Improving Packet Predictability of Scalable Network-on-Chip Designs without Priority Pre-emptive Arbitration

Bharath Sudev

Doctor of Philosophy

University of York
Computer Science

July 2015

Abstract

The quest for improving processing power and efficiency is spawning research into many-core systems with hundreds or thousands of cores. With communication being forecast as the foremost performance bottleneck, Network-on-Chips are the favoured communication infrastructure in the context mainly due to reasons like scalability and power efficiency. However, contention between non-preemptive NoC packets can result in variation in packet latencies thus potentially limiting the overall utilisation of the many-core system. Typical latency predictability enhancement techniques like Virtual Channels or Time Division Multiplexing are usually hardware expensive or non-scalable or both. This research explores the use of dynamic and scalable techniques in Network-on-Chip routers to improve packet predictability by countering Head-of-line blocking (blocked low priority packet blocking a high priority packet) and tailbacking (low priority packet utilising the link that is required by a high priority packet) of non-preemptive packets.

The Priority forwarding and tunnelling technique introduced is designed to detect Head-of-line blocking situations so that its internal arbitration parameters can be altered (by forwarding packet parameters down the line) to resolve such issues. The Selective packet splitting technique presented allows resolution of tailbacking by emulating the effect of preemption of packets (by splitting packets) by using a low overhead alternative that manipulates packets. Finally, the thesis presents an architecture that allows the routers to have a notion of timeliness in data packets thus enabling packet arbitration based on application-supplied priority and timeliness thus improving the quality of service given to lower priority packets. Furthermore, the techniques presented in the thesis do not require additional hardware with the increase in size of the NoC. This enables the techniques to be scalable, as the size of the NoC or the number of packet priorities the NoC has to handle does not affect the functionality and operation of the techniques.

Contents

List of Figures

Acknowledgements

I would like to thank my supervisor Dr. Leandro Soares Indrusiak for his guidance and encouragement throughout my research work. I am also thankful to Dr. Indrusiak for supporting my research through his EPSRC project LowPowNoC (EP/J003662/1) and EU FP7 project DREAMCLOUD (611411). I would like to thank my internal assessor Prof. Neil Audsly for his insightful suggestions and discussions during my research and for supporting me with his EU FP7 project T-CREST (288008).

I would then like to thank my parents; Dr. Sudev and Mohana and my brother Aravind for their consistent support.

I would also like to thank James (Dr. James Robert Harbin) for being a tolerant office mate humouring all of my weirdest ideas; discussing, filtering and refining the promising ones out and for proofreading my thesis. All of my research group members deserve special mention for providing a positive, friendly and resourceful atmosphere.

Finally, I would like to thank the friends I made in York during my Masters and Doctoral study; James, Piotr, Yunfeng, Antonio, Rosh and Silvia as well as my old friends Liju, Shanawaz and Shahin for their companionship.

Dedicated to

The Chief Designer
(1907-1966)

Author's Declaration

I declare that the work presented in this thesis is original except where otherwise stated or cited to a source. This work has not previously been presented for an award at this, or any other, University. Some of the work in this thesis has resulted in the following publications.

- PFT- A Low Overhead Predictability Enhancement Technique for Non-Preemptive NoCs, B. Sudev and L. S. Indrusiak, 21st IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC), Istanbul-Turkey, October 2013.

- Low Overhead Predictability Enhancement in Non- preemptive Network-On-Chip Routers using Priority Forwarded Packet Splitting, B. Sudev and L. S. Indrusiak, 9th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), Montpellier-France, May 2014 **(Best Paper Award).**

- Predictability Enhancement in Non-preemptive NoCs using Selective Packet Splitting, B. Sudev and L. S. Indrusiak, 12th IEEE International Conference on Industrial Informatics (INDIN), Porto Alegre-Brazil, July 2014.

- Dynamic Time Multiplexed Virtual Channels, a Performance Scalable Approach in Network-On-Chip Routers to Reduce Packet Starvation, B. Sudev and L. S. Indrusiak, 7th York Doctoral Symposium on Computer Science and Electronics (YDS), York-UK, Oct 2014 **(Best Presentation Award).**

- Network-on-Chip Packet Prioritisation based on Instantaneous Slack Awareness, B. Sudev, L. S. Indrusiak and J. Harbin, 13th IEEE International Conference on Industrial Informatics (INDIN), Cambridge-UK , July 2015.

Chapter 1

Introduction

The advent of integrated circuits around the early 1960s brought about a quantum leap for computing, both in terms of magnitude of computation as well as in the envelope of its applications [1]. As the quest for improving processing power forced more and more transistors onto chips, the quest for a wider range of applications forced integration of different types of components onto chips [2] [3] [4]. To get the best performance out of the chips, the number of transistors was increased and their size continually reduced [5][6]. The clocking frequency was also increased to improve performance to power utilisation ratios. Although this shrinking reduced dynamic power dissipation and intrinsic latencies of the individual modules, it had an adverse effect on the inter-modular communications as those were traditionally implemented using shared buses or point-to-point connections [7]. The reduction in wire dimensions resulted in higher resistances and reduced wire spacing inducing capacitance, delays and crosstalk and subsequently imposed practical limitations in performance [8].

Compared to long connection lines, short lines with repeaters performed better for multiprocessor communication, which lead to the development of the Network-on-Chip (NoC) concept [9] [10]. As far as manufacturing is concerned, gates cost commercially less than wires [7].

Application convergence brought about a diversity of signals onto a single platform, which favoured a heterogeneous communication infrastructure like NoCs over customised ones like buses [11]. NoCs were also surpassing buses in devel-

opment time as Time-To-Market for NoCs were considerably less due to the extensive use of synthesizable Register Transfer Level (RTL) based approaches rather than manual layout. Superiority of NoCs over buses and point-to-point connections is due to many factors like efficiency, reliability, scalability, reusability and cost effectiveness [7].

Compared to traditional bus based networks, NoCs have lower capacitive load per transmission due to its shorter wire lengths, ultimately resulting in lower dynamic power consumption [12]. As a result, NoCs emerged as a promising communication infrastructure for the communication centric designs that would enable many-core systems [13] [14]. As Moraes et al stated in [15] "An NoC is an on-chip network composed by cores connected to switches, which are in turn connected among themselves by communication channels". As NoCs consist of a network of routers communicating with each other using data packets, a wide variety of topologies can be utilised.

Regardless of the topology used, contention between packets intensified by the multi hop nature of communication in NoCs typically introduces uncertainty into the system. As NoC packets compete for arbitration traversing multiple routers to get to the destination, packet latencies can vary. With applications in which the workload is static and known beforehand, static analysis can be used to determine suitable packet priorities and to assign the application tasks to particular cores (task mapping) [16]. However, in open applications where traffic pattern is dynamic and cannot be predicted, increased latency could be encountered for packets regardless of its Quality of Service (QoS) requirements due to contention and blocking from other packets. Similarly in heterogeneous applications in which packets from known applications may have to coexist with dynamic traffic, similar decrease in performance can occur [17].

For example, in parallelised video processing applications, there would be transmissions needing high QoS while others can tolerate lowering of QoS intermittently [18]. For such systems which deal with dynamic traffic that needs multiple levels of QoS, the NoC should have appropriate infrastructure to deal with conten-

tion between packets so that the required functionality of the design can be maintained [19].

For resolving uncertainty with dynamic traffic, designers employ several strategies like multiplexing link utilisation of packets in time or space domain, providing separate logical channels or by employing adaptive approaches.

Even though the classical time multiplexed approach [20] ensures complete predictability of the system, it comes at the overhead of restriction in scalability and dynamic behaviour. Though dynamic, multiplexing in the space domain (bandwidth) [21] and the use of separate channels [22] would result in excessive hardware requirements. Considering packet predictability (QoS) as the reduction in the variability in packet latency [23], this research aims to improve packet predictability by using dynamic, scalable and lightweight methods.

For example, if a low priority packet is utilising a connection link that is required by a higher priority packet (blocking), the high priority packet will suffer an increase in the magnitude and variation of its latency. With time multiplexed approaches, the functionality of the routers are multiplexed in time so that such scenarios will never occur. However time multiplexed routers would not be able to handle traffic which is not known in advance (dynamic traffic) unless the routers are reconfigured to handle those. As a result, time multiplexing can limit the router's ability to handle dynamic traffic and they have limitation in scalability.

Such blocking scenarios can be resolved by multiplexing in space domain; by providing separate logical channels. This is achieved by classifying packets into several service levels and then by providing separate buffering for each service level. As a result, this would result in significant increase in hardware overhead in terms of both logic as well as buffering [24].

This thesis explores the use of scalable dynamic techniques in simple non-preemptive routers that modify arbitration policies and packets to counter unpredictability in NoC packet latencies. To resolve blocking of packets by other lower priority packets, the techniques presented aim at modifying the internal parameters of routers and packets to improve QoS of packets depending on its priority.

The thesis also looks into introducing a timeliness element in arbitration decisions so that packets will get preference over others not only based on its application supplied priority value, but also based on its timeliness.

For example, consider the situation where there is a high priority packet which is well ahead in time (with respect to its expected reception time) contenting for arbitration with a low priority packet which is late in time (with respect to its expected reception time). By utilising a timeliness element in arbitration decisions, routers would be able to identify such scenarios so that the lower priority packet will get better QoS if the competing high priority packet can afford to be delayed.

1.1. Thesis Hypothesis

The hypothesis addressed in this thesis is that "**Latency predictability can be enhanced in scalable non-preemptive NoC designs using modifications that dynamically alter arbitration policies or packet structure**".

The thesis addresses packet predictability in non-preemptive NoCs as NoCs employing preemptive arbitration for predictability enhancement are hardware intensive and have scalability limitations. To verify this hypothesis, this research introduces techniques that allow routers to alter its arbitration policies and packets and evaluates their effectiveness using Hardware Description Language coded models.

1.2. Thesis Structure

The thesis has a three tier structure.

Tier 1 – Introductory chapters

Chapter 1: **Introduction**

Chapter 2: **Literature review**

Chapter 3: **Metrics and problem statement**

Tier 2 – Key techniques and evaluation

Chapter 4: **Starvation resolution by priority manipulation**

Chapter 5: **Predictability enhancement by packet splitting**

Chapter 6: **Predictability enhancement through dynamic slack awareness**

Tier 3 –Supporting chapters

Chapter 7: **Conclusion**

Glossary of terms

References

Appendix

Tier 1 chapters comprise the introductory section of the thesis. Chapter 2 provides a literature review in which the basics of NoCs are discussed followed by specifics on existing predictability enhancement techniques. As case studies for the application of such techniques, some of the popular NoC architectures are discussed next, followed by details on how prototyping is done for such systems. This acts as a prologue to the later part of Chapter 3; where the implementation methodologies in this thesis will be introduced.

Chapter 3 presents the metrics used in the thesis and present some experimentation results that depict the variability in latency encountered by non-preemptive NoC packets compared to pre-emption based designs. Chapter 3 will then continue with the problem statement followed by the details on the implementation and test infrastructure used.

