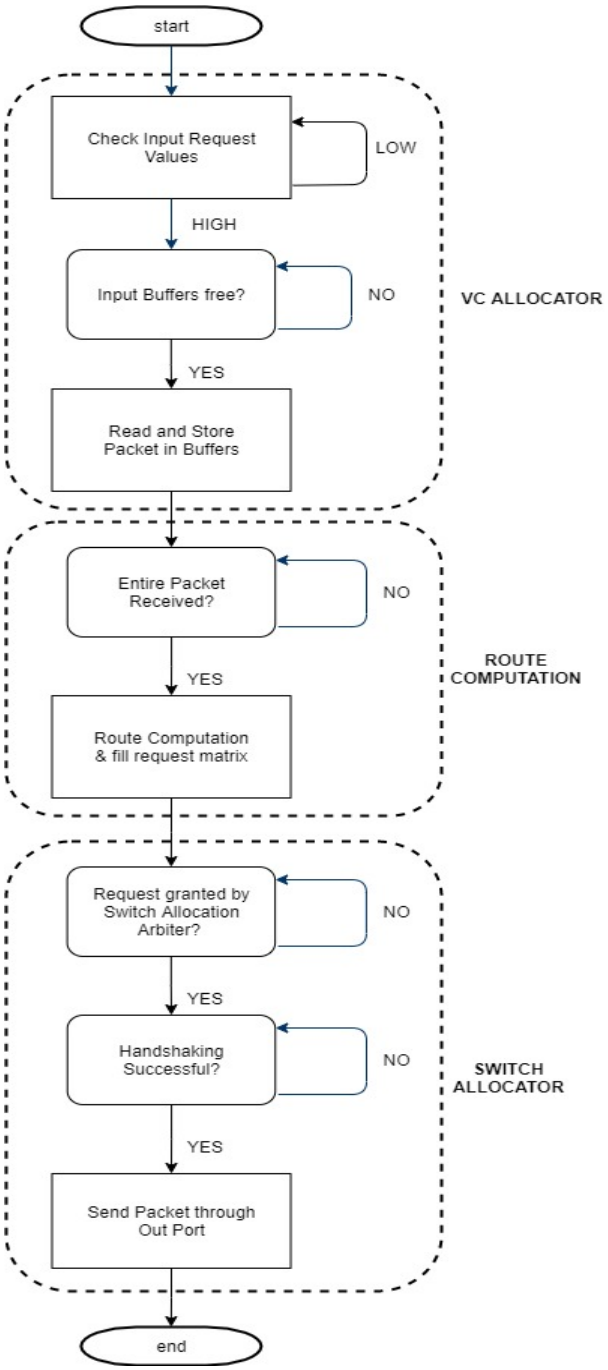

Figure 3.3: Router Behavioural Level Operation

As mentioned before, the router employs store and forward flow control. Even though this technique has the disadvantages like long delays at each hop and long buffers at each router are required to store the entire packet, the handshaking signals can be implemented using a single bit request and ready signal for each port instead of more sophisticated structures like the ones used for credit based flow control[17].

The router is synthesized for the target device **xczu11eg-ffvc1156-1-i** with single FIFO buffers for each port and without any virtual channels. The synthesized 5 port router IP block is shown in Figure 3.4. The timing summary of the synthesized RTL is shown in Figure 3.5 and the hardware utilization report is shown in Figure 3.6

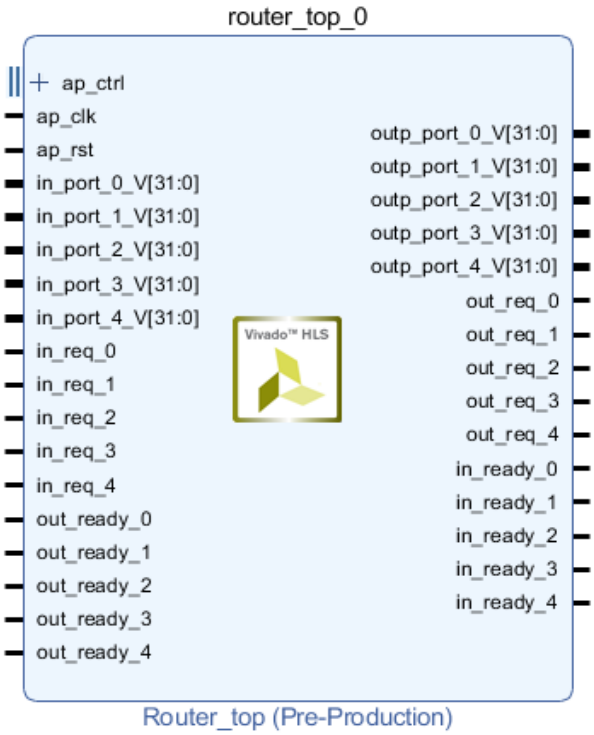

Figure 3.4: Synthesized Router Block

- **Timing (ns)**
  - **Summary**

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	7.600	1.25

- **Latency (clock cycles)**
  - **Summary**

Latency		Interval		Type
min	max	min	max	
4	4	4	4	none

Figure 3.5: Timing Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	-	-	-
FIFO	-	-	-	-	-
Instance	-	-	373	9050	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	1098	-
Register	-	-	1860	-	-
Total	0	0	2233	10148	0
Available	1200	2928	597120	298560	80
Utilization (%)	0	0	~0	3	0

Figure 3.6: Hardware Utilization Estimate

3.5 RTL Simulation of Router IP Block

For the RTL verification of synthesized router IP, Vivado Design Suite is used. An additional 32 bit port termed **num_pkts** is added to the previously designed router block. This port indicates the number of packets received by the router through the corresponding input port. Another IP block named Local CPU which represents the local IP core connected to the routerś local port (4) is also synthesized using Vivado

HLS. The local IP core also uses the same handshaking signals as the router. Both these blocks are exported as RTL IP from Vivado HLS and imported in the Vivado Design Suite for simulation. They are interconnected as shown in Figure 3.7. All other input ports of router are connected to constant 0 which represents a LOW voltage value.

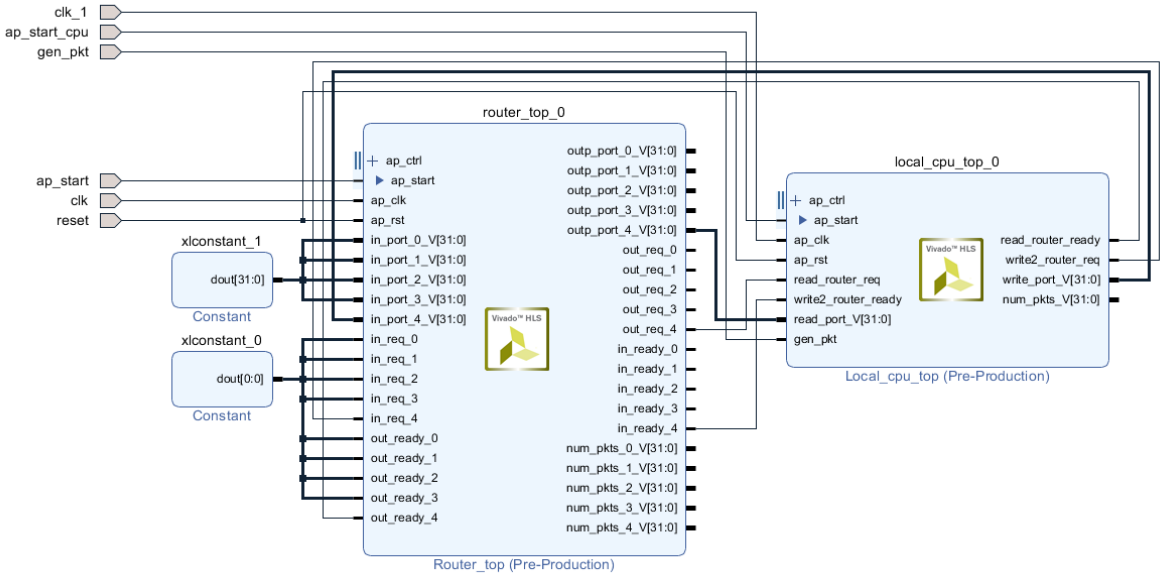

Figure 3.7: Router and CPU Connection

The simulation procedure is as follows:

1. All blocks are provided clock and **ap_start** signals to begin functioning.
2. When a **gen_pkt** request is send to the cpu, it generates a packet that has the same destination id as the router id and request to send it to the router. For this simulation this id is set as 12.
3. The router after successful handshaking with the cpu loads the packet into its buffer connected to input port 4 (local port) in a flit by flit procedure.
4. After the entire packet is received by the router, it performs RC and will generate a output request for Out port 4.

5. After successful handshaking between cpu and router, the cpu will read the packet send by router flit by flit.

6. When an entire packet is received by the cpu, it will increment the **num_pkts** port of the cpu block by 1 to indicate the same.

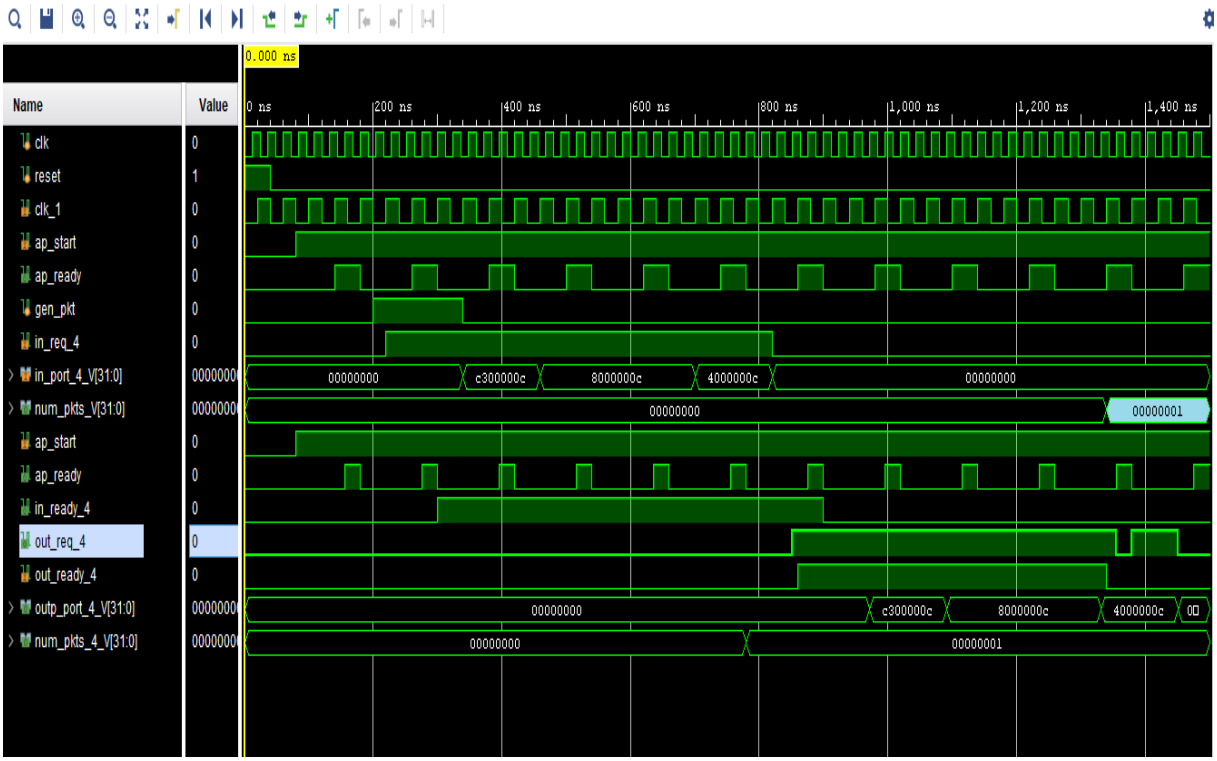

Figure 3.8: RTL Simulation Result

The simulation result is shown in Figure 3.8. Note the value at num_pkts pin (9th Row from top) is incremented by 1 (highlighted) at the end of simulation. Also all other ports such as gen_pkt, in port 4, in req 4, in ready4, out port 4, out req 4, out ready 4 are shown in the Figure. Also it is seen that the individual flits written at the input port 4 are the same ones coming back through the output port 4. This simulation successfully verifies the functionality of the RTL generated by Vivado HLS for the C++ code used.