Tier 2 chapters describe the techniques proposed in this thesis. Chapter 4 introduces the technique that use priority manipulation to resolve packet starvation (resulting in publication [25]). Rather than using multiplexing techniques in space or time, the technique features routers that exchange blocking information so that its internal parameters will be modified to resolve blocking. The later part of the

chapter presents the specific architectural details of the model and performance analysis.

In Chapter 5, the technique that utilises packet splitting to improve predictability is presented (resulting in publication [26]). The technique introduced is aimed at emulating pre-emption functionality (by splitting packets) without the high hardware overheads associated with the classical pre-emption approach. As the technique uses splitting of packets rather than the typical pre-emption functionality, the routers can be simpler and scalable. The chapter will continue with the details on the hybrid design that employ the technique complemented by the technique introduced in Chapter 4 (resulting in publication [27]). With the technique employed, NoC designs could be scalable and dynamic still providing quality of service in end-to-end latency without major hardware overheads compared to the pre-emption approach.

Chapter 6 introduces a scalable technique that will enable routers to have a notion of timeliness in arbitration decisions (resulting in publication [28]). This will allow routers to initiate predictability enhancement measures not just based on the priority of the packet (application-supplied priority) but also based on its timeliness. This is achieved by introducing a dynamic field in the packet header to represent the timeliness component which would be modified by routers when the packet waits for arbitration. For arbitration decisions, the routers employ this value combined with the application supplied priority value. This would allow NoC routers to improve end-to-end latency of lower priority packets when higher priority packets have residual slack (earliness compared to its expected reception time) to spare for dynamic traffic. As a practical application for the system, the notion of residual slack is used to trigger the use of the techniques introduced in Chapter 4 and 5 hence improving their effectiveness.

Finally, Tier 3 contains the Conclusion as Chapter 7 which includes thesis summary, details of novel contributions and further work, followed by the Glossary of Terms, References and Appendix as subsequent chapters.

Chapter 2

Literature Review

The initial part of this chapter will cover the basics of NoCs as a prologue to the following section that details the classical predictability enhancement techniques. While predictability enhancement techniques like Time Division Multiplexing and Link Division Multiplexing employ division of operational time and available bandwidth respectively, other techniques cover multi-channel approaches as well as adaptive routing. As a demonstration of the practical use of these techniques, the chapter will then discuss some of the NoC architectures that use such techniques.

The final part of the chapter deals with the prototyping techniques that can be employed for NoCs. This will act as a preface to the later part of Chapter 3 in which the implementation methodologies used in this research will be explained.

2.1. Network-on-Chip Basics

A NoC consist of Intellectual Properties (IPs) or cores connected to routers which are interconnected between each other using connection links as shown in Figure 2.1 (where a 3x3 2D mesh type NoC is shown). The routers and links act as the communication infrastructure for the IPs and to receive and send data, each router will have input and output ports.

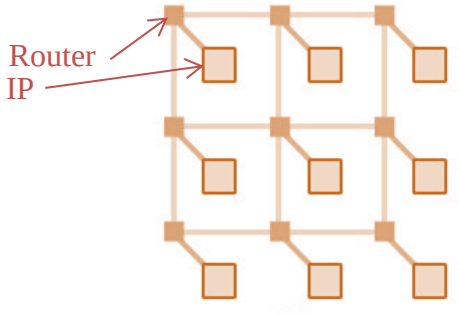


Figure 2.1: Mesh type NoC

In a typical NoC router designed for 2D mesh topology (as in Figure 2.1), there will be five pairs of input and output ports with one connected to the local IP and the other four to neighbouring routers as shown in Figure 2.2.

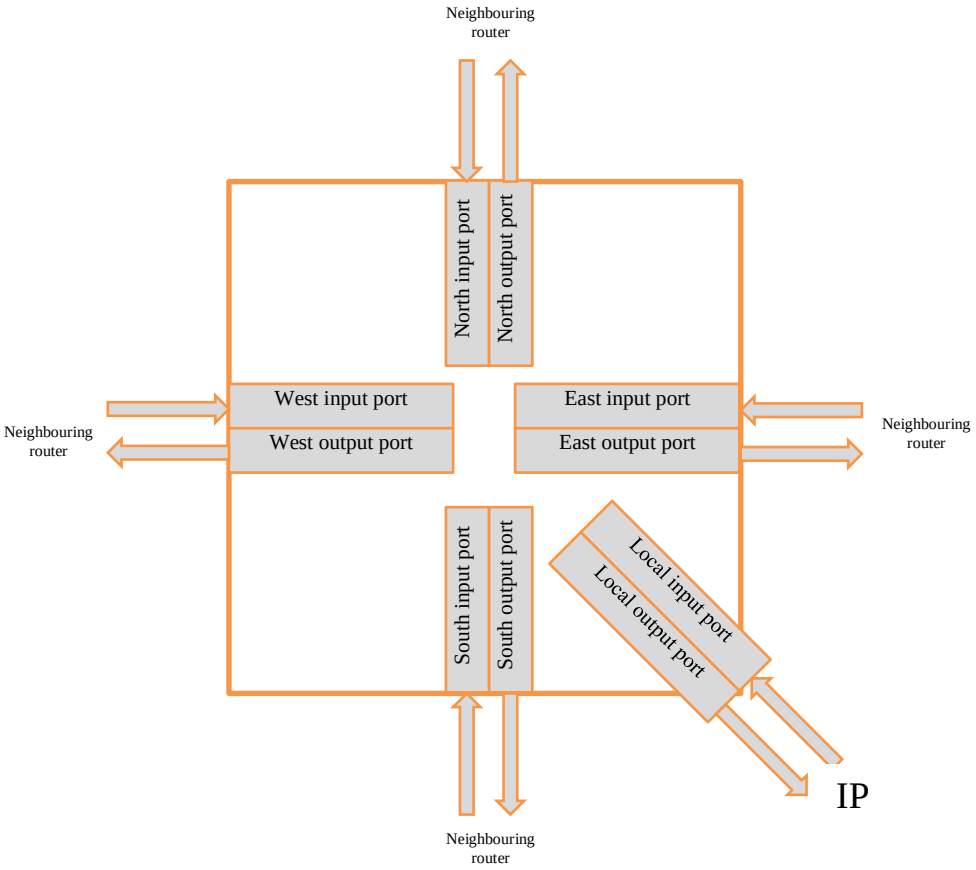


Figure 2.2: NoC router

Communication between NoC routers are done with flits (flow control digits) which is the smallest unit of flow control maintained by the NoC [29]. It can be

seen as the set of bits that can be transmitted through a connection line in a single clock cycle and hence the maximum bit width of a flit will be the bit width of the communication channel. The data from the application layer is converted into flits (for communication) by the Network Interface that acts as the link between the IP and the NoC router. In NoCs that feature packet based communication, multiple flits are grouped into data packets, which usually will have a header part holding information about the characteristics of the packet followed by a payload part that contains the transmission data. For example, if the connection links are 8 bits wide, the flits will be 8 bits wide (if the whole bandwidth is used) as shown in Figure 2.3a. As seen in Figure 2.3b, several flits are grouped into a data packet and typically the packet will have a header comprising of one or more flits followed by a set of payload flits which hold the actual data.

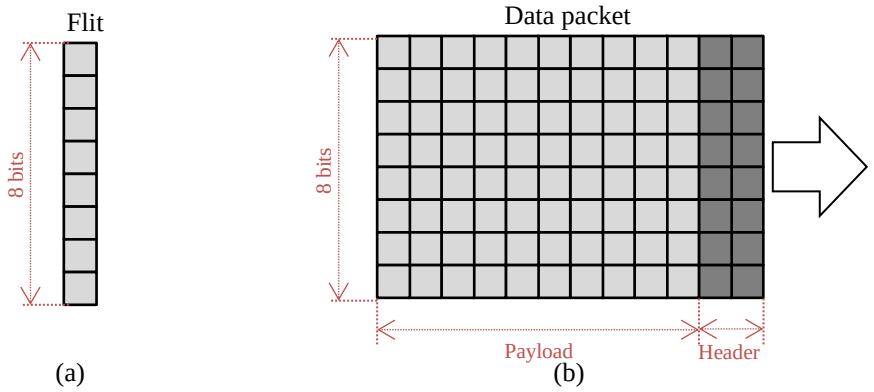


Figure 2.3: (a) NoC flit (b) NoC packet

To form a NoC, routers can be interconnected in either uniform or non-uniform topology. Typical NoC designs use uniform topologies, as non-uniform topologies require complex routing considerations and have scalability issues as the routing logic will have to be customised with variation in size of the NoC.

2.1.1. Switching Techniques

Switching techniques specifies how the data traverses from its source IP to its destination IP. Depending on the type of communication, two types of switching techniques can be employed in NoCs; Circuit switching and Packet switching [30].

For communication that involves streaming of data without limitation in maximum message length, Circuit switching technique can be used. With Circuit switching [31] [32], for an IP to transmit data to another IP, the transmission path will be reserved before transmission and the path will be held until the transmission is complete. As the path is entirely reserved for the communication, the technique features low latency data transfer (once a path is reserved) but as a whole, the NoC will suffer from several issues like blocking and low overall network utilisation especially under contention [11] [30].

The terms packet latency blocking, network utilisation and contention are defined as follows.

> ***Packet latency:*** *The time interval between time instant when the network interface of the source core is supposed to inject the header flit of the packet to the instant when the whole of the packet is received by the network interface of the destination core in simulation ticks.*
>
> ***Blocking:*** *A communication is said to be blocked by another when the communication path needed for the former is being utilised by the later communication thus preventing its transmission.*
>
> ***Network utilisation:*** *Network utilisation is defined as the percentage of total number of connection links being used for communication at any point of time.*
>
> ***Contention:*** *Contention is defined as the situation when two or more communication flows require transmission through the same connection link.*

Circuit switching can be significantly efficient in conditions where transmission time is considerably higher than the setup time (infrequent long messages) as the whole path is reserved beforehand.

With Packet switching, packets are sent from source to destination without reserving paths and they independently negotiate its path through the network. The routers employing the Packet switching technique called Wormhole switching [33] are

designed to send a flit as soon as it can be accommodated by the next router. Wormhole switching technique can be seen in a wide range of NoC architectures including Hermes [15] and AEthereal [34] primarily due to the lower memory requirements than the other approaches [35].

There are also other Packet switching techniques like Store And Forward (SAF) [36] and Virtual Cut Through (VCT) [30]. In SAF, a packet is sent to the next router only if the next router has buffer space to accommodate it completely, and the receiving router starts further transmission only when the whole packet is received. The advantage of SAF is that a packet will only block other packets inside a single router at any time unlike wormhole switching where a packets can block packets simultaneously in more than one router. As this technique has high memory requirements in buffering [11] [30], it is not widely used in NoC architectures (however it is used in Nostrum [10] [11] NoC).

Though similar to SAF, VCT is intended to reduce latency by enabling the router to forward the packet as soon as there is space to accommodate the whole packet in the next router's buffer. VCT supports lower latency packet transmission than SAF but it has similar memory requirements to SAF.