Chapter 4

Scalable Square Mesh NoC Simulator

Network on Chip is the interconnected network of different router objects or nodes to facilitate multi core communication. There are numerous architecture topologies for NoC like mesh, torus, ring, butterfly, fat tree and so on. The most common

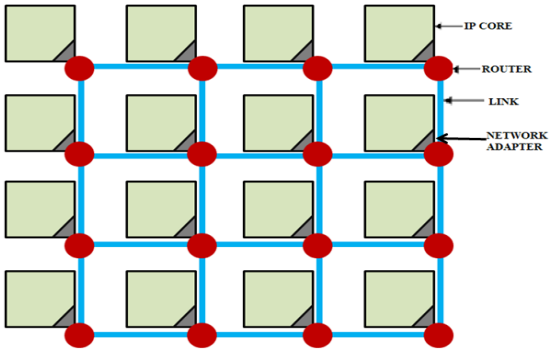

Figure 4.1: 4x4 Mesh Topology

topology used is the mesh due to its simplicity. In a mesh topology the nodes are arranged along multiple rows and columns. For a 4x4 Mesh NoC, there is a total of 16 nodes arranged in 4 rows and 4 columns as shown in Figure 4.1.

4.1 Simulator Design using C++

The router designed for the previous work is a five port router that can send and receive flits in all four directions and the local port which connects the core. The same C++ code is used to create router_5port objects. In addition to this a new class named **noc_mesh** is defined which is used to create a scalable square mesh NoC using these router_5port objects which represents the nodes in an NoC. The global parameters header file **para.h** has a parameter named MESH_SIZE which is used to represents one side of a square mesh. Thus a square of MESH_SIZE * MESH_SIZE is created inside the noc_mesh class. This class also has methods to execute the internal functions of all the routers inside the mesh simultaneously. These methods are called inside the top function. The top function named **noc_mesh_top()** has

ROWS	COL 1	COL 2	COL 3	COL 4
ROW 4	13	14	15	16
ROW 3	09	10	11	12
ROW 2	05	06	07	08
ROW 1	01	02	03	04

Table 4.1: Node Numbering and Arrangement For 4x4 Mesh NoC

a collection of N number of input, output and the associated request and ready signal ports to connect to the core of N nodes where N represents MESH_SIZE X MESH_SIZE. All these ports are connected to the port number 4 or local port of each

node. The nodes are numbered from 1 to N. The arrangement for a 4x4 mesh is as shown in the Table 4.1 and the arrangement for a 8x8 square mesh is as shown in the Table 4.2. This numbering is important for making the correct connection between ports of different router_5port objects and also for route computation purpose.

ROWS	COL 1	COL 2	COL 3	COL 4	COL 5	COL 6	COL 7	COL 8
ROW 8	57	58	59	60	61	62	63	64
ROW 7	49	50	51	52	53	54	55	56
ROW 6	41	42	43	44	45	46	47	48
ROW 5	33	34	35	36	37	38	39	40
ROW 4	25	26	27	28	29	30	31	32
ROW 3	17	18	19	20	21	22	23	24
ROW 2	09	10	11	12	13	14	15	16
ROW 1	01	02	03	04	05	06	07	08

Table 4.2: Node Numbering and Arrangement For 8x8 Mesh NoC

The top function creates a static object for noc_mesh class and calls the method to perform internal functions of all routers. The router functions are executed in two stages. In the first stage all the input ports are read and buffers are loaded using VC allocation function for all nodes. In the second stage the rest of the router functions like RC, Switch allocation and traversal are performed. These 2 stages are considered as 2 cycle operation and so for each top function call, an equivalent two hardware cycles are necessary.

A testbench is written which will make successful handshakes between router and IP Core and thereby transfer packets in and out of the NoC. It also has a

global counter variable named **fn_call_ctr** which records the number of times the top function, **noc_mesh_top** is called. Due to the 2 stage operation of the routers, the total cycles taken is twice the global counter value. The generation and receiving time for each packet are also recorded in the testbench file in terms of the count from the global counter variable. These values are used in estimation of latencies as explained in Section 4.2

The top file and its associated headers along with the testbench completes the Scalable Square Mesh NoC Simulator.

4.2 Estimation of Latency

For every packet injected into the NoC, the tail flit is assigned a packet id and its generation time is recorded. Once a tail flit of any packet is received from the NoC as per its packet id the received time is recorded. Structures in C++ are used for recording packet details. The modified tail flit structure is shown in Table 4.3.

Bits	31,30	29, ..., 23	22, ..., 8	7, ..., 0
Notation	Flit Type	Unused bits	Packet ID	Destination ID

Table 4.3: Modified Tail Flit Structure

Source IP Core to Router Handshaking takes 2 cycle
Router to Destination IP Core Handshaking takes 2 cycle
Total IP Core Operating Time = 4 Cycles per Packet

A snippet of Packet details recorded using the simulator is shown in Figure 4.2. The hops indicated includes the source core to router hop, router to router hops and router to destination core hop. The WAIT_T represents waiting time for each packet, ST_LAT represents link traversal latency. The estimation of latency on the designed NoC is explained in the below equations[51].

Total Latency	= (Pkt Rec Time - Pkt Gen Time) * Router Operation Stages
	= (Pkt Rec Time - Pkt Gen Time) * 2
Link Traversal Latency	= Total Hops * Packet Size * Router Operation Stages
	= Total Hops * 4 * 2
Waiting Time	= Total Latency - Traversal Latency - IP Core Operating Time
	= Total Latency - Traversal Latency - 4

PRINTING DETAILS OF PACKET

PKT_ID	DST_ID	SRC_ID	GEN_T	REC_T	HOPS	TOT_LAT	ST_LAT	CPU_CYC	WAIT_T
1	10	1	10	34	5	48	40	4	4
2	4	2	10	29	4	38	32	4	2
3	15	3	10	40	5	60	40	4	16
4	5	4	10	41	6	62	48	4	10
5	2	5	10	29	4	38	32	4	2
6	13	6	10	34	5	48	40	4	4
7	11	7	10	24	3	28	24	4	0
8	15	8	10	59	5	98	40	4	54
9	3	9	10	39	6	58	48	4	6
10	1	10	10	34	5	48	40	4	4

Figure 4.2: Packet Details Recorded

4.3 Traffic Patterns

There are several synthetic traffic patterns that can be used in a mesh network. The random traffic pattern is the easiest to design and implement. But for practical purposes static synthetic traffic patterns like bit reversal, bit complement, bit rotation, shuffle, transpose, tornado, neighbor etc.[4] are desirable. The table 4.4 shows the logic for traffic pattern generation[5].

Different kinds of traffic patterns cause distinct sharing of the physical channels. Also different routing algorithms cause difference in channel usage for the same traffic pattern. The congestion in a network is caused by the packet injection rate,

Type	Pattern
Random	λsd = 1/N
Bit Permutation	
Bit complement	di = ¬si
Bit reverse	di = sb-i-1
Bit rotation	di = si+1 mod b
Shuffle	di = si-1 mod b
Transpose	di = si+b/2 mod b
Digit Permutation	
Neighbor	dx = sx + 1 mod k

Table 4.4: Traffic Patterns

packet spatial distribution and the packet size[3]. Each traffic patterns creates its own spatial distribution and thereby congestion[35].

The traffic patterns implemented with this simulator are random, shuffle, neighbor and transpose. In a random traffic pattern, each node is equally likely to send a packet into any other node on the network. In shuffle traffic pattern, the nodes at odd location sends packet to nodes at even location an vice versa. In neighbor traffic, each node sends packet into its adjacent node except the nodes at the top and right edges. They send packets to the nodes on the opposite edge. The transpose traffic is similar to the matrix transpose where each node will send packets to the node at it transpose location.

4.4 Simulation Results

The Scalable Square Mesh NoC build using C++ is used as a simulation tool to estimate and verify the effect of different parameters on the performance of the NoC. The simulator has provisions for varying the parameters like traffic pattern, packet size, FIFO depth, mesh dimension and packet injection interval. Routers with XY DOR Algorithm is used with store and forward flow control. The performance

metrics estimated by the simulator are the total packets injected, average value of latency and waiting time as well as the peak values of latency and waiting time. The effect of varying different parameters on the performance of the NoC are explained in the subsequent sections.

4.5 Performance Metrics

4.5.1 Total Packet Handled

When the IP cores tries to inject packet at certain intervals into the NoC, the network or more precisely, the routers can accept the packets only if there are sufficient free buffers at its input port connected to the IP core. The free buffer availability is also a measure of the congestion within the network which affects the overall speed of network. A faster queuing and dequeuing operation at the buffers will result in more number of packets handled. Other factors like traffic pattern and routing method also directly influence the dequeuing rate. For a store and forward flow control, the total number of packets that can be handled by the network shows the saturation rate and the total information exchange possible using the network. Thus this is an important metric to estimate the performance of the NoC.

4.5.2 Latency

Latency is a measure of how long it takes for a packet to start from its source to reach its destination by executing several hops through multiple intermediate nodes. The latency includes the link traversal time which is directly proportional to the packet size for store and forward flow control[51]. The latency also includes waiting time which is explained below. The average latency is an indication of the overall speed of the NoC. It is estimated by taking the average of latencies incurred for every packets injected to the network. The peak latency is another important metric that

summarizes the entire performance of the whole chip. The peak values indicates how long the associated IP cores has to wait before proceeding with the previous task.

4.5.3 Waiting Time

Waiting time is the time during which the packet sits idle inside the routers waiting for its turn in route computation and switch allocation. This is a metric that indicates the fairness of shared resources as well as the resource wastage. Longer waiting times leads to huge latencies and the associated destination IP core process will also be delayed considerably. It is always the objective to reduce the waiting time and thereby the latency of packet transfer. Here also the average value indicates the overall speed of the network while the peak value indicates the delay incurred for processes and threads inside the IP cores.

4.6 Simulation Methodology

The performance of the NoC with different parameters are estimated through multiple simulations. In every case the simulator runs for more than 5000 cycles with the first 250 cycles as the warm-up period for the NoC. During this warm-up period, traffic builds up in the NoC and the estimation of latency and waiting times are performed only after this period to improve the consistency of values. Also this warm-up period builds congestion within the network so that the estimated values will be more reliable. In the first 5000 cycles the packets are injected at the certain interval into the NoC by different nodes at random time. After 5000 cycles the packet injection process stops and the simulator waits for all the injected packets to come back from the NoC before stopping the simulation and estimating the performance metrics. The number of cycles the simulator will run after 5000 cycles depends on the remnant packets on NoC at the end of injection process.

4.7 Effect of FIFO Depth and VC

The above explained simulation employing XY DOR is performed on both 4x4 and 8x8 Mesh NoC for average packet injection intervals of 12,15,18,25 and 40 cycles for the following router input buffer configurations.

(i) Configuration 1: 1 VC: FIFO Depth 8

(ii) Configuration 2: 1 VC: FIFO Depth 16

(iii) Configuration 3: 2 VC: FIFO Depth 8

The traffic pattern used is Random traffic where each node is equally likely to send a packet to any other node. The packet size is fixed at 4 flits.

The heat map of the traffic, Figure 4.3 shows the distribution of traffic on a 4x4 NoC for a packet injection interval of 12 cycles with routers having 2 VC of fifo depth 8 flits. The Figure 4.4 shows the same for an 8x8 Mesh NoC under same conditions.

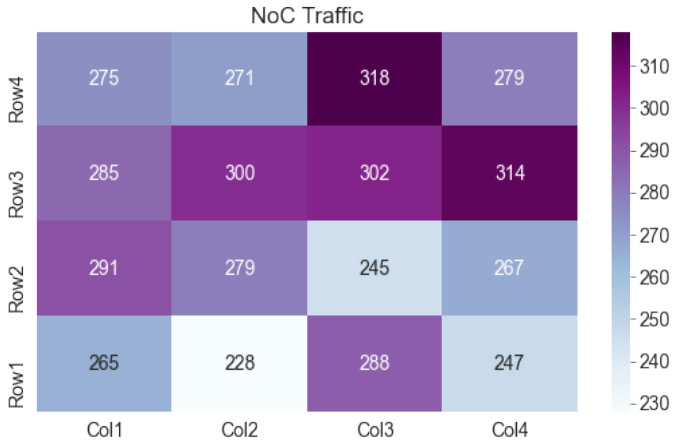

Figure 4.3: Random Traffic Distribution on 4x4 NoC

The numbers on each cell represents the number of packets handled by the network with that particular destination node. So it indicates the total packets received by the node as per the node arrangement table. The arrangement of routers or nodes are shown in Table 4.1 and Table 4.2 for 4x4 and 8x8 NoC respectively.