With VCT and SAF, the maximum packet size possible depends on the size of the buffers in the NoC. This would prevent transmission of packets that are bigger than the buffer size in routers, which is not an issue with Wormhole switching.

2.1.2. Routing Algorithm

Another design choice in NoCs is the routing algorithm which determines through where the packets will be routed through the network. The choice of routing algorithm presents a trade-off between factors like power, logic area, delays, and robustness [35]. Typically implemented as algorithms or as lookup tables, routing can be done in three methods; Source, Distributed and Centralized [39].

In source routing, the route for data transport is calculated at the sender and that information is added into the header of the packet before starting transmission. The routers down the line determine the path of that packet through the NoC using

this route information and hence the router design can be simpler than the distributed approach.

In distributed routing, routing decisions are made by each router individually by evaluating the destination information carried in the packet header. However, with centralised routing, routing decisions are made by a centralised module and the information is communicated to the appropriate router on need. This can result in increased latency to initiate routing as routing messages will have to be transmitted to and from the centralised routing module to the router. As a result, this will affect the latency performance of the system (in magnitude and variability) negatively and the use of the centralised module will limit scalability.

In the thesis, the term scalability is defined as follows.

> ***Scalability:*** *The ability of the NoC router to handle packets with a wider range of priority values thus enabling the use of the router in bigger NoC topologies than it was initially designed for.*

With source routed packets; as the path of transmission is calculated ahead of transmission, adding dynamic predictability enhancement behaviours will involve adding complicated logic to recalculate routing path which will result in both increased overhead as well as latency (due to the additional route recalculations).

2.1.3. Arbitration

Routers also require arbitration logic to deal with contention between data packets. There are several arbitration techniques used in NoCs like Round Robin, First Come First Served, Priority Based and Priority Based Round Robin [40] [41] [42].

Typically, Round Robin and First Come First Serve arbitration are employed for NoCs aiming at best effort service. Priority Based and Priority Based Round Robin are used in NoCs aimed at providing guaranteed service as the approaches use packet priority parameter for arbitration.

2.2. Packet Predictability in NoCs

As in [23] and [43], packet predictability can be considered as the reduction in variation in packet latency and it is a key design parameter [44] as far as NoCs are concerned. With NoC based communication systems that deal with dynamic traffic, there could be variation in packet latency regardless of packet priority [17]. Murali et.al. in [45] states that "designing an interconnect architecture with predictable behaviour is essential for proper system operation" and Huang et al in [22] reports a steep drop in throughput of links in NoCs without predictability enhancement features. Without predictability enhancement measures, packets could have high variation in latency [46] which would increase the probability of missing their deadlines. The term deadline is defined in the thesis as follows.

> ***Deadline:*** *The desired bound on packet latency in simulation ticks.*

If the traffic flow pattern is known in advance, static analysis aided task mapping or time division based approaches can be used to ensure that the packets will meet their hard deadlines. The term hard deadline is defined as follows.

> ***Hard deadline:*** *The latency deadline of a packet, missing which can result in a catastrophic failure of the design target.*

However in situations where the traffic is not known beforehand, ensuring a hard deadline is not possible. Even with priority based arbitration, the packets cannot be set with a soft deadline due to unforeseeable contention scenarios in the NoC. The term soft deadline is defined as follows.

> ***Soft deadline:*** *The latency deadline of a packet, missing which may result in performance degradation of the design target and would not cause a catastrophic failure of the design target.*

This could be unacceptable with applications that have traffic with different latency requirements.

For instance, in non-preemptive NoCs, high priority packets could fail to secure arbitration due to Head-of-line (HOL) blocking [47] which is defined as follows.

> ***Head-of-line blocking:*** *A packet is said to be Head-of-line blocked when it is blocked by a lower priority packet which is already blocked.*

Quoting Huang et al from [22] "due to HOL blocking, the throughput of the links is typically limited to 58% under uniform traffic with fixed packet length" (derived from [62]). To provide an idea on the uncertainty, a scenario is depicted in Figure 2.4 where squares represent NoC routers and arrows represent packets flows with the number inside the circles depicting the packet priority. In the scenario, all packets have destination south of the router (1,2).

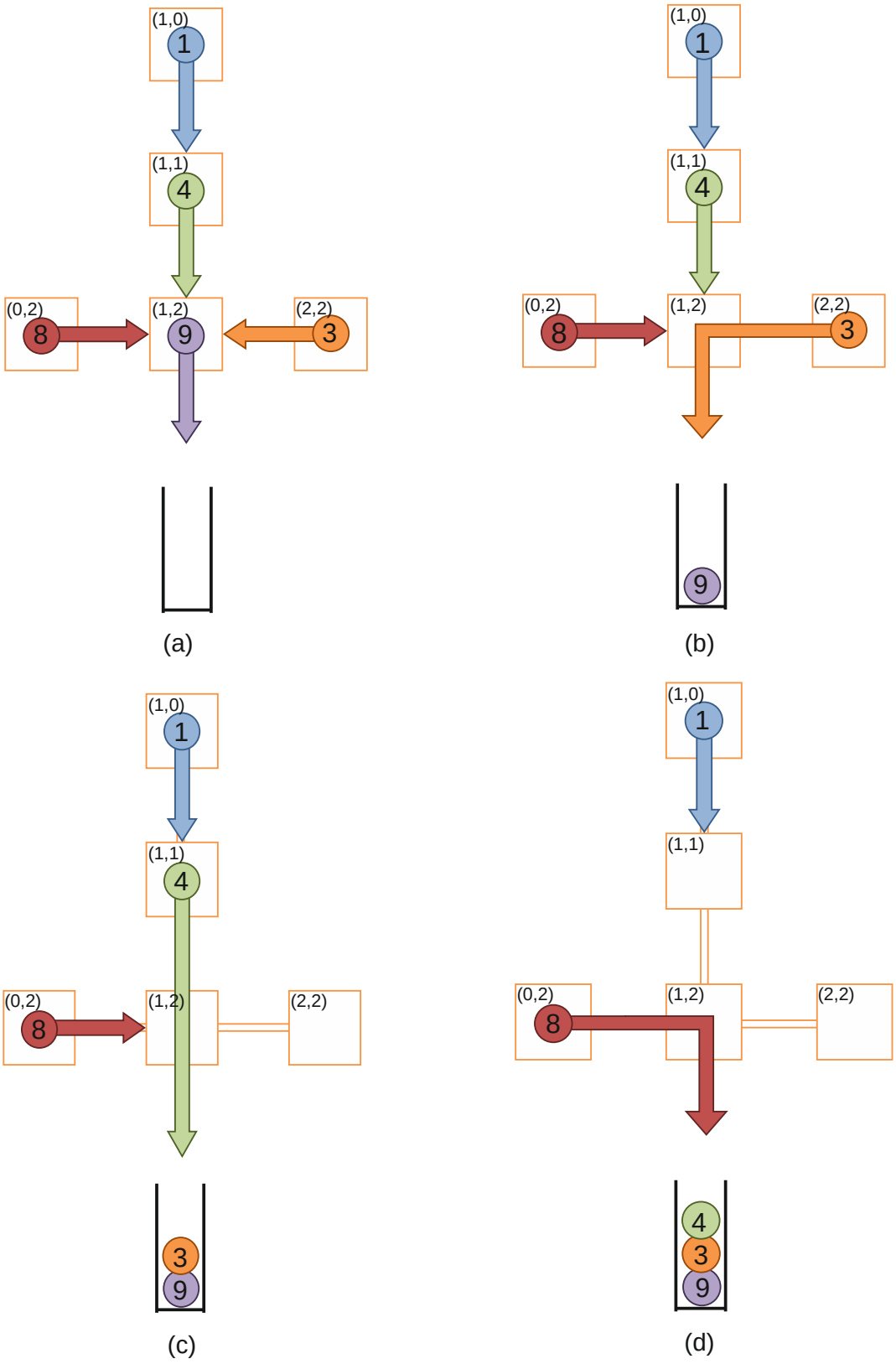


Figure 2.4: Head-of-line blocking example

In Figure 2.4a, it can be observed that packets 3, 4 and 8 are withheld from securing arbitration as packet 9 is tailbacking them by utilising the south port of router (1,2). The term tailbacking is defined as follows.

> ***Tailbacking:*** *A packet is said to be tailbacked when the link required for its transmission is being utilised by a lower priority packet.*

As packet 9 is of very low priority, it could be blocked down the line by many packets; hence indefinitely blocking higher priority packets like 1, 3, 4 and 8 up the line despite their higher priorities. After packet 9 gets transmitted and releases the path, the issue elevates further as packet 3 will get arbitration ahead of packet 4 forcing packet 1 to wait further up the line as depicted in Figure 2.4b. Here packet 4 is HOL blocking packet 1 thus preventing its transmission.

When packet 3 finishes transmission, packet 4 will be transmitted followed by packet 8 ahead of 1 (Figure 2.4c and Figure 2.4d) unless the routers are designed to provide arbitration to packets in a single clock cycle.

As a result, despite the highest priority value possible, packet 1 will have to wait until all the other packets get transmitted. Since all the other packets are susceptible to further blocking down the line due to their lower priority values, packet 1 is susceptible to have further waiting stages which could worsen its latency. Thus, under an ordinary situation, the final transmission order of router (1,2)'s south port will be 9-3-4-8-1 (8 before 1 if arbitration in routers take more than a clock cycle) which goes against the application-level priority assignment.

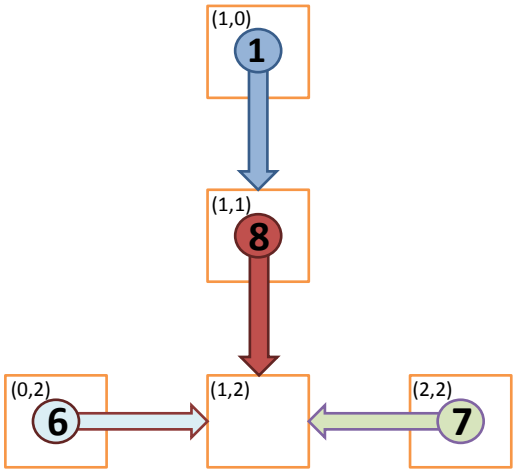


Figure 2.5: Starvation example

Consider the situation in Figure 2.5 where the destination for the packets is south of router (1,2) and the transmission periods of packets 6 and 7 are short compared to their packet sizes. Packet period is defined as below.

> ***Packet period:*** *Packet period is defined in the thesis as the interval in simulation ticks between successive injection of packets into the NoC from an IP.*

Under this situation, packet 6 will secure arbitration first followed by packet 7 and since the period of packet 6 is short, it will again request arbitration at router (1,2) before packet 7 is transmitted completely. As a result, packet 6 will secure arbitration after 7 is transmitted. As packet 7 also has a short period, it will again request arbitration to the link before packet 6 is transmitted completely and hence get arbitration to the link after packet 6 is transmitted. As a result the south port of router (1,2) will be used by packet 6 and 7 over and over and hence packet 8 will never get arbitration. Since packet 1 is behind packet 8, it will never get arbitration as well despite possessing the highest priority possible.