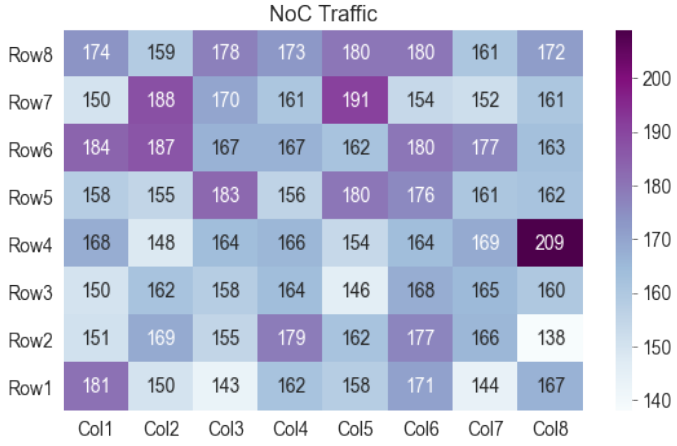

Figure 4.4: Random Traffic Distribution on 8x8 NoC

4.7.1 Total Packets Handled

As explained before, the simulator will try to inject packets at various intervals but the NoC can only accept packet if there is sufficient buffer space in the corresponding routers. The Figure 4.5 shows the performance curve for 4x4 Mesh and the Figure 4.6 shows the performance curve for 8x8 Mesh.

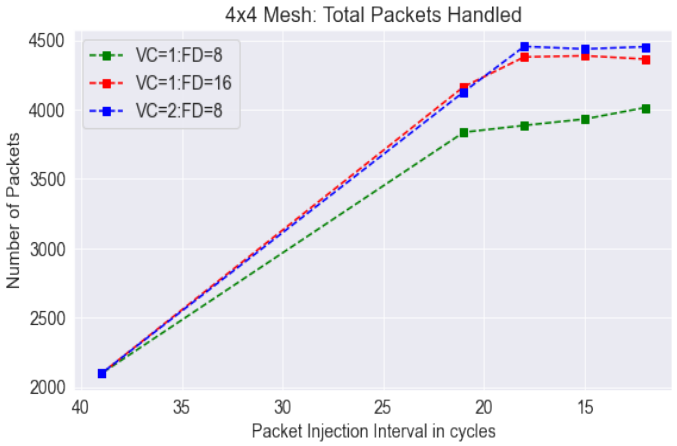

Figure 4.5: 4x4 Mesh: Total Packets Handled

The total packets handled will reach a saturation point as the packet injection

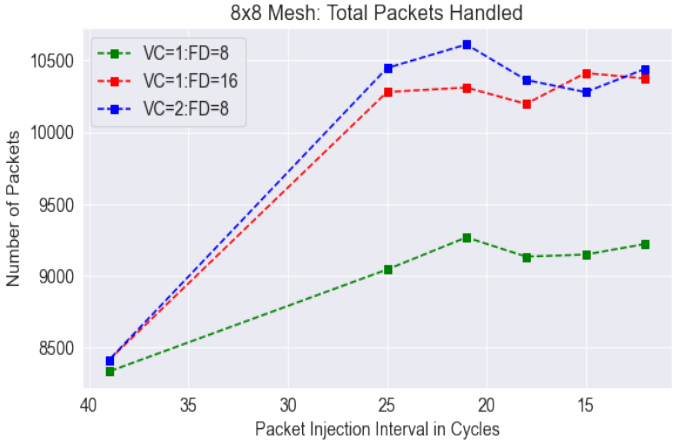

Figure 4.6: 8x8 Mesh: Total Packets Handled

interval is decreasing or packet injection rate increasing. The saturation occurs due to the build up of congestion in the network. The performance of three configurations are shown which clearly indicates the effect of VC over increasing FIFO depth. It is seen that the saturation limit is highest for 2 VC configuration and lowest for single VC with FIFO depth of 8 flits.

4.7.2 Average Latency and Waiting Time

The average latency and waiting time is a measure of the network speed for varying traffic. The congestion is directly proportional to these parameters as shown. The Figure 4.7 shows the performance curve for 4x4 Mesh and the Figure 4.8 shows the performance curve for 8x8 Mesh.

Here the effect of VC shall be compared against the 2nd configuration i.e. 1 VC and FIFO depth of 16. It is seen that employing another VC is better than doubling the FIFO depth. Any comparison of configurations 2 and 3 with configuration 1 is not appropriate as the traffic and the number of packets handled by them are entirely different.

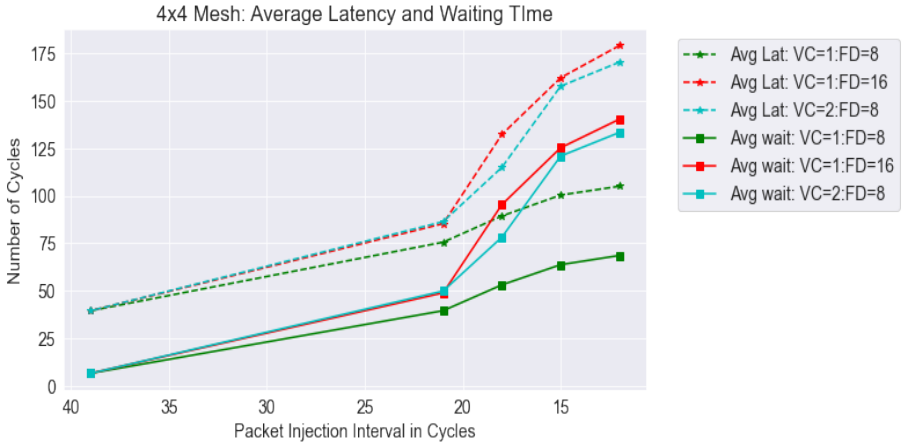

Figure 4.7: 4x4 Mesh: Average Latency and Waiting Time

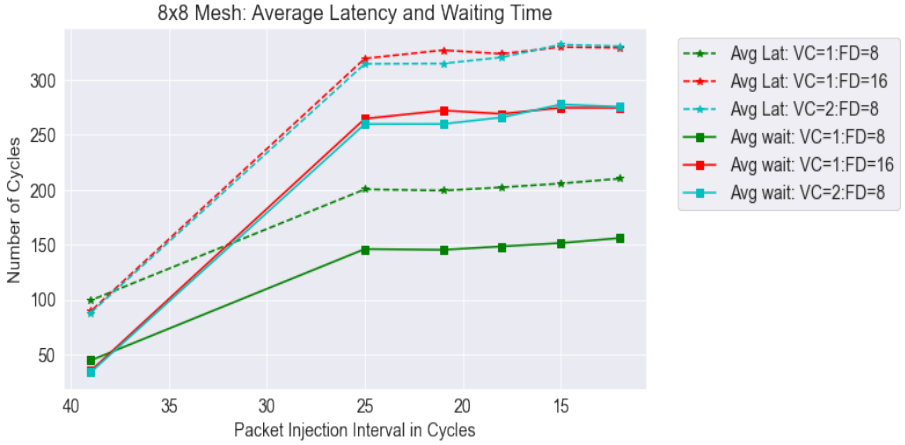

Figure 4.8: 8x8 Mesh: Average Latency and Waiting Time

4.7.3 Peak Latency and Waiting Time

The peak latency and waiting time is an important metric that summarizes the entire performance of the whole chip. The Figure 4.9 shows the performance curve for 4x4 Mesh and the Figure 4.10 shows the performance curve for 8x8 Mesh.

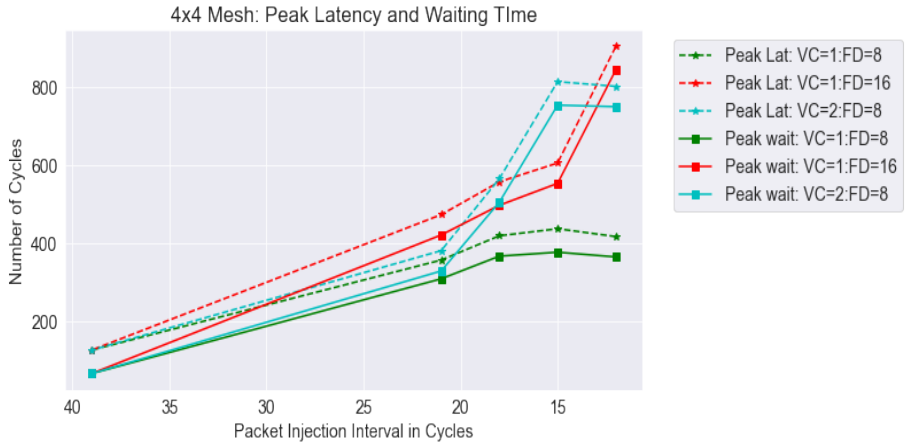

Figure 4.9: 4x4 Mesh: Peak Latency and Waiting Time

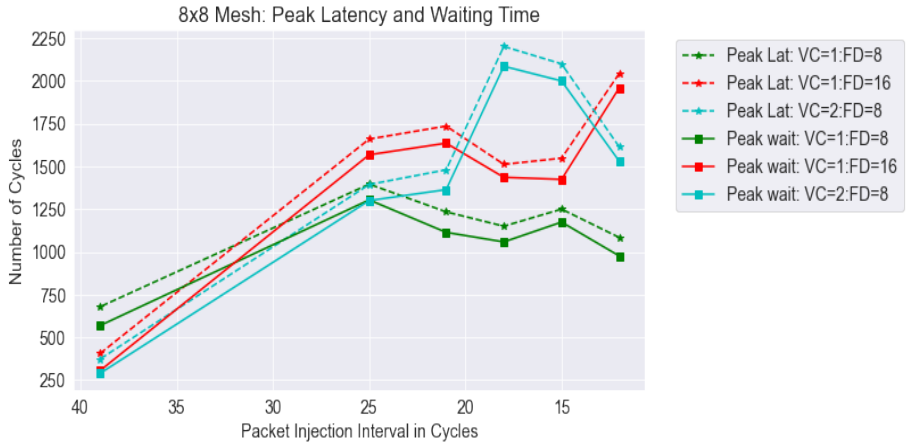

Figure 4.10: 8x8 Mesh: Peak Latency and Waiting Time

Here also the effect of VC shall be compared against the 2nd configuration. It

is seen that employing another VC is better than doubling the FIFO depth under highly congested traffic. But for packet injection interval of 15 cycles the 2nd configuration leads to lower peak values. This is due to first in first out scheme employed under no VC for configuration 2. The designed router with 2 VC operates as follows, the 2nd VC starts filling only when the 1st VC is filled entirely. When the 2nd VC loads a packet, the dequeuing operation for the next cycle is prioritized for the packet in VC2 and so during this time the packet in the 1st VC is waiting. Once the packet in VC2 is dequeued then the next priority is for VC1. So this alternating priority for route computation will adversely affect the first in first out nature of the NoC router. Such a priority switching between VC may cause the 2nd packet in VC1 to endure longer waiting queues.
As in the case of the previous metric, any comparison of configurations 2 and 3 with configuration 1 is not appropriate as the traffic and the number of packets handled by them are entirely different.

4.8 Performance with Synthetic Traffic Patterns

The designed NoC is simulated with static synthetic traffic patterns explained in Section 4.3. This simulation is done with a packet size of 4 flits and packet injection interval of 18 cycles. XY DOR is employed. The simulation is run on 3 different mesh networks viz. 4x4, 6x6 and 8x8 mesh. In each case the router used has a FIFO depth of 16 flits. Different router buffer configurations will show similar trend for this parameter variation.

4.8.1 Total Packets Handled

Figure 4.11 shows the total packets handled against traffic patterns for different mesh dimensions. The total packets handled will depend on the mesh size directly. The bigger mesh will have more IP cores and routers and therefore the packets

handled will be higher compared to smaller mesh dimension. All the traffic patterns show similar trend and the neighbor traffic will handle much more packets as here the nodes inject packets into the adjacent nodes. So the congestion build up will be lower in this case compared to other traffic patterns.