To resolve such predictability-degrading issues and hence reduce variation in packet latencies, several predictability enhancement techniques can be employed. The rest of the section considers the contemporary predictability enhancement techniques used in NoCs. As unpredictability is caused by contention between

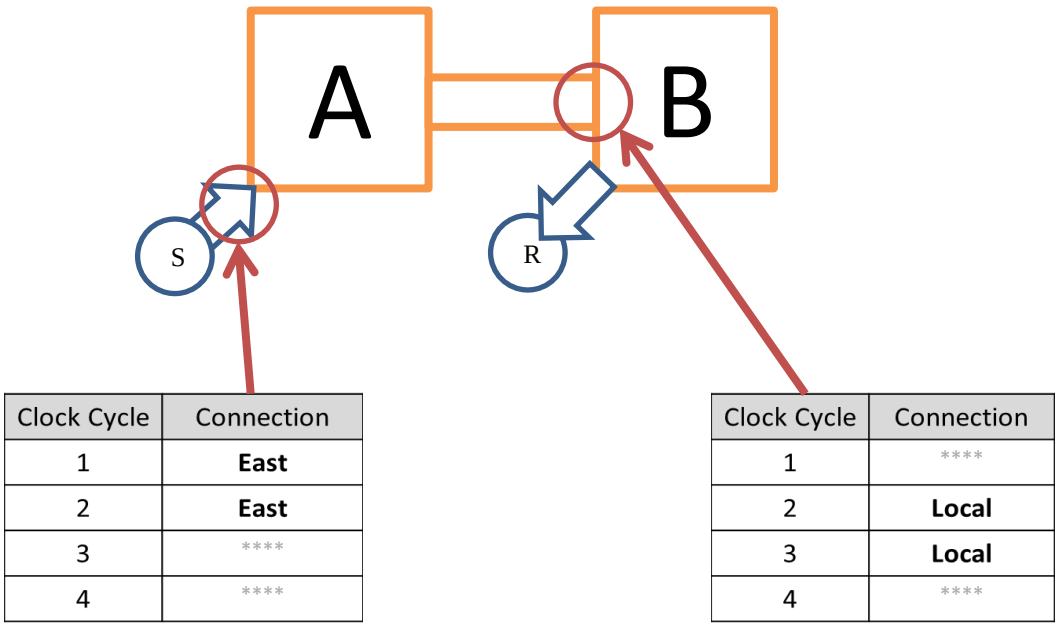
packets, the techniques explained here exploit a variety of methods which multiplex, preempt or divert competing packets to ensure better predictability than trivial systems.

2.2.1. Time Division Multiplexing

Time Division Multiplexing (TDM) [49] is one of the classical methods to ensure predictability in NoCs. With TDM, the functionality of the router is multiplexed in the time domain thus providing utmost predictability. Use of TDM can allow packets to meet their hard deadlines as the functionality of the routers is defined in the time domain.

TDM routers work based on slot tables that dictate every input port what to do at each clock cycle. To understand the technique in detail, an example is shown in Figure 2.6 where router A is sending flits from its IP to the IP of router B. An example slot allocation for the slot table of the local port (sender) of router A and the west port (receiver) of router B is provided in the figure. As the example does not consider the functionality of the other ports, the slot tables governing those are not added in the figure.

From the slot table of the local port of router A, it is evident that it is configured to send a flit through the east port at the first clock cycle towards router B. On the next clock cycle when the flit will reach router B, the west port of router B will forward it to the local IP connected to router B as its slot table dictates so.



Clock Cycle	Connection
1	**East**
2	**East**
3	****
4	****

Clock Cycle	Connection
1	****
2	**Local**
3	**Local**
4	****

Figure 2.6: Time Division Multiplexing functionality example

Simultaneously, router A will sent the next flit (on the second clock cycle) which will be received on router B on the third clock cycle which will then be routed to its local IP.

Even though TDM systems are highly predictable and simple, they are not scalable and have restricted dynamic behaviour. The word dynamic behaviour is defined in the thesis as follows.

> ***Dynamic behaviour:*** *The ability of the router to respond in run time to incoming packets (regardless of its destination) without reconfiguration to the routing logic.*

As the NoC work based on slot tables, for adapting the router for bigger NoC sizes the slot table will have to be made bigger (to service the resultant increase in packet flow numbers) hence becoming its scalability limiting factor.

Furthermore, to account for any new packet flows, the slot tables of all associated routers will have to be modified thus highly limiting its dynamic behaviour [50]. As slot allocation calculation is a complex and time consuming procedure [51], TDM based routers have limited application in dynamic traffic scenarios.

2.2.2. Link Division Multiplexing

While TDM work by sharing link access to packets in the time domain, Link Division Multiplexing (LDM) [52] work by sharing the access of sections of the link itself. With LDM, multiple packets can be transmitted simultaneously by designating sections of the connection link for each packet thus enabling communication even under blocking. In Figure 2.7, a possible use of LDM technique is demonstrated where the bandwidth of the connection link is shared in part by packets A, B and C simultaneously.

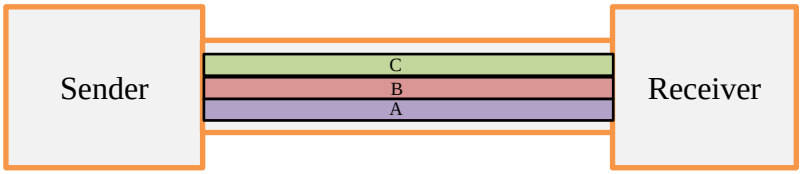


Figure 2.7: Link Division Multiplexing

This will allow better utilisation of the links compared to TDM but it will be hardware expensive [21]. As shown in Figure 2.8, the technique relies on serialising and de-serialising logic (with buffering) to allow conversion of flits into less wider format and back along with a control mechanism. As a result, LDM implementation results in high hardware requirement both in terms of logic complexity and buffer requirement.

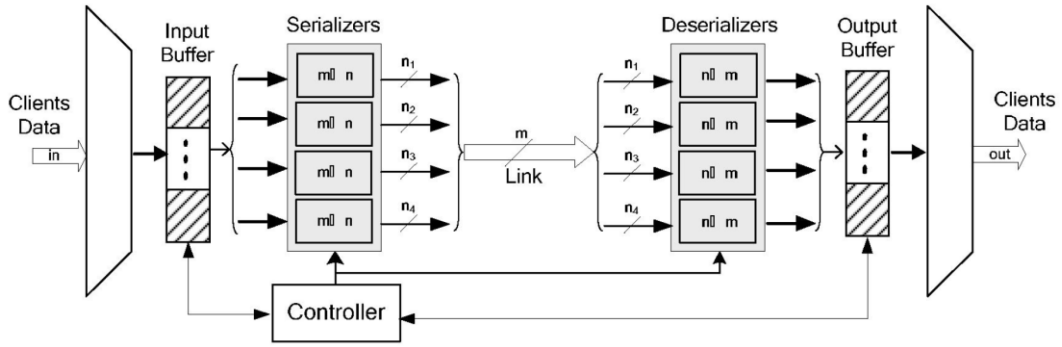


Figure 2.8: LDM implementation (taken from [21])

Furthermore, the increase in size of the NoC would result in sharing of the bandwidth between higher numbers of packets and hence as the size of the NoC increases, the effectiveness of the LDM will decrease thus limiting scalability.

2.2.3. Virtual Channels

With Virtual Channels (VCs) [53] a physical connection path is multiplexed into separate logical channels so that multiple packets (already arbitrated) can be made to use the same path. Introduced by Dally in [54], the Virtual Channel technique relies on the use of multiple buffers for each channel on the network so that communication through a link will be possible even with blocked flits.

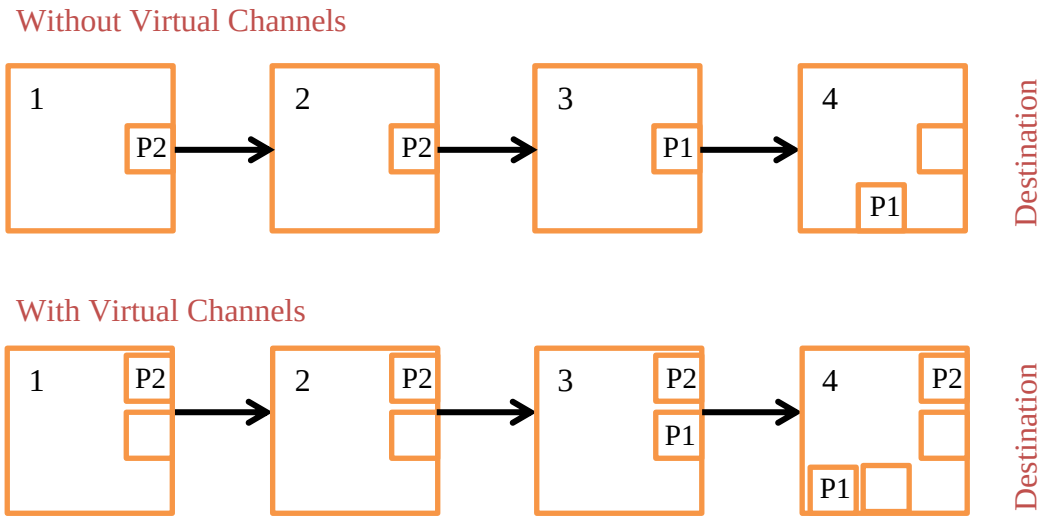


Figure 2.9: Blocking example with and without Virtual Channels [54]

For example, consider the example in Figure 2.9, where there are four routers and two packets P1 and P2. It can be seen that without the use of Virtual Channels, flits of P2 are blocked behind flits of P1 when P1 is blocked somewhere down the line (as P1 and P2 share the path through router 3).

With the use of Virtual Channels (Figure 2.9), it can be seen that P2 flits will be able to reach the destination even if the shared port has blocked packets as Virtual Channels use separate buffers for each channel.

To prevent the upstream routers from sending flits if the packet is blocked downstream (hence buffers fully occupied), routers also have credit based flow control mechanism for each VC. As a result if the buffer in the receiving router is full (of that VC), the transmitting router will stop transmission (of flits of that service level) and allow transmission of flits from lower priority VCs if applicable.

Therefore hypothetically, a two Virtual Channel router will act as two routers stacked on top of each other (one router for each Virtual Channel) sharing the same communication link as shown in Figure 2.10.