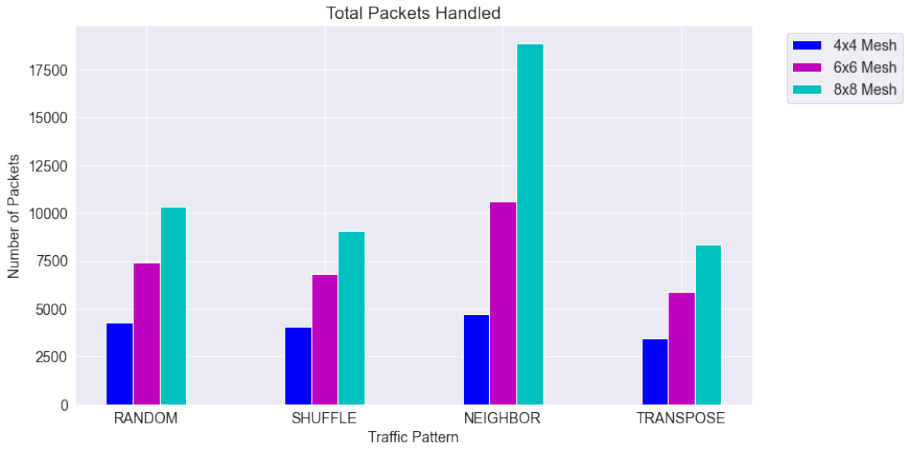

Figure 4.11: XY Routing: Packets Handled Vs Traffic Patterns

4.8.2 Average Latency

Figure 4.12 shows the variation of average latency with traffic patterns. The average latency also depends on the mesh size directly. The bigger mesh will have packets with longer hops between source and destination and therefore the average latency will also grow with mesh dimension. It can be seen that with shuffle traffic, the bigger mesh will cause substantial increase in network congestion leading to a higher growth rate for average latency. In neighbor pattern, the congestion build up will be lower due to adjacent source destination pairs and hence here the latency endured will be much less.

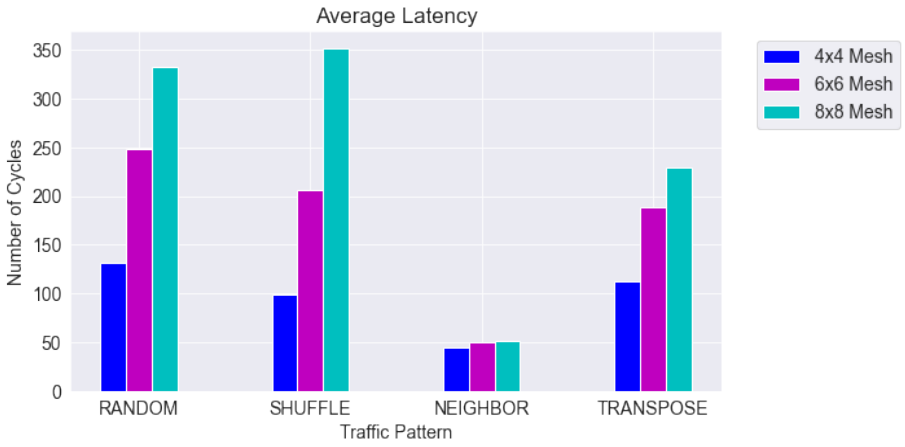

Figure 4.12: XY Routing: Average Latency Vs Traffic Patterns

4.8.3 Average Waiting Time

Figure 4.13 shows the average waiting time plot against traffic patterns. Here also a similar trend as in case of average latency is seen. The reasons being the same mentioned before. It can be seen that with shuffle traffic, the bigger mesh has

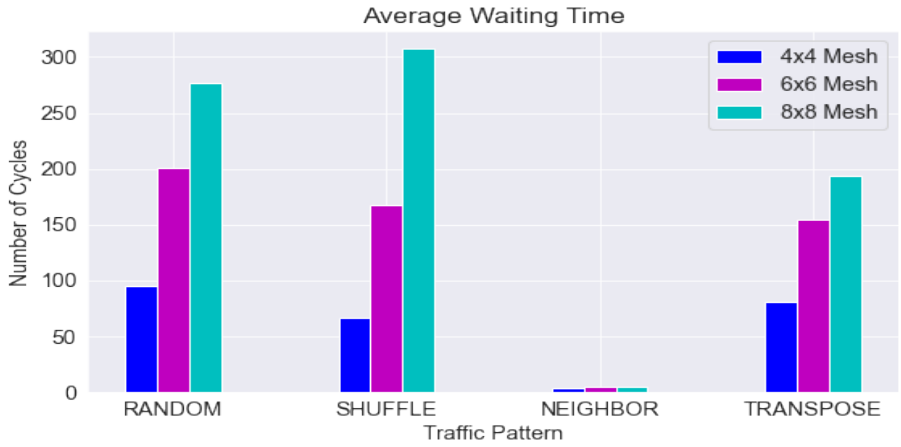

Figure 4.13: XY Routing: Average Waiting TIme Vs Traffic Patterns

considerable waiting time which proves the build up of network congestion. Also for neighbor pattern, the average waiting time is negligibly small which cause the

least latency and highest number of packets handled suggesting there is hardly any congestion for neighbor traffic. Also it is noteworthy that the same injection rate is causing considerable amount of congestion for other traffic patterns.

4.8.4 Peak Waiting Time

Figure 4.14 is the graph showing peak waiting time for different traffic patterns. The peak waiting time is direct indication of the network congestion. It is seen here

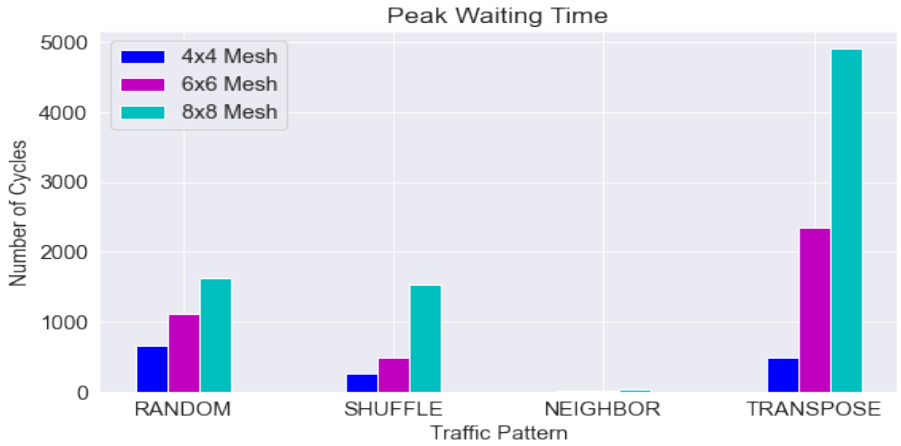

Figure 4.14: XY Routing: Peak Waiting TIme Vs Traffic Patterns

that for shuffle traffic pattern the peak waiting time incurred is much higher for the 8x8 mesh as compared to other two meshes. This explains the growth of latency and waiting time for shuffle traffic pattern in the previous curves Figure 4.12 and Figure 4.13. The interesting case here is the transpose traffic. It is known that in transpose traffic the the node at top left corner will send packets to bottom right corner (diagonally opposite) and vice versa. This causes the total hops necessary to grow along with the mesh dimension for these particular source destination pairs. These packets are the ones that will suffer longer waiting time and latency. Hence even though the average values are smaller than random and shuffle pattern, the peak waiting time for transpose traffic is comparatively huge.

4.9 Effect of changing Packet Size

To study the effect of changing packet size, a 4x4 mesh is used with the 3 router buffer organization as explained in Section 4.7. XY DOR is used here as well. Random traffic pattern is used with a packet injection interval of 21 cycles. Packet size of 2 flits, 4 flits and 8 flits are used for comparison using the performance metrics Section 4.5. Note that the packet size was 4 flits for all the previous simulations.

4.9.1 Total Packets Handled

Figure 4.15 shows the total packets handled against packet size for different buffer organization. The total packets handled will depend on the packet size inversely. Bigger packet size will need more buffer space and therefore the number of packets handled will be lower for increasing packet size. All the configurations show similar

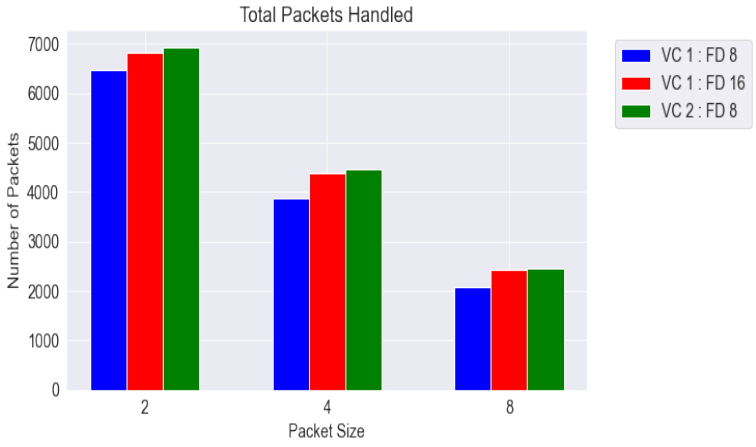

Figure 4.15: XY Routing: Packets Handled Vs Packet Size

trend but the saturation point will be higher with increasing FIFO depth and VC. So they will handle more packets. With the packet injection rate below saturation both configuration 2 and 3 will handle same number of packets at different speeds as explained in subsequent sections.

4.9.2 Average Latency

Figure 4.16 shows the variation of average latency with packet size. The average latency directly depends on the packet size. The larger packets need more time to traverse between source and destination and hence their average latency is more as expected. It can be seen that with 2 VC i.e. configuration 3, the average values

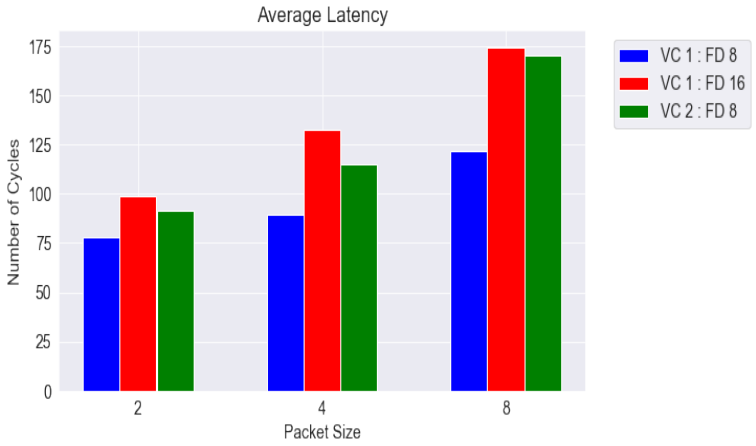

Figure 4.16: XY Routing: Average Latency Vs Packet Size

of latency is always lower than the value for configuration 2. This is due to the breaking of longer queues of configuration 2 which will result in improvements in waiting time and shared resource allocation. The average waiting time also shows similar trend for these parameters and hence is omitted.

4.9.3 Peak Latency

Figure 4.17 shows the peak latency for different packet size. The peak latency with configuration 1 is more for packet size of 2 than with packet size being 4. This is because with a packet size of 2 flits, the network with FIFO depth of 8 can load and transfer a larger number of packets. This increase in the number of packets also lead to longer queues. It can also be seen that with packet size of 8 flits, the peak

Figure 4.17: XY Routing: Peak Latency Vs Packet Size

latency for the configuration 2 is considerably higher than the value in configuration 3. This proves the adverse impact of longer queues as in case of average latency as well. The peak waiting time also has similar trend and therefore is not included.

4.10 Effect of configuration on Traffic Patterns

For this scenario, a 8x8 mesh is used with the 3 router buffer organization explained in Section 4.7. XY DOR is used with packet size of 4 flits and a packet injection interval of 16 cycles. Figure 4.18 shows the number of packets handled and Figure 4.19 shows the average latency values respectively for each configurations.

The effect of different buffer configurations is visible when the network congestion varies. It is parameters like packet injection interval and spatial distribution of the packets which cause congestion in the network rather than the traffic pattern itself. The random traffic pattern can change the spatial distribution of the packets and thereby affect the congestion even when the injection rate is fixed. So with random traffic pattern we obtain different values of total packets handled and average latency for configuration 2 and 3 as in Section 4.7. The static synthetic traffic patterns has

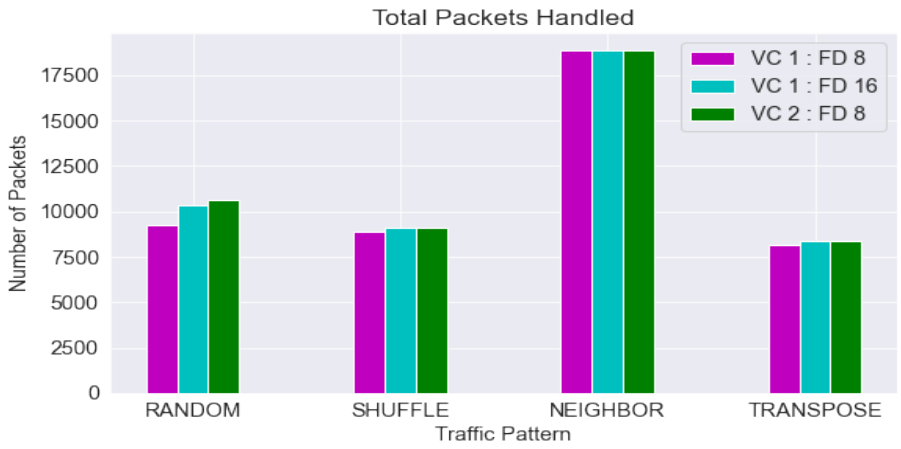

Figure 4.18: Buffer configuration: Packets Handled Vs Traffic Patterns

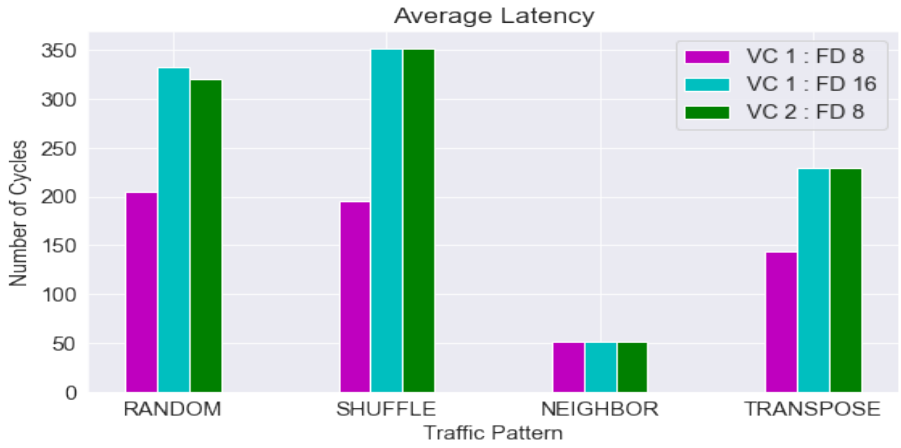

Figure 4.19: Buffer configuration: Average Latency Vs Traffic Patterns

fixed spatial distribution for packets and hence cannot affect the congestion if the packet injection interval is fixed. The packet injection interval of 16 cycles does not cause adequate congestion with static traffic patterns and as a result the number of packets handled and average latency remains the same for configuration 2 and 3. As before comparison of configurations 2 and 3 with configuration 1 is not appropriate as the traffic and the number of packets handled by them are entirely different.

Chapter 5

Congestion Aware X/Y Routing

In this chapter a new simple and sophisticated congestion aware X/Y routing algorithm is proposed. The performance and the effect of this new routing algorithm is checked on the previously build NoC simulator. Some features are added to the routers of NoC to facilitate congestion estimation as well as for adaptive routing. The additional features and functionalities of the router as well as the results of the proposed routing strategy are explained in this chapter. In addition, these results are compared against the ones obtained for conventional XY routing algorithm from the previous chapter.

5.1 Background

In conventional XY Routing, a packet is first traversed along the X direction and then along the Y direction. The disadvantage of this method is that if the packets already in queue inside the router are requesting for the same X port then the packet will have to wait for its turn as per the round robin arbitration scheme. This non

adaptive nature of XY routing leads to considerable loss in performance and may increase the overall latency. Adaptive routing strategies are implemented in order to overcome this limitation and to explore alternative paths to send packets through a network when under congestion.

Before explaining the new routing strategy, one should get familiarized with the request matrix structure used for the router design as shown in Figure 5.1.

In the request matrix, each row corresponds to one of the input ports and each column corresponds to an output port. The top row represents Input port 0 or North bound port and the subsequent rows below it represents the ports in order as explained in Figure 3.2 with the bottom row representing input port 4 or input local port. Similarly the left most column corresponds to Output port 0 or North bound port and each subsequent columns corresponds to other the output port in similar order with right most column for output port 4 or output local port.

For e.g. a 1 in the first row at 3rd column implies the packet waiting in the buffer of input port 0 is requesting the output port 2. Note that indexing of the ports begin from 0 and not 1 as C++ arrays are used for port implementation. Note that a row

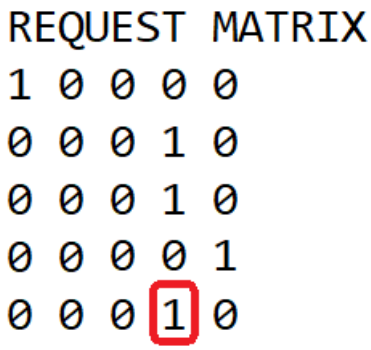

Figure 5.1: Request Matrix Structure

can only have a single 1 as an input port can make request for only one output port. When a column has more than one entry as 1, each request is granted as per the priority assigned by round robin arbiter inside the switch allocation block of router.

5.2 Proposed X/Y Routing Scheme

As seen from Figure 5.1, the red circled entry in the request matrix is being added to the already congested queue for output port 3. Say the arbiter priority for this out port is currently at output port number 1 which implies there is a long waiting time for the packet in buffer of input port 4 before it is granted. One way to mitigate this long waiting time is by avoiding the X dimension traversal when X ports are congested and looking for the Y traversal.
The proposed congestion aware X/Y routing scheme will simultaneously calculate both X and Y direction hops required to be made by a packet and will assign the request on to the output port which has less pending requests and congestion. But this method is applicable only for packets that have both X and Y hops. These packets are called reroute-able packets. For packets that has to do either X or Y hop alone to reach the destination will not have such a flexibility and shall remain in its own route. These are the non reroute-able packets.

5.2.1 Parameters of Importance

One method to ensure first in first out strategy is to implement the proposed X/Y Routing strategy as an oblivious routing where packets traverse different paths from source to destination without regard to the network congestion. But such implementation without giving regard to the congestion in the NoC may give rise to local hotspots and will degrade the performance. Thus oblivious nature of routing is not acceptable and so there are two factors that are to be considered before employing this routing scheme:

1. Congestion on Network
2. Number of Pending Requests for an Output Port

One can find that the adaptive nature of the proposed congestion aware X/Y routing

scheme is centered around the availability of free buffers in the adjacent routers and a more evenly distributed request matrix generation. Together this ensures a strong first in first out strategy for the routing of packets over XY DOR. Also it makes the proposed routing strategy a minimal adaptive routing. Minimal adaptive routing will take into consideration the network congestion and at the same time will take the minimal path to reach the destination. Thus there is hardly any energy wastage over the minimal DOR counterpart.

5.2.2 Simple Congestion Estimation

The occupied buffer slots in a router is a direct measure of the local congestion and therefore in order to quantify the congestion the concept of Buffer Occupancy Value (BOV) is used. The BOV is defined as the ratio of number of occupied buffer slots to the FIFO depth of a router. A BOV is calculated for the all four buffers of the directional ports using the flit_cnt variable which stores the number of flits present in that buffer. The BOV calculated can be compared against a threshold value and a Boolean signal can be used to indicate the buffer availability to the adjacent routers.

$$BOV\ of\ Input\ Port = \frac{Buffer\ Slots\ Occupied}{FIFO\_DEPTH * No\ of\ VC} \tag{5.2.1}$$

The Figure 5.2 shows the pin configuration for the congestion aware router IP block. Only two additional ports viz bov_in and bov_out, each consisting of 4 boolean pins are added to each router from the previous configuration3.4.

A BOV output port of 4 pins one pin for each direction and a similar BOV input port. So overall only 8 pins are increased over the former router architecture. The BOV output port is set to 1 when the flit_cnt of the corresponding buffer exceeds the threshold else it remains 0. The BOV input port is used to read the BOV output port of adjacent routers to assert the buffer availability of downstream router before proceeding with the routing algorithm. So if a pin in the bov_in port is read as 1

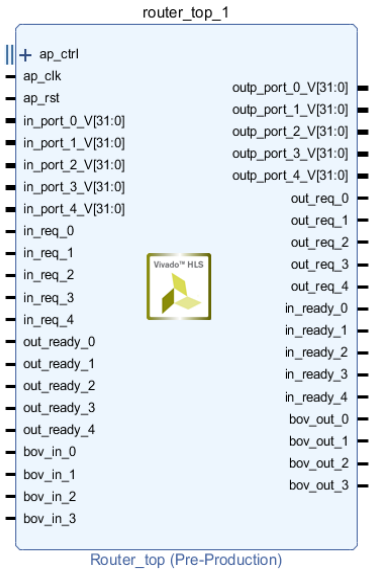

Figure 5.2: Congestion Aware Router IP Block

then it indicates a congestion for the output port in that corresponding direction. Setting the threshold is also important for congestion aware routing. For instance, threshold is set at 75% for FIFO depth of 16 (Configuration 2 Section 4.7) which is 12 flits. So when the flit_cnt of a buffer is greater than 12 then the BOV output pin is set to 1 and this information is used by the adjacent router along the direction of that pin for its route computation.

Another important parameter is the number of pending requests on an output port which adds to the waiting time and thereby the overall latency. So it is equally important to generate a request matrix which is more evenly distributed in the sense that it is necessary to avoid some output ports remaining idle while the other ports being congested.

5.2.3 Implementation

In the new route computation scheme, both the X direction port and Y direction port are computed simultaneously if applicable. If there is a column match or a row match, then only Y hop and X hop is possible respectively. These are the non-reroutable packets and so their requests are unaltered. But for the packets that has both X and Y hops or the reroutable packets, the number of pending requests on both the computed X and Y ports are counted.
Congestion on the X port of the downstream router could be due to two reasons:

1. Packets waiting in input buffers of next router.

2. Pending requests within current router having higher priority in arbitration.

In either case it is preferred to reroute the packets through the Y port if and only if the Y port is not congested. In the event of Y port also being congested, the X port is chosen to implement XY DOR. This will make the proposed routing to behave like XY DOR at high injection rate. If the number of requests on the X port is less such that it will not build up congestion in the downstream router and if the X port is not already congested, it shows low packet injection rate and therefore less congestion. For this case also X port is selected. So only at medium injection rate and less congestion the proposed routing will deviate from the conventional XY DOR behaviour.
In effect if there is no congestion and no chance for congestion build up in the buffers of X port for downstream router, then X port is chosen. The Y port is chosen if there is either of these occurring with Y port having free buffers. If the Y port buffers are also congested then X port is used. If the number of Y port requests is more then X port is used just like XY DOR.
It can be seen that using the concept of BOV and 8 boolean pins with minimal changes to the XY DOR routing algorithm, congestion awareness and adaptivity is added. Thus the hardware overhead required for the up gradation is also very low. The algorithm can be summed up as follows.

Algorithm 2: Congestion Aware Minimal Adaptive X/Y Routing

```
if No. of Requests for X-port > No. of Requests for Y-port then
    if (BOV_in for X-port = 0 ) AND (Pending requests for X-Port <
     Threshold) then
        Use X Port;
    else if BOV_in for Y-port = 0 then
        Use Y Port;
    else
        Use X Port;
else
    Use X Port;
end
```

5.3 Simulation Results

The performance of the proposed congestion aware X/Y routing scheme is compared against the conventional XY routing scheme using the designed Scalable Square Mesh NoC Simulator. The simulation methodology is also same as explained in Section 4.6. The effect of varying parameters like buffer configuration, BOV threshold and traffic patterns on the performance of the proposed routing scheme is explored. For each case the results obtained are compared against the results using conventional XY DOR scheme. It is seen that there is considerable performance improvement for the proposed congestion aware X/Y routing scheme.

5.4 Effect of VC and FIFO Depth

The performance of the proposed congestion aware X/Y routing is studied for the three configurations explained in Section 4.7. The performance metrics already described are used to estimate and compare the performance. To study the effect of

VC and FIFO depth, a 4x4 mesh is used with random traffic pattern and a packet size of 4 flits. The packet injection intervals are varied here.

5.4.1 VC 1: FIFO Depth 16

The Figures 5.3, 5.4 and 5.5 shows the comparison of the proposed congestion aware X/Y routing strategy against conventional XY DOR in terms of total packets handled, peak latency and peak waiting time respectively.

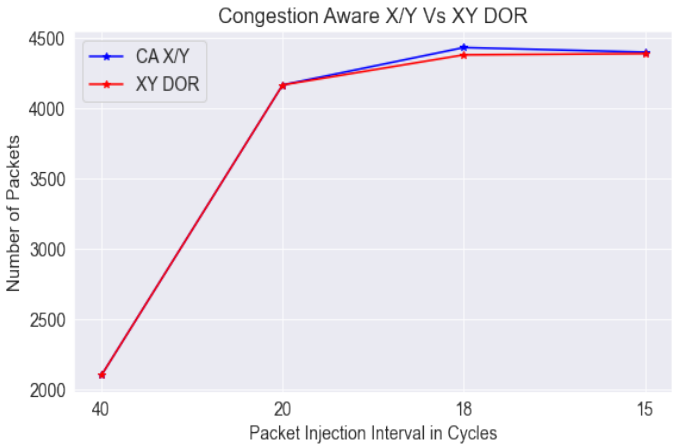

Figure 5.3: Comparison 4x4 Mesh: Total Packets Handled

The total number of packets that can be handled by the new algorithm is slightly better than XY routing under medium packet injection rate. For very low injection rate, this algorithm becomes same as the conventional XY routing scheme.