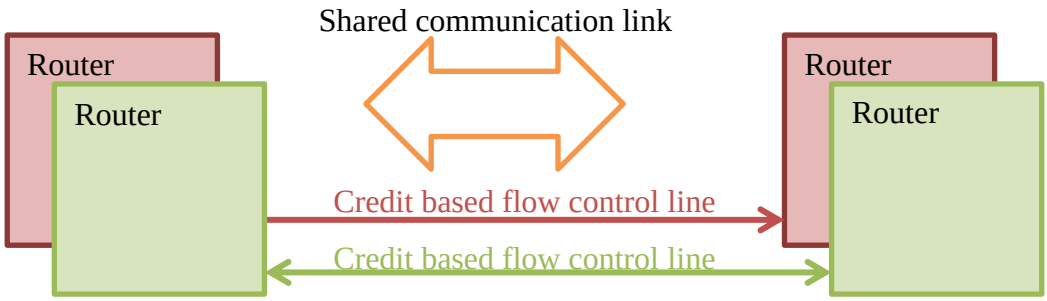


Figure 2.10: Virtual Channel functionality

As the link is being shared, Virtual Channels will have priority assignments in between each other which will enable the system to prioritise a Virtual Channel over the other under contention.

Even so, VCs would not be able to ensure packets meeting their hard deadlines as there can be packets of the same service level competing for arbitration. However, the use of VCs can increase the probability of packets meeting their soft deadline.

Mello et al [24] compared performance of a Hermes NoC with and without Virtual Channels and their tests report reduction of average latency of more than 50% for the 8x8 NoC under test. Although VCs provide significant performance improvements, VC implementation results in significant hardware overhead. As a result, their tests with designs housing 1, 2 and 4 VCs resulted in hardware overhead of 17%, 33% and 75% respectively on their target platform.

2.2.4. Adaptive Routing

Adaptive routing approaches aim to improve packet predictability by dynamically varying the routing by monitoring the traffic pattern in the NoC. Ge et al. in [55] utilised a centralised monitoring module in their design to alter the source routing depending on the traffic on the NoC.

Traffic pattern is defined in the thesis as follows.

> ***Traffic pattern:*** *The pattern of traffic flow through the routers over the whole NoC over the entire simulation run (Traffic pattern consists of all the packet flows through the NoC, each specifying parameters like source-destination information, packet priority, injection time and packet size)*

Under non-congested state, the system follow source routing (simple, low latency) and the routing table of each router is initialised in advance and broadcasted to all routers.

If a link fails or gets congested, the adjacent router informs the central monitor about the issue and after receiving the overall network condition, the monitor calculates optimal alternate paths using Dijkstra's shortest path algorithm [56] and routing tables are updated. To decrease latency further, while the tables are being updated, the system has an in built deadlock/live-lock free routing logic using partially adaptive XY algorithm [57] to forward the packets past the congested or faulty links.

With partially adaptive XY algorithm, when congestion occurs, the router will evaluate neighbouring router's load status using dedicated lines to alter the routing to pass the packets through a lightly loaded path. The major drawback of the system is that the techniques aim at spreading load rather than resolving predictability degrading issues. As a result, if the NoC is evenly loaded, the advantages brought about will be limited.

Cidon et al in [58] presents an adaptive routing architecture which employs Traffic Load Maps (TLMs) to store the congestion info so that the source routing algorithm can be altered according to the traffic load pattern.

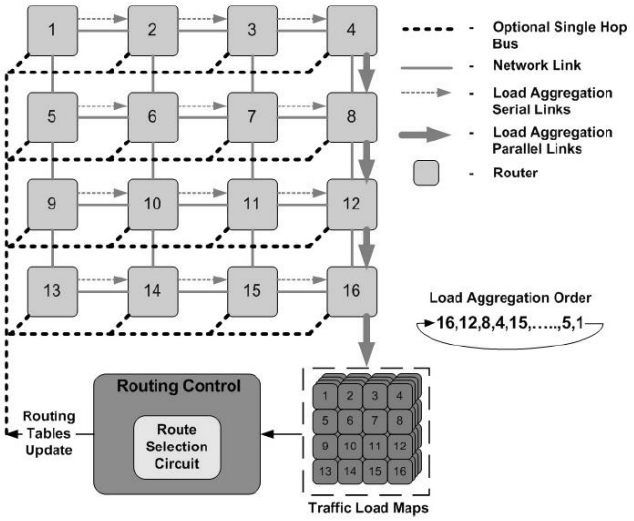


Figure 2.11: Approach based on Traffic Load Map (taken from [58])

This system has monitors embedded in the routers to monitor the network load using any of the metrics like buffer occupation, Virtual Channel usage etc. However, due to the use of traffic maps, the scalability of the system is limited and similar to the previous technique; the efficiency of the system could decrease with the overall increase in load in the NoC.

Rantala et al. in [59] dealt with adaptability in a distributed perspective where the source routing at each network interface was altered depending on the congestion information retrieved from neighbouring routers. The design had monitor modules connected to each router and the monitor modules were in turn connected in between each other. The monitor modules were designed to check the load situation at the respective router (using metrics like number of packet flows or buffer utilisation) and communicate with the neighbouring routers so that the routing can be altered using adaptive XY-routing method.

The work mentions two approaches of load estimation; router state based and FIFO status based.

In router state based monitoring system, each router is provided with dedicated monitoring devices to monitor its switching activity along with activity of its neighbours in the mesh as shown in Figure 2.12.

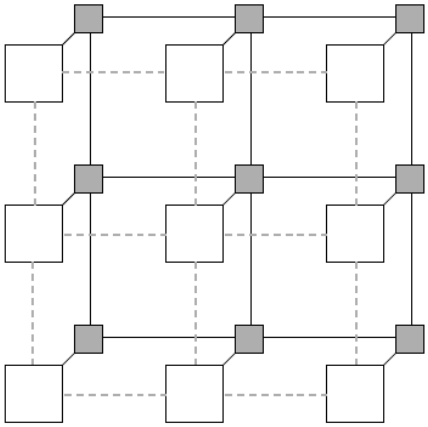


Figure 2.12: Router status based monitoring (taken from [59])

As shown in the figure, the monitor modules (white squares) are interconnected by 2-bit lines (thin dotted lines) so that the status information can be interchanged between neighbouring routers (routers shown as grey squares) while rest of the network follows a typical mesh topology.

The FIFO status based monitoring approach however used output FIFO occupation as the network monitoring metric. Converse to the router status based approach which monitors switching activity, FIFO based approach focusses on link utilization between routers hence provides a more detailed information on traffic levels than the first approach.

Nevertheless, with the increase in load in the network, the efficiency of the technique will decline as the techniques rely on diverting flows and not intended to resolve predictability degrading issues. Also, on an evenly loaded NoC, the advantages brought about by adaptive routing is limited and as a result, the research presented in the thesis refrains from using adaptive routing.

2.2.5. Timeliness in NoCs

Timeliness is a parameter along with application-supplied priority that can be used to improve QoS of data packets. Timeliness (or the notion of time) is typically introduced in data packets with time stamping as seen in [60] and [61]. As time stamping requires the notion of a global time thus requiring long counters, its use in NoC routers is limited.

There have been approaches that aimed at improving packet predictability by introducing a time element without depending on the notion of a global time. For example, Das et al in [62] presented a slack aware system where the packet header will include the priority value which consisted of both its packet priority and acceptable slack. The slack value in the system was static and was based on parameters such as the number of hops or maximum latency level. The approach was focussed on dealing with computational delays and hence it did not take into account the time spent by packets waiting in NoC routers for arbitration.

Andreasson et al in [63] presented an approach which relied on using slack (or unused slots) on TDM based systems for improving network utilisation. With this approach, the TDM based functionality of the router made the notion of timeliness in packets unnecessary but as with the classical TDM approach, it limited its scalability and dynamic behaviour.

Similarly, Diemer et al in [64] depicted a back suction based flow control which was used to improve Best Effort (BE) service latency by utilising the free bandwidth available with their Guaranteed Service (GS) infrastructure. The work portrays a router architecture with VCs where a number of VCs are allocated permanently for GS and the rest for BE traffic. The system allowed downstream routers to notify upstream routers of low activity in the BE service (using dedicated connection lines) by evaluating the buffer utilisation in the router. This will allow upstream routers to prioritise BE service VCs momentarily which otherwise will have to wait.

Berejuck et al in [65] presented a system in VC based NoCs to improve QoS by targeting ageing of packets. In the work, the packets were added with fields in

their headers that will be incremented as packets wait for arbitration. This value in the field is then utilised by the arbitrator when packets of same service levels compete for arbitration as the packets did not have any priority field.

Similarly, Correa et al in [44] presents a NoC framework that allows the routers to increase packet priority when a packet waits for arbitration for certain number of clock cycles. However, under high load condition, there is possibility of multiple packets acquiring highest priorities thus compromising the predictability of the high priority spectrum of packets. The design also features dropping of low priority packets if they fail their deadlines to ease congestion. This is achieved by utilising a notion of global time and deadlines, which requires significantly higher hardware resources.

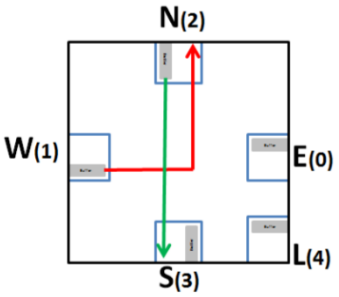
2.3. NoC Architectures

This section describes several case studies of NoC architectures. The initial part of Section 2.3 considers the simple Hermes [15] NoC developed by Moraes et al, following which the Virtual Channel based QNoC [66] developed by Bolotin et al is presented. The final part of this section presents the AEthereal [34] NoC developed by Goossens et al which utilises TDM followed by a review of other NoC architectures that employ a combination of techniques as well as others.

2.3.1. Hermes

Hermes [67] is a simple NoC architecture which provides low hardware overhead communication through its distributed routing scheme. Used typically in uniform topologies, each Hermes router has five input buffered bi-directional ports (one connected to the local IP and the rest to neighbouring routers) and a control logic module. To confine the hardware overhead to a minimum, Hermes employs wormhole switching with a configurable flit size. The first and second flits of a Hermes packet are the headers, which contain the target address and the number of flits in the entire package respectively.

A Hermes router can hold up to five connections simultaneously enabled by a



	E (0)	W (1)	N (2)	S (3)	L (4)
Free	1	1	0	0	1
In	-	2	3	-	-
Out	-	-	1	2	-

Figure 2.13: Hermes routing table example

switching table inside the router that keeps track of the communication. In Figure 2.13, a sample switching table is provided corresponding to the Hermes switching configuration displayed alongside. Hermes's switching table consists of three rows, one denoting free output port and other two pointing to the input and output connections of the corresponding port. The arbitration block inside the control module operates in round robin fashion and as the Hermes design features configurable port numbers, it can be used for more complex regular topologies like torus [68] or hypercube [69] with appropriate changes in routing algorithm and header format.

Hermes is able to provide BE service to packets however Hermes packets can be blocked and hence get delayed indefinitely. Even though Hermes provides a low overhead NoC architecture, its inability to provide performance guarantees, lack of packet prioritisation and inability to tolerate irregular topologies remain its major handicaps.

2.3.2. QNoC

QNoC (Quality of service NoC) [70] was designed to support diverse QoS (Quality of Service) requirements by providing different service levels for communication. Each QNoC router has five input buffered ports; one connected to the local IP and the rest to the neighbouring IPs on the mesh.