It is seen that the peak latency and waiting time for the packets can be reduced drastically under this configuration. When the packet injection interval is around 18 to 20, the network is lightly loaded and hence there are sufficient free buffers for rerouting the packets to reduce the waiting times. But as the injection interval is lowered, the free buffer availability will also decrease leading to smaller performance gain as is the case for packet injection interval of 15 cycles. The reason is the increasing congestion which results in lower free buffers.

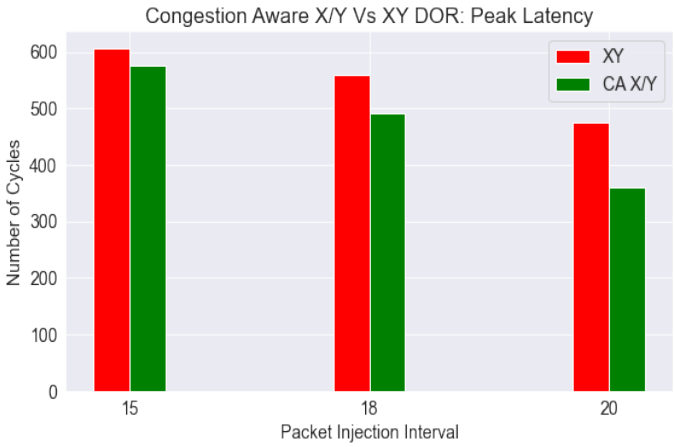

Figure 5.4: Comparison 4x4 Mesh: Peak Latency

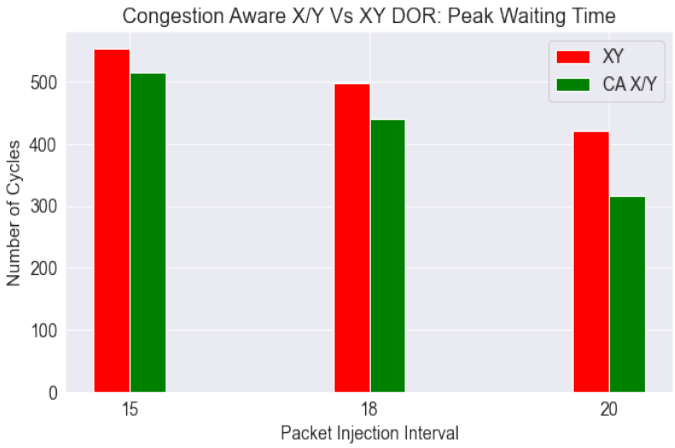

Figure 5.5: Comparison 4x4 Mesh: Peak Waiting Time

5.4.2 VC 1: FIFO Depth 8

As explained in the previous section, the proposed routing scheme works well when there are free buffers available in the network. For Store and Forward flow control with packet size of 4 flits, the free buffers available for this configuration is really low for higher packet injection rate. Even though the peak latency and waiting time can be reduced due to the strong first in first out nature of the proposed scheme,

it is not of major relevance as more packets cannot be handled by the network. So these performance graphs are not included here.

5.4.3 VC 2: FIFO Depth 8

The Figures 5.6 show the number of packets the proposed congestion aware X/Y routing strategy can handle against conventional XY DOR for configuration 3.

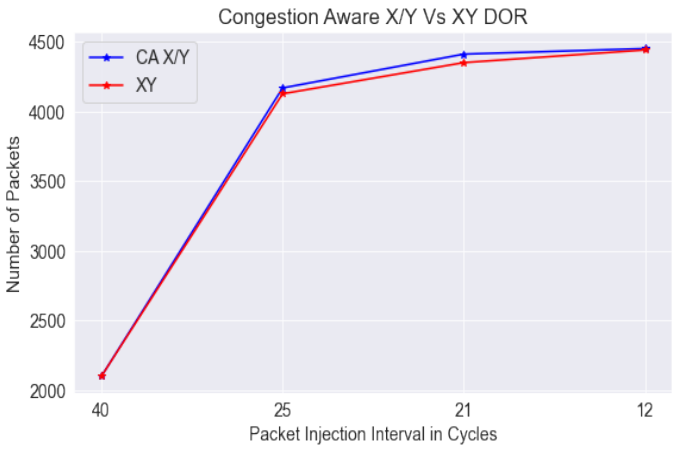

Figure 5.6: Comparison with VC: Total Packets Injected

It is seen that the total number of packets handled can be slightly improved using the proposed X/Y routing under medium packet injection rate. For very low and high injection rates, the proposed X/Y routing can handle equal number of packets as does the conventional XY routing scheme. The results of peak latency and waiting times are shown in Figure 5.7 and 5.8 respectively.

Here it can be seen at high packet injection rate or low injection interval, the peak latency and waiting time under the proposed routing has a minor degradation. This is partly due to the fact that the allocation of VC under congested traffic affecting the first in first out nature of the new algorithm and partly due to the nature of router operation with VC as explained in Section 4.7.3. At this high injection rate the number of packets handled by the network is also same as XY DOR. But for

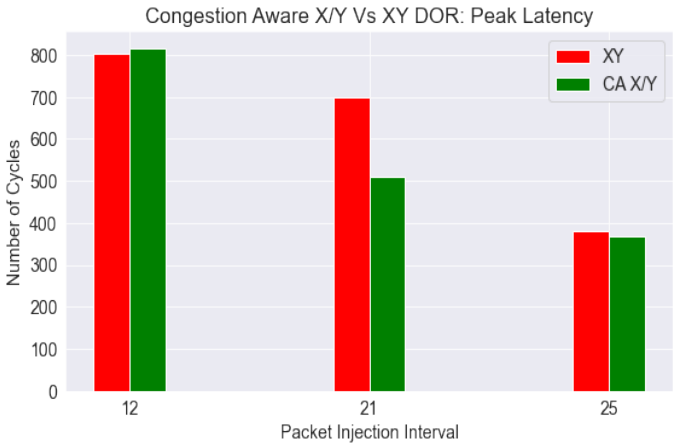

Figure 5.7: Comparison with VC: Peak Latency

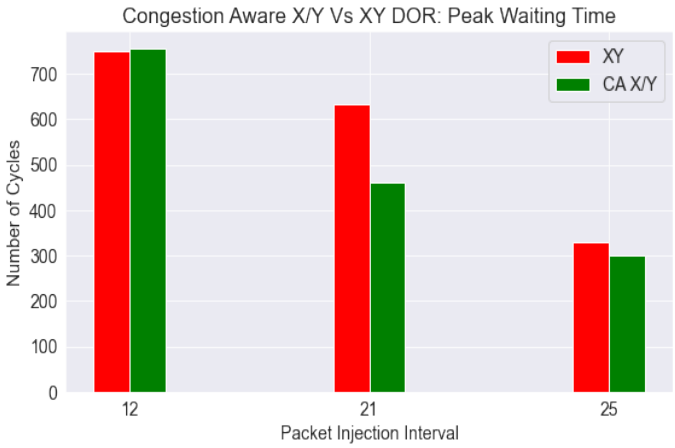

Figure 5.8: Comparison with VC: Peak Waiting Time

a medium injection rate which is for a packet injection interval of 21 cycles, the proposed routing scheme shows promising results. This is due to the same reason as explained in Section 5.4.1.

Also it can be seen under very low injection rate, the peak latency and waiting time are more similar to the values obtained under XY DOR because of less need for rerouting.

5.5 Effect of BOV Threshold

The BOV threshold is an important parameter which decides the buffer congestion and thereby affects performance of the proposed congestion aware X/Y routing. As explained in Section 5.2.2 the entire rerouting process occurs based on the incoming values at the BOV input pin from the adjacent routers.
For this simulation, a 4x4 mesh is used with router having FIFO depth of 16 flits. Packet injection interval is selected as 21 cycles which corresponds to medium traffic congestion. Random and Transpose traffic patterns are used and the results are compared against the conventional XY routing performance.

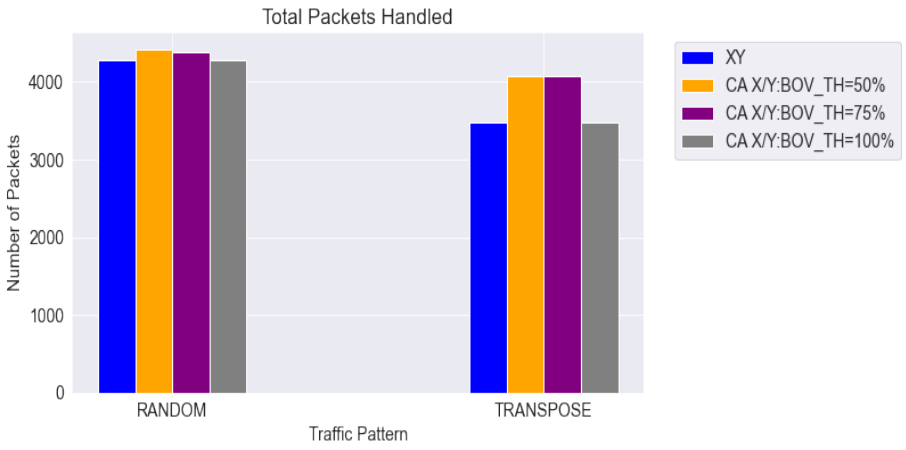

Figure 5.9: Effect of BOV Threshold: Total Packets Handled

The Figure 5.9 shows the total packets handled by the proposed congestion aware X/Y routing. For BOV threshold at 50% and 75% of the total FIFO depth which is 8 and 12 flits respectively in this case, the number of packets handled are higher for both traffic patterns. The transpose traffic seems to performance considerably better. Also it should be noted that if the threshold is set to 100% of FIFO depth (16 here), the proposed routing algorithm will not perform rerouting and hence behaves exactly like XY DOR.

Figure 5.10 shows the average latency incurred for different BOV threshold values. A threshold of 50% is good for random traffic but 75% gives more gains from

transpose traffic in terms of average latency.

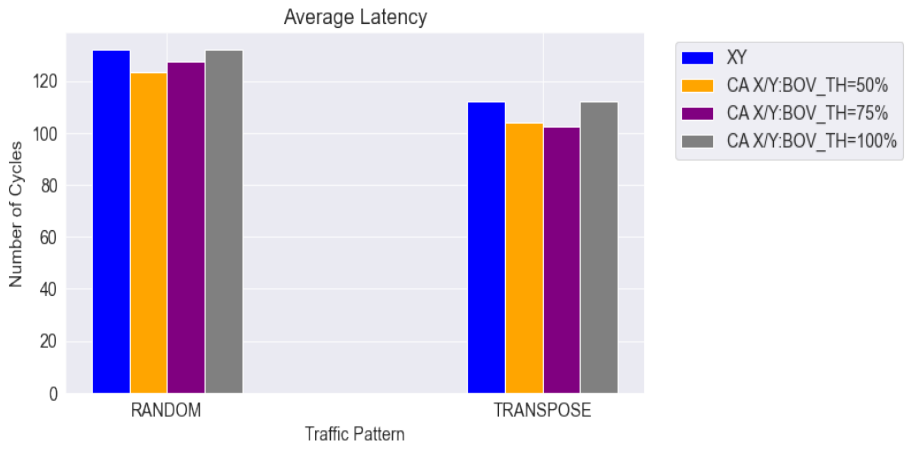

Figure 5.10: Effect of BOV Threshold: Average Latency

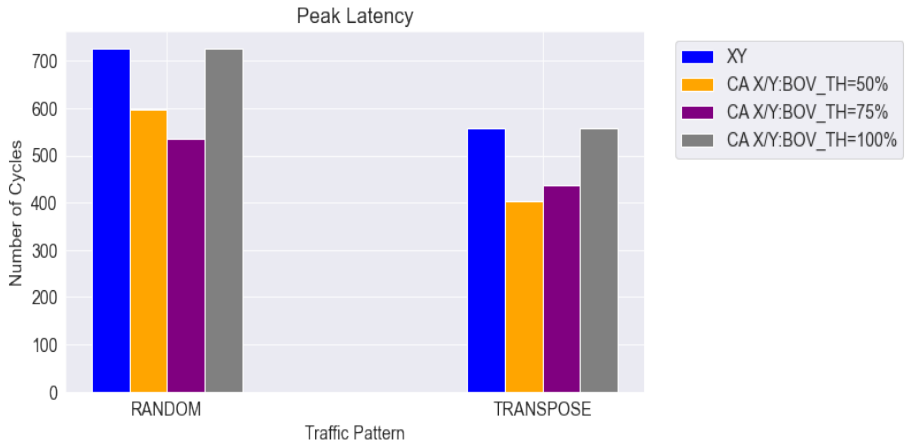

Figure 5.11: Effect of BOV Threshold: Peak Latency

Figure 5.11 shows the peak latency incurred for different BOV threshold values. Here it is seen that a threshold of 75% is good for random traffic but 50% gives more gains from transpose traffic. When threshold is 100% of FIFO depth, average and peak latency are same as with XY DOR.