For satisfying different communication requirements, QNoC packets have four service levels and hence four Virtual Channels. The 'Signalling' service level has the highest priority and is used for urgent messages (usually very short in nature)

like interrupts and control signals. The ‘Real-Time’ service level has a lower priority than signalling but it provides guaranteed bandwidth and can be used for applications like streaming audio/video data. The ‘Read/Write’ service level has lesser priority than real-time and can be used for short data transfer like short memory or register access. The ‘Block-Transfer’ service level has the least priority and is used to transfer large blocks of data or for long messages like DMA access.

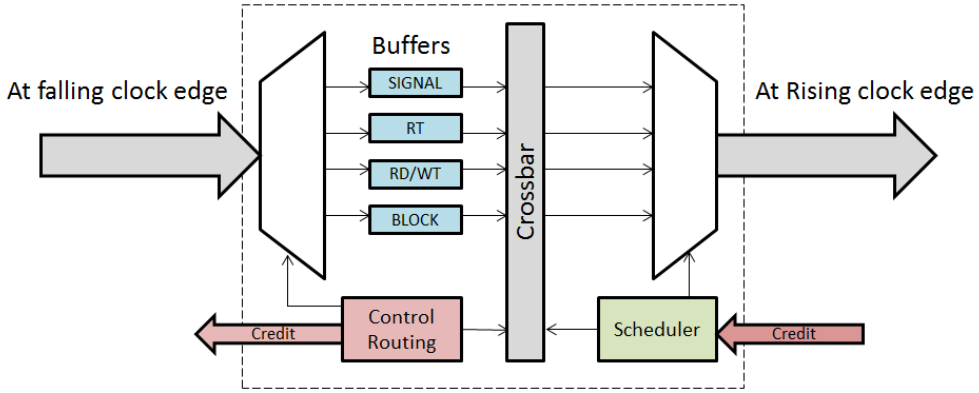


Figure 2.14: QNoC structure ([66])

Nevertheless, the use of VC come with increased hardware requirements and limitation in scalability as seen in section 2.2.3.

2.3.3. AEthereal

AEthereal NoC [71] is a synchronous indirect network which supports both contention free GS and BE traffic. AEthereal’s design philosophy is based on the arguments that the causes of unpredictable behaviour are packet dropping (due to buffer overflows, misrouting, router failure etc.) and contention and congestion. As a **N**oC working reliably aided by a flow control mechanism can resolve packet dropping, the idea was to deal with contention and congestion by using Circuit switching to ensure distinct spatial isolation and TDM to enforce distinct timing isolation.

GS is provided by employing TDM and the BE router utilises the bandwidth unused by the GS router. The BE router utilises input queued wormhole routing

with round robin arbitration and the packets are source routed to the destination thus making it simple.

Configuring the slot tables to enable TDM based communication can either be done in a centralised manner or in a distributed fashion. In the centralised method, programming is done from a centralised module using special packets to setup or remove slot allocation. Due to the use of a centralised programming module, this approach has low hardware requirements but the design will have high reconfiguration time which will increase with the size of the NoC. Thus the centralised methodology has limitation in its dynamic behaviour as well as limitation in scalability.

To enable better dynamic behaviour and scalability, AEthereal design also features a distributed programming model. In this model, IPs sending packets will be able to configure the slot tables along the path of its transmission using packets that use the BE service. This reduces the reconfiguration time compared to the centralised approach however the reconfiguration time depends on the load on the NoC at that time as it is using the BE service. As a result, the reconfiguration time can get increased if the NoC is heavily loaded and due to the distributed nature of the approach, the hardware overhead is higher than the centralised approach. Though scalable, as per their own admission in [34], implementation of a distributed run time slot allocation algorithm is complex.

2.3.4. Other NoCs

There are also architectures that employ hybrid approaches. For example, in MANGO [26][27] NoC, BE services are provided by employing credit based source routing and GS services using VCs. As a result, a MANGO router will internally consist of a BE and GS router. Similar to AEthereal, the GS router has the upper hand in priority and GS router use the VCs that are unused by the GS router. Similarly in Nostrum NoC [74] a hybrid of TDM and VC mechanism is used to provide GS with decreased power consumption.

With Express Virtual Channels (EVC), Kumar et al. in [75] as well as Krishna et al. in [76] aimed at improving latency by providing extra connection links be-

tween routers that are longer thus bypassing intermediate routers. As a result, with EVC, packets that usually will have to do multiple hopes to the destination will be able to bypass intermediate routers to improve its latency performance.

Grot et. al in Kilo-NoC [77] used VCs as a means to improve QoS (predictability). The routers were equipped with logic to pre-empt and discard a low priority packet if in contention with a higher priority with dedicated connections to initiate the packet sender to retransmit the discarded low priority packet. With such a system, Kilo-NoC did not have service levels for packets and hence dedicated buffers; thus enabling it to be scalable.

Other designs like Octagon [78] and SoCBus [50] employ circuit switching as the means to support QoS. However the use of circuit switching can result in severe blocking (when the messages are frequent) and would limit the overall network utilisation of the NoC as the network path is reserved ahead of the transmission of the packet and is held until end of transmission (as seen in section 2.1.1).

The commercial NoC architectures like seen from companies like NetSpeed Systems [79], Arteris [80], Sonics [81] and Aims Technology Inc [82] provide some QoS support in their design. However, the techniques employed in those are proprietary trade secrets and are not available in contemporary literature.

Table 2.1: QoS support on NoCs

NoC	QoS support	Key feature
AEthereal [71]	Circuit-switching/TDM	QoS
aSOC [83]	Circuit-switching	Energy saving
Catnap [84]	-	Energy saving
DSPIN [85]	-	Energy saving
Eclipse [86]	-	Fault tolerance
EVC [75]	Virtual Channels	QoS

Hermes [15]	-	Low hardware overhead
MANGO [72]	Virtual Channels	Energy saving & QoS
Kilo-NoC [77]	Virtual Channels	Scalability & QoS
NetSpeed [79]	Adaptive routing (proprietary)	Customisability & Scalability
Nostrum [37]	Virtual Channels/TDM	Energy saving & QoS
Octagon [78]	Circuit-switching	QoS
Proteo [87]	-	Customisability
QNoC [66]	Virtual Channels	QoS
SoCBus [50]	Circuit-switching	QoS
SoCIN [88]	-	Customisability & Scalability
SoCWire [89]	-	Fault tolerance
Xpipes [90]	-	Customisability

Table 2.1 shows the QoS support available with the prominent NoC architectures. It can be seen from the table that many of the designs are aimed at low energy, fault tolerance and customisability as key goals.

As seen in the table, the designs that aim at predictability (QoS) typically use VC, TDM, Circuit switching, adaptive routing or a combination of those. As seen in section 2.2, the use of TDM results in limitation in dynamic behaviour and scalability while circuit switching limits the overall network utilisation of the NoC [50] [51]. With the use of VCs, the NoC can suffer from limitation in scalability [77] and high hardware requirements. As adaptive routing aims at spreading the load on the NoC rather than resolving the contention issue, it can be ineffective if the NoC is evenly loaded (without hotspots) and under intense load.

Although the combination of the above techniques can resolve some of the limitations, it can be seen that none of the architectures considers simple routers with added logic functionality that would adapt its internal arbitration policies or packets to improve packet predictability. The chief novel advantage of such predictability enhancement measures will be the low hardware overhead, scalability and dynamic behaviour support achieved by abandoning the use of VCs and TDM based functionality.

2.4. NoC Modelling

NoC modelling can be classified into two approaches, direct (where the whole NoC is modelled simultaneously) and virtualised (where a limited number of components are modelled at any time to emulate NoC functionality).

2.4.1. Direct Modelling

In direct modelling, the whole NoC including routers and links are simulated/implemented simultaneously providing a timing accurate model of the whole system. As a result, the resource requirements are higher than virtualised approaches (which will be introduced in the next section).

In [91] and [92], Genko et al presents such a direct modelled NoC emulation framework shown in Figure 2.15.

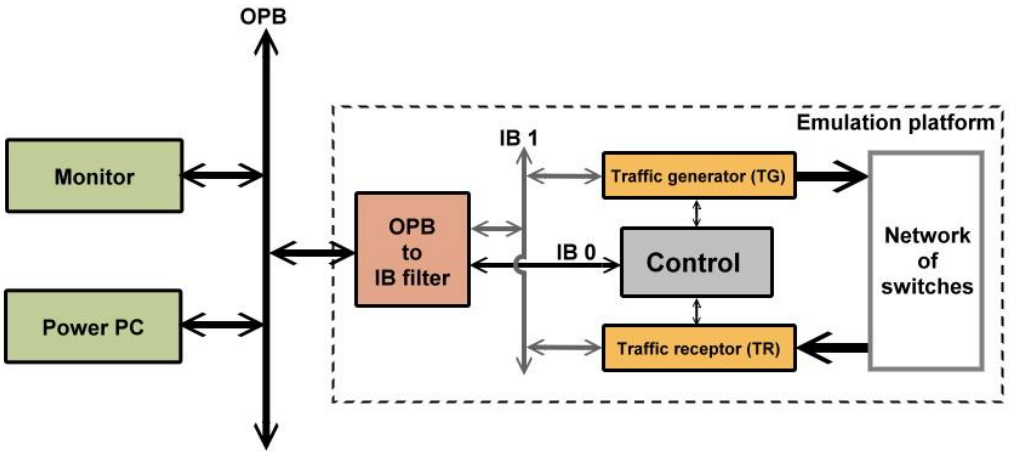


Figure 2.15: Direct modelling example [91].

As seen in the figure, the traffic generator (TG) is managed by a control module and is used to inject traffic into the NoC in stochastic pattern or in a trace driven pattern. For stochastic data, the logic is modelled as C code and for trace driven pattern, C code can be embedded with the data which will be executed by the hard-core processor 'Power PC' included in the system. Just before simulation, the whole traffic data is stored in the RAM ready for injection with each packet containing distinct fields specifying packet length, destination and time of injection along with its payload.

The traffic receptor (TR) receives the packets from the network of switches so that the performance can be interpreted in a basic format and stored into the traffic receptor or stored in memory so that the Power PC can analyse the data in detail later. The traffic generator is capable of performing a basic analysis by counting acknowledgements at a certain interval of time and could interpret it as histograms which could be monitored by the Power PC. Similarly in [93], Papamichael et al presents a direct architecture modelled in HDL. In the design each HDL coded router had a traffic source to inject traffic and a traffic sink to drain the packets once it reach its destination. The routers were interconnected as per the required topology with all the routers functioning in parallel.

As a result, direct modelling based designs will have high resource requirements but will provide a timing accurate performance model of the NoC system.

2.4.2. Virtualised Modelling

In contrast to direct modelling, virtualised approaches have lower hardware requirements, as the whole NoC will not be modelled at any single point of time. Virtualised systems are designed to use a limited set of NoC elements to emulate the functionality of a complete NoC without full modelling.

For example, RAMP gold [94] is a virtualised many-core system simulator by which shared memory many-core system of up to 64 cores can be simulated. The RAMP gold simulator was designed in System Verilog [95] as two separate modules, firstly a 'functional model' to meet the required functional requirements and

secondly a ‘timing model’ to manage timing of the cores in executing their respective instructions.