In conclusion, the choice of threshold limit is crucial for setting the congestion.

A variable threshold value as per the traffic pattern of the current workload will give better performance gains. In general the threshold has to be between 50% and 100%. This section proves the dependency of the proposed X/Y routing on the free buffer availability in the network.

5.6 Performance with Synthetic Traffic Patterns

The performance of the proposed congestion aware X/Y routing with different traffic patterns like random, shuffle, neighbor and transpose is compared against the XY DOR counterpart. For these simulations, a 8x8 mesh is used with an injection interval of 15 cycles. For a FIFO depth of 16 flits with 1 VC and BOV threshold of 75% is used. The packet size used is 4 flits.

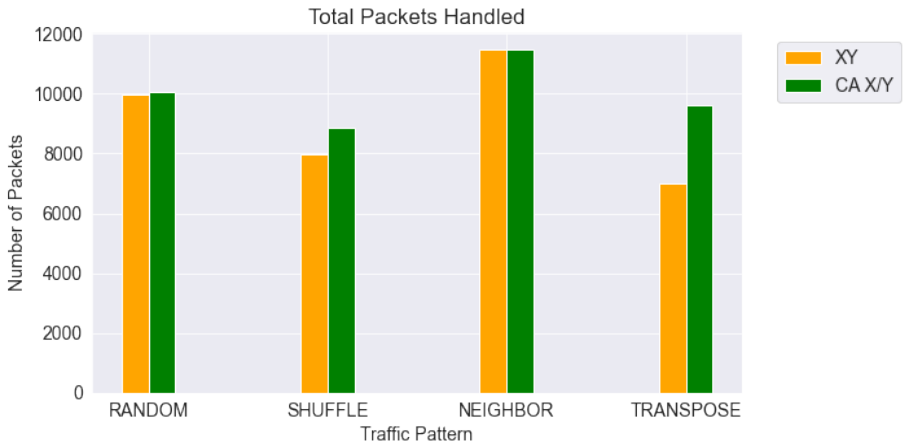

Figure 5.12: Comparing Effect of Traffic: Total Packets Handled

Figure 5.12 shows the total packets handled by the proposed routing against XY DOR. There is a slight increase in the number of packets handled for random traffic and no increase with neighbor traffic. The reasons are unbalanced traffic and less number of hops respectively. For shuffle traffic the increase is 11% and for transpose traffic the increase is 37% in terms of the total packets handled by the network.

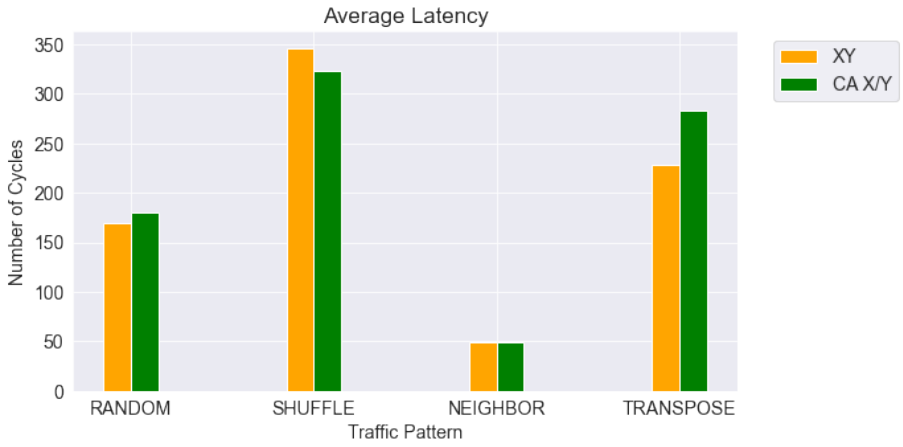

Figure 5.13: Comparing Effect of Traffic: Average Latency

Figure 5.13 shows comparison of average latency in each case. The average latency increases slightly for random traffic. For neighbor the proposed X/Y routing behaves exactly like XY DOR. With shuffle traffic pattern, the average latency reduced by 6% while with transpose traffic it increases by 24%. It should be noted that with latter, the number of packets handled by XY DOR was 7005 while it is 9615 with the proposed routing scheme. This huge increase in the number of packets will increase congestion in the network causing average latency to increase here.

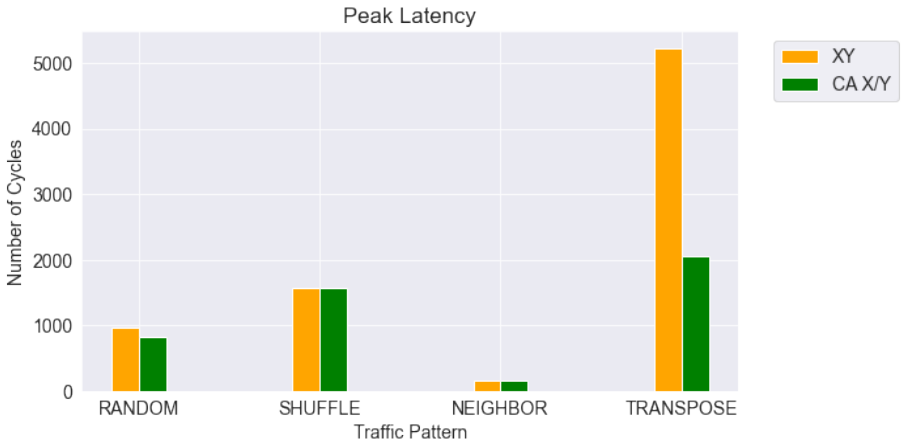

Figure 5.14: Comparing Effect of Traffic: Peak Latency

Figure 5.14 shows the comparison of peak latency with different traffic patterns. It is seen that the peak latency does not change for shuffle and neighbor traffic. For the former this is a positive indication as handling of more number of packets is not causing any adverse performance impacts. The reduction of peak latency seen for transpose traffic is enormously huge. The peak latency is reduced by a factor of 60% here. Even for random traffic the reduction in peak latency is 14%.

Comparing all the metrics the following conclusion is derived. For random traffic the performance gain in terms of number of packets is negligible but there is considerable reduction in peak latency. In case of shuffle traffic, the proposed XY routing method transfer considerably more packets at lower average latency and not affecting the peak latency. So in effect the performance gain for shuffle traffic using the proposed XY routing scheme is showing promising results. With neighbor traffic the proposed routing method is not effective and it behaves exactly like conventional XY DOR in terms of every metric.

The effect of proposed routing in case of transpose traffic is also a more promising result. Here the total number of packets handled is increased significantly and the reduction of peak latency is enormous. The considerable increase in the average latency is understandable because the proposed routing will push the NoC towards saturation which would otherwise operate at a lower congestion. But the performance gains clearly appears to dominate here.

5.7 Summary of Performance

The proposed congestion aware XY routing strategy is a simple congestion aware adaptive routing technique that takes into consideration the congestion as well as the pending output port requests. This method is an exploitation of the free buffers available in the adjacent routers and the idle output ports of the current router over the XY DOR method. So as in case of any exploitation there is an optimum level which gives maximum returns.

The performance of this algorithm depends upon the BOV of downside router to a greater extend. If there are sufficient free buffers on the NoC, the rerouting is effective and this strategy will give better results. Due to the strong first in first out nature of this algorithm, all performance metrics can be improved drastically for packet injection rates lower than the saturation point where the buffer availability is guaranteed. As for the peak latency and waiting time, it will come down drastically for packet injection rates lower than the saturation point. Under very low packet injection rate the proposed routing strategy behaves similar to conventional XY routing because there is hardly any need for rerouting. Also with high packet injection rates, there is less free buffers for rerouting and hence deadlock freedom at higher injection rates need to be studied. The traffic or packet injection rate is the key parameter in deciding the buffer occupancy in an NoC and therefore it plays a key role in the performance of this routing method as well.

Chapter 6

Conclusion and Future Scope

This work is a comprehensive study and implementation of a basic router architecture. Also design of a scalable square Mesh NoC simulator using the routers is presented. Extensive simulations are performed to verify the functionality of the designed simulator. Using different parameters like injection interval, buffer depth, traffic pattern and packet size the performance of 4x4, 6x6 and 8x8 mesh NoCs are evaluated.

Also a new simple local congestion detection mechanism based on BOV and a congestion aware minimal adaptive X/Y routing strategy is proposed. This new routing is implemented by upgrading the conventional XY Dimension Order Routing strategy to incorporate local congestion awareness and by switching between XY and YX Routing. We have established that the area overhead for the proposed routing is very low. The effect of packet injection rate and BOV threshold on the performance of the proposed routing is also evaluated. The proposed congestion aware adaptive routing shows peak latency reduction of 14% for random traffic and 60% for transpose traffic against the XY DOR at injection rates with medium congestion on the network. It is also shown that the proposed routing can handle and traverse more packets through the network. For shuffle traffic there is 11%

increase in the total number of packets handled with average latency reduced by a factor of 6% against XY DOR. With transpose traffic, the proposed routing method handles 37% more packets for injection rates not causing congestion. It is also shown that at very low and high injection rates, the proposed routing behaves exactly like the conventional XY DOR.

Our work can be improved in the future by adding more features to the simulator. The store and forward flow control method could be replaced with wormhole flow control. Scalability of number of Virtual Channels will give more flexibility to the simulator. The traffic patterns available can also be upgraded by including realistic traffic patterns like tornado. The design of the router is already optimized for synthesis using Vivado HLS. The C++ code for NoC can also be optimized for synthesis by the use of directives. As the design and implementation is using Vivado HLS, the synthesized NoC can be verified on FPGA to estimate real world performance.

References

[1] V. Raghunathan, M.B. Srivastava, and R.K. Gupta. A survey of techniques for energy efficient on-chip communication. In *Proceedings 2003. Design Automation Conference (IEEE Cat. No.03CH37451)*, pages 900–905, 2003. doi: 10.1145/775832.776059.

[2] Ng Yen Phing, M.N. Mohd Warip, Phaklen Ehkan, R Badlishah Ahmad, Fazrul Faiz Zakaria, and Farah Wahida Zulkefli. Towards high performance network-on-chip: A survey on enabling technologies, open issues and challenges. In *2016 3rd International Conference on Electronic Design (ICED)*, pages 259–263, 2016. doi: 10.1109/ICED.2016.7804649.

[3] Jose Duato, Sudhakar Yalamanchili, and Ni Lionel. *Interconnection Networks: An Engineering Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002. ISBN 1558608524.

[4] Natalie Enright Jerger, Tushar Krishna, and Li-Shiuan Peh. *On-Chip Networks: Second Edition*. Morgan & Claypool Publishers, 2nd edition, 2017. ISBN 1627059148.

[5] William Dally and Brian Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003. ISBN 0122007514.

[6] Tetala Neel Kamal Reddy, Ayas Kanta Swain, Jayant Kumar Singh, and Kamala Kanta Mahapatra. Performance assessment of different network-on-chip topologies. In *2014 2nd International Conference on Devices, Circuits and Systems (ICDCS)*, pages 1–5, 2014. doi: 10.1109/ICDCSyst.2014.6926188.

[7] Fawaz Alazemi, Arash AziziMazreah, Bella Bose, and Lizhong Chen. Routerless network-on-chip. In *2018 IEEE International Symposium on High Performance*

Computer Architecture (HPCA), pages 492–503, 2018. doi: 10.1109/HPCA.2018.00049.

[8] S Swapna, Ayas Kanta Swain, and Kamala Kanta Mahapatra. Design and analysis of five port router for network on chip. In *2012 Asia Pacific Conference on Postgraduate Research in Microelectronics and Electronics*, pages 51–55, 2012. doi: 10.1109/PrimeAsia.2012.6458626.