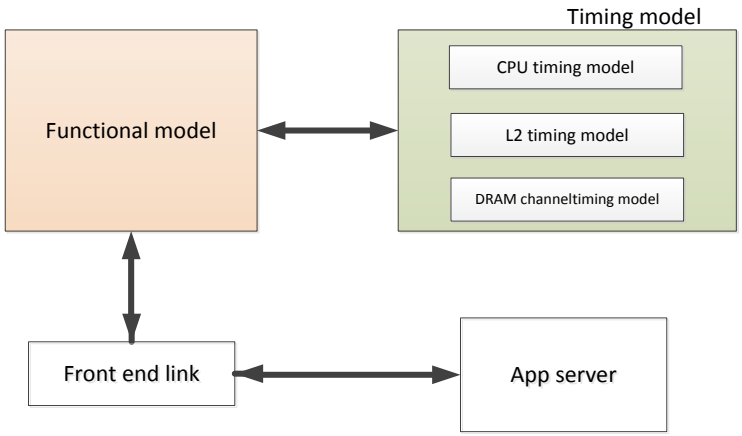


Figure 2.16: RAMP gold virtualised simulator [94]

Due to the lack of flexibility in using soft processors to simulate different cores, the functional model is designed as a 64-thread feed-through pipeline (host-multithreading [96]) with each thread simulating a separate core which is tracked and managed by the ‘timing model’. Each core is provided with a L1 cache (for instruction and data) and a shared L2 cache which is connected to a DRAM via a controller which has the additional functionality of modelling the delays using FIFO queues. For evaluating the performance of the cores, each core is provided with counters along with global counters to monitor cache events like hits, misses and write backs along with target clock cycles. The work reports simulation of a 64-core system with several times speedup compared to their reference software simulator.

A similar approach for simulating medium to large virtualised networks was presented in [93].

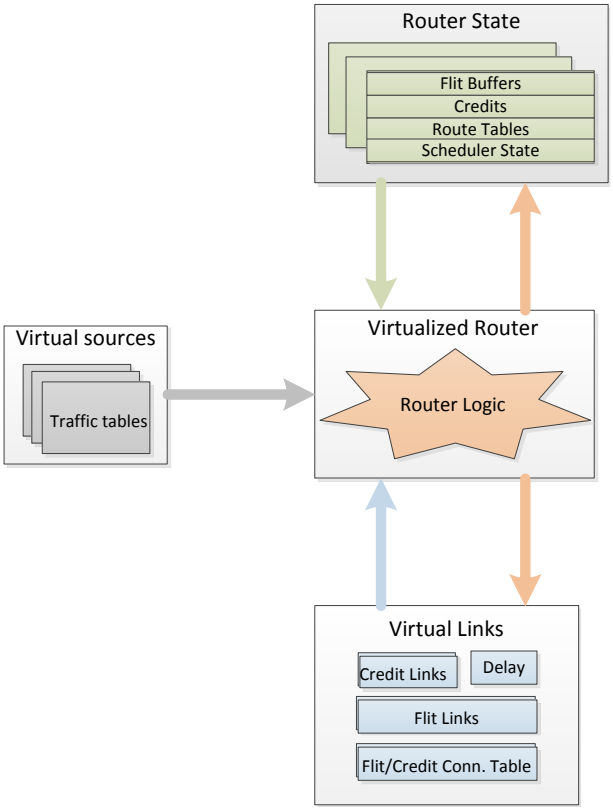


Figure 2.17: Medium large many-core virtualised modelling [93]

As seen in Figure 2.17, the design has a 'router state' module that stores the state of all routers (like flit buffers, routing tables, credits etc.) and there is a 'virtual links' module which is responsible for moving flits and credits between each virtualized router. The router logic consist of FSM which is used to transform the virtual router state from one to the next with respect to the traffic injected by the 'virtual sources' module. As the system simulates a single router at a clock cycle, it can be used to evaluate large networks but the system requires complex FSM logic to manage the whole simulation procedure. However, as the whole NoC is not modelled simultaneously, the speed of simulation will be lower than a comparable direct modelled design.

There are also cycle accurate [97] [98] and Transaction level [99] [100] software simulators that are designed to be faster than HDL based approaches [101]. As this research aimed at HDL based cycle accurate prototypes (for accurate overhead evaluation), such options were not explored in detail.

2.5. Summary

This chapter initially looked into the motivating factors for NoC based research and the basics of NoC designs. The initial part also covered the design choices such as switching techniques, routing algorithm and arbitration techniques.

The predictability enhancement technique TDM that improved predictability by multiplexing the functionality of the NoC in the time domain followed next. It was seen why multiplexing in the time domain can work with static traffic where the traffic pattern is known ahead of time. In such cases, designers would even be able to provide quality of service with hard deadlines. Hoverer, with dynamic traffic, assuring hard deadlines is not possible as the contention pattern in the NoC is not known beforehand. Though centralised TDM slot configuration can allow TDM based routers to function in a dynamic environment (dynamic traffic), the high and uncertain re-configuration time required for routers can result in significant latency variation in packets during the initial setup period. Furthermore, the centralised approach limits the scalability of the NoC. Even though a distributed programming approach (for TDM slot tables) can ensure NoC scalability, such approaches require complex design and implementation considerations [102].

The chapter then looked into LDM and VC based approaches which provided QoS support to dynamic traffic. While VCs rely on separate logical channels, LDM relied on multiplexing the communication link itself.

With LDM, the bandwidth of the link is multiplexed to allow multiple packet flows through the same link simultaneously. This required serialising and de-serialising logic at output and input port resulting in significant hardware overhead. Furthermore, LDM has scalability issues as with the increase in the number of packet priorities in the NoC (on a bigger NoC), the efficiency of the approach can decreases unless the link width is increased proportionately resulting in further increase in hardware requirements.

With VCs however, the functionality of the router is split into separate logical channels with dedicated buffers for each virtual channel. As a result, VC based

designs have high hardware requirements. Furthermore, with the increase in size of the NoC, the advantages brought about by the technique will be reduced. To counter this, the number of VCs would have to be increased which will result in even higher hardware requirements.

The chapter also looked into centralised and distributed adaptive routing approaches that tried to improve predictability by distributing traffic load by monitoring the instantaneous traffic. As the techniques rely on distributing the load rather than resolving the core predictability degrading issues, they can be ineffective in heavily loaded NoCs and where the NoC is evenly loaded. Also, in systems where the change in traffic is considerably faster compared to the adaptation time, the techniques will be ineffective. Section 2.3 looked into some of the common NoC architectures from the simple Hermes to the VC based QNoC and the AEthereal NoC that employ TDM with Circuit switching, along with other commercial NoC architectures and their features.

It can be seen that most NoC architecture that are designed for QoS (predictability) typically use VCs or TDM thus resulting in high hardware requirements or limitation in scalability and dynamic behaviour. As a result, it can be seen that there is a clear gap in the literature as designers rarely consider scalability, QoS and hardware overhead reduction (for dynamic traffic) simultaneously as design goals. As a result, the thesis presents techniques that will enable routers to be scalable and relatively hardware inexpensive, while providing QoS to packets by using techniques that dynamically modify router parameters and packets.

The final section looked into the NoC modelling techniques; Direct and Virtualised modelling. The section also discussed how direct modelling provided a timing accurate model of the whole NoC though requiring more resources than the virtualised approach which does not model the whole NoC simultaneously (hence allows modelling of large NoCs). This supports the use of the direct modelling technique in the prototypes presented in the following chapters.

.

Chapter 3

Metrics and Problem Statement

Although NoCs have been proposed as a promising communication infrastructure for many core systems, contention between non-preemptive packets intensified by the multi-hop nature of communication can result in variation in latency of packet reception.

In an embedded system that uses NoCs as the communication infrastructure, there can be packets that have to be consistently delivered with low latency (and hence without high variability) as denoted by their high application supplied priority values [103]. Considering packet predictability as the reduction in the variability in packet latency [23], predictability enhancement (depending on packet priority) is hence an important consideration while designing an embedded system that deals with dynamically varying traffic. The terms packet predictability and dynamically varying traffic are defined as follows.

> ***Packet predictability:*** *Packet predictability enhancement is defined as the reduction in variation in latency of the packet. So a packet with lower variation in latency is considered more predictable than one with higher variation.*

> ***Dynamically varying traffic:*** *Traffic that has no bounded time interval between successive packets and no upper or lower bounds on packet length.*

If the packets have high variation in latency due to contention between packet flows, the probability of missing their soft deadlines would get higher. To counter this, the mapping of tasks would have to be done conservatively and higher performing IPs (e.g. faster CPUs) would have to be employed thus resulting in lower overall resource utilisation and excessive hardware requirements respectively [46][104][105].

As a real world example, take the case of an Electronic Control Unit (ECU) that manages the engine operation of an automotive system. With such an ECU (does not necessarily use NoCs) there will be data with different QoS requirements. For example, high latency variation in the throttle position sensor data or air temperature sensor would be acceptable. Even though packet latency variation in data from more critical systems like crankshaft position sensor, ignition or fuel injection system can be tolerated occasionally, consistent latency variation in the data can result in unfulfilled emission guarantees [106] which can cause serious legal issues for the manufacturer.

This makes it important to improve packet predictability, and quantifying the magnitude and variation in latency of packets along with the associated overhead allows comparison between predictability enhancement techniques.

As the work in the thesis aims at improving packet predictability in scalable NoC routers, the initial part of this chapter will look into the metrics that will be used to evaluate the techniques presented so that the advantages and disadvantages can be analysed. The chapter will then present the problem statement followed by experimental results that show the significance of the work presented in the thesis. The final section of the chapter details the evaluation model used in the thesis.

3.1. Metrics

3.1.1. Performance

Starvation of packets is a critical issue as far as a NoC is concerned as it can result in failure in packet delivery. The term starvation is defined as follows.

Starvation: *Blocking of packets indefinitely, resulting in packet delivery failure.*

Starvation of packets is quantified in the thesis by using cumulative count of packet reception. The term cumulative count of packet reception is defined as follows.

Cumulative count of packet reception: *(For each packet priority) Number of packets that were received successfully at that priority level or higher.*

So, when packet starvation is evaluated, the performance of NoC designs are presented as line graphs with cumulative count of packet reception in the Y axis and packet priority in the X axis. An example cumulative count plot is shown in Figure 3.1. In this thesis, it is assumed that packet priorities decrease with the increase in the numeric value of priority (i.e. Packet priority 1 > 2 > 3). In the plot, it can be seen that the number of packets received with packet priorities 1, 2 and 3 are higher in the case of NoC B than NoC A. Even though the packet priority values are discrete values, the points on the plots are connected using lines to aid visualisation.