[9] Amit Bhanwala, Mayank Kumar, and Yogendera Kumar. Fpga based design of low power reconfigurable router for network on chip (noc). In *International Conference on Computing, Communication Automation*, pages 1320–1326, 2015. doi: 10.1109/CCAA.2015.7148581.

[10] Michael K. Papamichael and James C. Hoe. The connect network-on-chip generator. *Computer*, 48(12):72–79, 2015. doi: 10.1109/MC.2015.378.

[11] Feiyang Liu, Huaxi Gu, and Yintang Yang. Performance study of virtual-channel router for network-on-chip. In *2010 International Conference On Computer Design and Applications*, volume 5, pages V5–255–V5–259, 2010. doi: 10.1109/ICCDA.2010.5541185.

[12] Daniel U. Becker and William J. Dally. Allocator implementations for network-on-chip routers. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, pages 1–12, 2009. doi: 10.1145/1654059.1654112.

[13] Rajeev Kamal and Juan M. Moreno Arostegui. Rtl implementation and analysis of fixed priority, round robin, and matrix arbiters for the noc's routers. In *2016 International Conference on Computing, Communication and Automation (ICCCA)*, pages 1454–1459, 2016. doi: 10.1109/CCAA.2016.7813949.

[14] Santanu Chattopadhyay Santanu Kundu. *Network-on-Chip The Next Generation of System-on-Chip Integration*. CRC Press, 2015. ISBN 9781138749351.

[15] Uma R, Sarojadevi H., and Sanju V. Network-on-chip (noc) - routing techniques: A study and analysis. In *2019 Global Conference for Advancement in Technology (GCAT)*, pages 1–6, 2019. doi: 10.1109/GCAT47503.2019.8978403.

[16] Maurizio Palesi and Masoud Daneshtalab. *Routing Algorithms in Networks-on-Chip*. Springer Publishing Company, Incorporated, 2013. ISBN 1461482739.

[17] Ioannis Seitanidis Giorgos Dimitrakopoulos, Anastasios Psarras. *Microarchitecture of Network-on-Chip Routers - A Designer's Perspective*. Springer, New York, NY, 1st edition, 2015. ISBN 978-1-4614-4301-8.

[18] Pinar Kullu and Suleyman Tosun. Comparison of 2d mesh and reconfigurabl mesh topologies for network on chip design. In *2020 5th International Conference on Computer Science and Engineering (UBMK)*, pages 136–140, 2020. doi: 10.1109/UBMK50275.2020.9219423.

[19] Chen Chen, Zirui Tao, and Joshua San Miguel. Bufferless nocs with scheduled deflection routing. In *2020 14th IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, pages 1–6, 2020. doi: 10.1109/NOCS50636.2020.9241585.

[20] Jing Lin, Xiaola Lin, and Liang Tang. Making-a-stop: A new bufferless routing algorithm for on-chip network. *Journal of Parallel and Distributed Computing*, 72(4):515–524, 2012. ISSN 0743-7315. doi: https://doi.org/10.1016/j.jpdc.2012.01.001. URL `https://www.sciencedirect.com/science/article/pii/S0743731512000020`.

[21] Ling Wang, Xiaohang Wang, and Yadong Wang. An approximate bufferless network-on-chip. *IEEE Access*, 7:141516–141532, 2019. doi: 10.1109/ACCESS.2019.2943922.

[22] Amlan Ganguly, Kevin Chang, Partha Pratim Pande, Benjamin Belzer, and Alireza Nojeh. Performance evaluation of wireless networks on chip architectures. In *2009 10th International Symposium on Quality Electronic Design*, pages 350–355, 2009. doi: 10.1109/ISQED.2009.4810319.

[23] Shalimar Rasheed, Paul V. Gratz, Srinivas Shakkottai, and Jiang Hu. Storm: A simple traffic-optimized router microarchitecture for networks-on-chip. In *2014 Eighth IEEE/ACM International Symposium on Networks-on-Chip (NoCS)*, pages 176–177, 2014. doi: 10.1109/NOCS.2014.7008781.

[24] R. Mullins, A. West, and S. Moore. Low-latency virtual-channel routers for on-chip networks. In *Proceedings. 31st Annual International Symposium on Computer Architecture, 2004.*, pages 188–197, 2004. doi: 10.1109/ISCA.2004.1310774.

[25] Saad Mubeen and Shashi Kumar. Designing efficient source routing for mesh topology network on chip platforms. In *2010 13th Euromicro Conference on*

Digital System Design: Architectures, Methods and Tools, pages 181–188, 2010. doi: 10.1109/DSD.2010.57.

[26] Mohammad Behrouzian Nejad. Parametric evaluation of routing algorithms in network on chip architecture. *Computer Systems Science and Engineering*, 35(5):367–375, 2020. doi: 10.32604/csse.2020.35.367. URL `http://www.techscience.com/csse/v35n5/40510`.

[27] Martin Radetzki, Chaochao Feng, Xueqian Zhao, and Axel Jantsch. Methods for fault tolerance in networks-on-chip. *ACM Comput. Surv.*, 46(1), July 2013. ISSN 0360-0300. doi: 10.1145/2522968.2522976. URL `https://doi.org/10.1145/2522968.2522976`.

[28] Ayodeji Ireti Fasiku, Benson Olabode Ojedayo, and Oghenerukevwe Elohor Oyinloye. Effect of routing algorithm on wireless network-on-chip performance. In *2020 Second International Sustainability and Resilience Conference: Technology and Innovation in Building Designs(51154)*, pages 1–5, 2020. doi: 10.1109/IEEECONF51154.2020.9319964.

[29] Sayed T. Muhammad, Mohamed Saad, Ali A. El-Moursy, Magdy A. El-Moursy, and Hesham F.A. Hamed. Cfpa: Congestion aware, fault tolerant and process variation aware adaptive routing algorithm for asynchronous networks-on-chip. *Journal of Parallel and Distributed Computing*, 128:151–166, 2019. ISSN 0743-7315. doi: https://doi.org/10.1016/j.jpdc.2019.03.001. URL `https://www.sciencedirect.com/science/article/pii/S0743731518303794`.

[30] Zhenyu HU, Michael Conrad MEYER, Xin JIANG, and Takahiro WATANABE. Multiple factors congestion prediction algorithm for network-on-chip. In *2020 35th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC)*, pages 211–216, 2020.

[31] Zhengqian Han, Michael Conrad Meyer, Xin Jiang, and Takahiro Watanabe. Low-cost congestion detection mechanism for networks-on-chip. In *2019 IEEE 13th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*, pages 157–163, 2019. doi: 10.1109/MCSoC.2019.00030.

[32] Habib Chawki Touati and Fateh Boutekkouk. Reliable weighted globally congestion aware routing for network on chip. *Int. J. Embed. Real-Time Commun. Syst.*, 11(3):48–66, July 2020. ISSN 1947-3176. doi: 10.4018/IJERTCS.2020070103. URL `https://doi.org/10.4018/IJERTCS.2020070103`.

[33] Yaoying Luo, Michael Conrad Meyer, Xin Jiang, and Takahiro Watanabe. A hotspot-pattern-aware routing algorithm for networks-on-chip. In *2019 IEEE 13th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*, pages 229–235, 2019. doi: 10.1109/MCSoC.2019.00040.

[34] Aqib Javed, Jim Harkin, Liam McDaid, and Junxiu Liu. Minimising impact of local congestion in networks-on-chip performance by predicting buffer utilisation. In *2020 31st Irish Signals and Systems Conference (ISSC)*, pages 1–6, 2020. doi: 10.1109/ISSC49989.2020.9180165.

[35] Leonel Tedesco, Aline Mello, Diego Garibotti, Ney Calazans, and Fernando Moraes. Traffic generation and performance evaluation for mesh-based nocs. In *2005 18th Symposium on Integrated Circuits and Systems Design*, pages 184–189, 2005. doi: 10.1109/SBCCI.2005.4286854.

[36] Siying Xu, Michael Conrad Meyer, Xin Jiang, and Takahiro Watanabe. A traffic-robust routing algorithm for network-on-chip systems. In *2019 IEEE 13th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*, pages 209–216, 2019. doi: 10.1109/MCSoC.2019.00037.

[37] Nan Wang and Pedro Valencia. Traffic allocation: An efficient adaptive network-on-chip routing algorithm design. In *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, pages 2015–2019, 2016. doi: 10.1109/CompComm.2016.7925054.

[38] Ahmad M. Shafiee, Mehrdad Montazeri, and Mahdi Nikdast. An innovational intermittent algorithm in networks-on-chip (noc). *International Journal of Computer and Information Engineering*, 2(9):2907 – 2909, 2008. ISSN eISSN: 1307-6892. URL `https://publications.waset.org/vol/21`.

[39] Mohsen Nickray, Masood Dehyadgari, and Ali Afzali-kusha. Adaptive routing using context-aware agents for networks on chips. In *2009 4th International Design and Test Workshop (IDT)*, pages 1–6, 2009. doi: 10.1109/IDT.2009.5404108.

[40] Ran Manevich, Israel Cidon, Avinoam Kolodny, Isask'har Walter, and Shmuel Wimer. A cost effective centralized adaptive routing for networks-on-chip. In *2011 14th Euromicro Conference on Digital System Design*, pages 39–46, 2011. doi: 10.1109/DSD.2011.10.

[41] Wooshik Myung, Zhao Qi, and Ma Cheng. Performance analysis of routing algorithms in mesh based network on chip using booksim simulator. In *2019 IEEE International Conference of Intelligent Applied Systems on Engineering (ICIASE)*, pages 297–300, 2019. doi: 10.1109/ICIASE45644.2019.9074082.

[42] Declan O'Loughlin, Aedan Coffey, Frank Callaly, Darren Lyons, and Fearghal Morgan. Xilinx vivado high level synthesis: Case studies. In *25th IET Irish Signals Systems Conference 2014 and 2014 China-Ireland International Conference on Information and Communications Technologies (ISSC 2014/CIICT 2014)*, pages 352–356, 2014. doi: 10.1049/cp.2014.0713.

[43] Konstantinos Georgopoulos, Grigorios Chrysos, Pavlos Malakonakis, Antonis Nikitakis, Nikos Tampouratzis, Apostolos Dollas, Dionisios Pnevmatikatos, and Yannis Papaefstathiou. An evaluation of vivado hls for efficient system design. In *2016 International Symposium ELMAR*, pages 195–199, 2016. doi: 10.1109/ELMAR.2016.7731785.

[44] Tudor Gherman, Dorin Petreus, and Remus Teodorescu. A method for accelerating fpga based digital control of switched mode power supplies. In *2019 International Aegean Conference on Electrical Machines and Power Electronics (ACEMP) 2019 International Conference on Optimization of Electrical and Electronic Equipment (OPTIM)*, pages 322–328, 2019. doi: 10.1109/ACEMP-OPTIM44294.2019.9007156.

[45] Young-Kyu Choi and Jason Cong. Hlscope: High-level performance debugging for fpga designs. In *2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 125–128, 2017. doi: 10.1109/FCCM.2017.44.

[46] Mostafa W. Numan, Braden J. Phillips, Gavin S. Puddy, and Katrina Falkner. Towards automatic high-level code deployment on reconfigurable platforms: A survey of high-level synthesis tools and toolchains. *IEEE Access*, 8:174692–174722, 2020. doi: 10.1109/ACCESS.2020.3024098.

[47] *Vivado Design Suite User Guide - High-Level Synthesis*. Xilinx Inc., 2018.

[48] Wang Zhang, Ligang Hou, Jinhui Wang, Shuqin Geng, and Wuchen Wu. Comparison research between xy and odd-even routing algorithm of a 2-dimension 3x3 mesh topology network-on-chip. In *2009 WRI Global Congress on Intelligent Systems*, volume 3, pages 329–333, 2009. doi: 10.1109/GCIS.2009.110.

[49] Zhipeng Zhao and James C. Hoe. Using vivado-hls for structural design: A noc case study (abstract only). In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '17, page 289, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450343541. doi: 10.1145/3020078.3021772. URL https://doi.org/10.1145/3020078.3021772.

[50] Momil Ijaz, Huma Urooj, and Muhammad Athar Javed Sethi. Implementation of noc on fpga with area and power optimization. *EAI Endorsed Transactions on Context-aware Systems and Applications*, 6(16), 3 2019. doi: 10.4108/eai.23-5-2019.158953.

[51] Zhiliang Qian, Da-Cheng Juan, Paul Bogdan, Chi-Ying Tsui, Diana Marculescu, and Radu Marculescu. A comprehensive and accurate latency model for network-on-chip performance analysis. In *2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 323–328, 2014. doi: 10.1109/ASPDAC.2014.6742910.