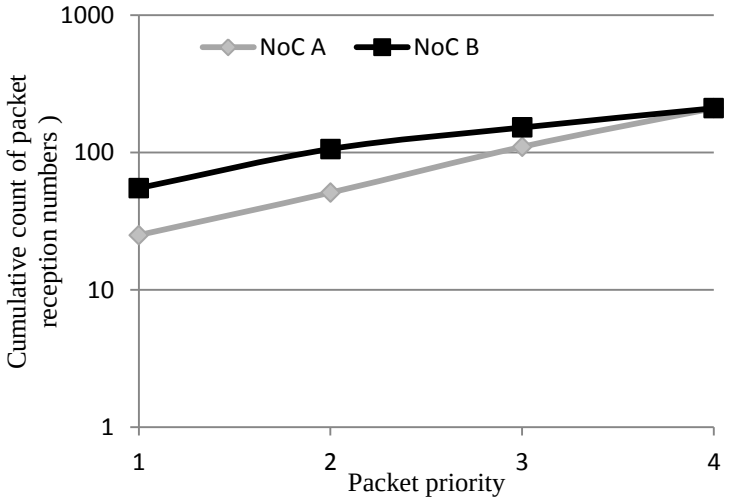


Figure 3.1: Example cumulative count plot

As seen in the initial part of the chapter, latency variation is an important aspect of the NoC design improving which will aid in resource optimisation. Latency variation is quantified using interquartile range (the difference between the 3rd and 1st

quartile of latency) of all packets of that priority level, during the course of a single simulation run. Latency variation is compared between NoC designs primarily using box plots that present an evaluation of both the magnitude and variation in latency between NoCs for the same traffic pattern.

An example box plot is shown in Figure 3.2 and in box plots, packet priority is presented in the X-axis and latency in shown in the Y-axis. The box plot whiskers show the extreme cases of latency while the boxes show the first and third quartile of latency. So, the shorter the box and whiskers are the lower the variation in latency and the lower the box and whiskers are the lower the magnitude of latency.

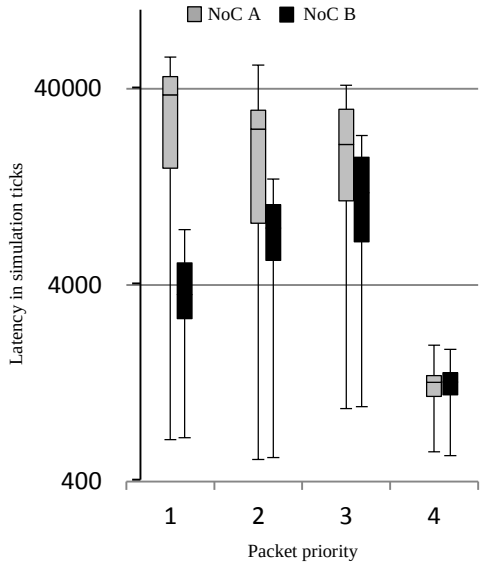


Figure 3.2: Box plot example

In the example, it can be seen that for packets 1 and 2, NoC B has lower magnitude and variation in latency compared with NoC A depicted by lower and shorter box and whiskers.

As the techniques also result in reduction of the magnitude of latency with respect to the packet priority, the thesis also present average latency plots depicting average latency in the Y axis and packet priority in the X axis. In places where multiple techniques are evaluated simultaneously, variation in latency is evaluated additionally using interquartile range plots and using the variation metric we call S-index.

As the thesis do not deal with packets with hard-deadlines, there can be a few extreme cases of latency per packet priority which has to be ignored. As a result, Interquartile range is used in the thesis to quantify latency variability as it is not affected by extreme cases. For the same reason, interquartile range has been used in works like [107] and [108] and as per Buch in [109], it is one of the robust methods of estimating the trend of a distribution which has non-deterministic events.

Even though the packet priority values are discrete values, the points on the plots are connected using lines to aid visualisation as with the cumulative count plots. Similarly, the average latency plots (which depict the magnitude of latency for each packet) will look similar to interquartile latency plots but converse to interquartile latency plots, those will have average latency in the Y axis.

As interquartile plots are lines that show the latency variability over the whole priority range, each of those lines are concatenated into a single metric called the S-index for the ease of comparison between NoCs.

S-index (estimated using equation (3.1)) is used to quantify the latency variation of all packets priorities of a NoC into a single metric within a specific traffic pattern.

$$S - \text{index} = \sum_{P = P_{range}} \frac{Q3_P - Q1_P}{P \times W} \qquad \textbf{(3.1)}$$

(P_{range}- Range of packet priorities, $Q3_P$ -3rd Quartile of latency, $Q1_P$ -1st Quartile of latency, P- Numeric value of packet priority, W- Weightage relation)

In the equation, the term W is used to specify the weightage between packet priorities when computing the S-index. In this thesis, as it is assumed that the weightage of packet priorities decrease linearly with the increase in the numeric value of priority and hence W is set at one.

S-index allows predictability (latency variation) comparison between predictability enhancement techniques within a specific traffic pattern. Thus, a lower S-index value for a NoC shows lower variation in latency compared to another for that specific traffic pattern. More details on the effect of latency variation of packets (depending of priority) on S-index is added in Appendix 4.

Additionally, to show the advantages brought about by the techniques presented, plots depicting maximum latency of packets are used thus showing extreme cases of variability as well as plots showing the cumulative count of late packets (compared to a soft deadline) thus showing magnitude of latency.

3.1.2. Load

With NoCs, packet latency variation increases with the increase in load (due to the increase in contention between packets) on the NoC. As a result, to monitor the latency variation of the NoC with the increase in contention, the load on the NoC has to be quantified.

At any instant, the network load can be quantified as link utilisation (as in [110]). This thesis however quantifies it as average link utilisation (as in [111]) as an estimate of the non-deterministic load over the whole simulation run into a single metric. The average utilisation per link V is used as the measure of load in the NoC and it is estimated using equation (3.2).

$$V = \left\{ \sum_{x=0\ to\ W-1} \sum^{y=0\ to\ H-1} \left(\frac{D_{x,y}}{P_{x,y}} \right) \right\} / L_{(W \times H)} \qquad \textbf{(3.2)}$$

(W- NoC Width, H- NoC Height, D- Total transmission time of that packet, No load latency, P- Period, L- Number of links)

To estimate V, the utilisation of the NoC by packets of each priority is estimated by taking the ratio of the total transmission time ($D_{x,y}$) and period ($P_{x,y}$) for that packet. The sum of the utilisation of all packet priorities provide the total utilisation and the value divided by the number of links ($L_{(WxH)}$) gives the average utilisation per link V as seen in equation (3.2).

As V is the average utilisation per link, it does not provide information on the load on the NoC at any instant of time or the load at any specific link in the NoC. However, V can be used to quantitatively compare loading on the NoC between traffic scenarios.

3.1.3. Hardware Resources

The hardware overhead for the techniques presented is evaluated by the complexity of the logic required for each design. In this thesis, hardware overhead is quantified in terms of Lookup table (LUT) and register utilisation figures for the design to be implemented on an FPGA.

However, depending upon the architecture of the FPGA, the LUT and register utilisation can vary for the same design. As a result, to enable accurate comparison between the techniques presented, xc7a350t Artix-7 [112] FPGA (having $2.25x10^5$ LUTs and $4.5x10^5$ slice registers) was chosen as the standard and all the designs were evaluated targeted at that specific FPGA. To enable this, the NoC models were designed in synthesisable HDL and was evaluated using Xilinx Vivado [113] tool.

Even though the hardware overhead for a design on an FPGA and Application Specific Integrated Circuit (ASIC) would not be numerically equal, an FPGA based implementation can give an estimation of the complexity of the logic to be implemented on ASIC for the same design. For example, Gaj et.al. in [114] compared overhead of several algorithms on FPGA and ASIC platforms and reported that there is a strong correlation between the hardware overheads associated in both cases with each algorithm.

ASICs have a much more complex implementation cycle and hence FPGA implementation is used in this thesis.

3.2. Problem Statement

As seen in section 2.2.1, the classical predictability enhancement technique TDM has limitations in scalability and dynamic behaviour.

Although LDM does not have limitation dealing with dynamic traffic, it is not scalable and it results in high hardware overhead as seen in section 2.2.2. Similarly, with VCs, preemptive arbitration brings about high hardware requirements and limitation in scalability [77] as seen in section 2.2.3.

Simple non-preemptive NoCs however are scalable and are dynamic with low hardware requirements. However, they suffer from packet latency variation regardless of packet priority. As non-preemptive NoCs are scalable and have limited hardware requirements, this thesis aims on resolving predictability degrading issues is such NoCs.

As an example, the cumulative count of packet reception numbers of a Hermes based NoC with priority based arbitration (under a HOL blocking scenario added in Appendix 1a) with load V= 0.3, 0.5 and 0.7 is shown in Figure 3.4. The definition of a Hermes based NoC is as follows.

> ***Hermes based NoC:*** *In the thesis, the non-preemptive NoC model (with XY-routing and wormhole switching) based on Hermes (explained in section 2.3.1) is referred to as the Hermes based NoC.*

It can be seen that with the increase in load on the NoC, HOL blocking resulted in the decrease of packet reception numbers of packet 1 and 2. At V= 0.7, they were completely starved.

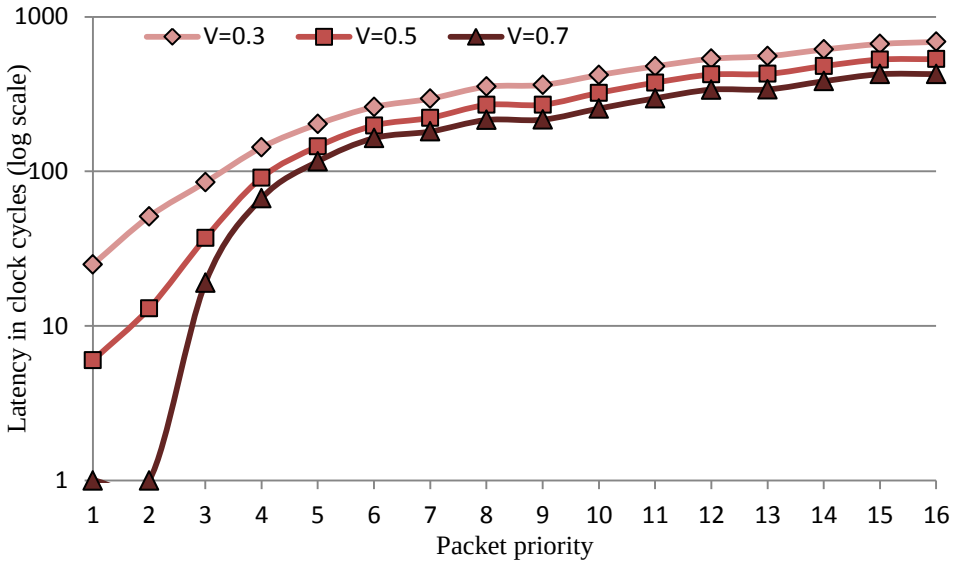


Figure 3.3: Packet starvation with non-preemptive NoCs

Furthermore, due to tailbacking and HOL blocking, non-preemptive NoC packets can have high magnitudes of latency regardless of the priority value as seen in Figure 3.4.

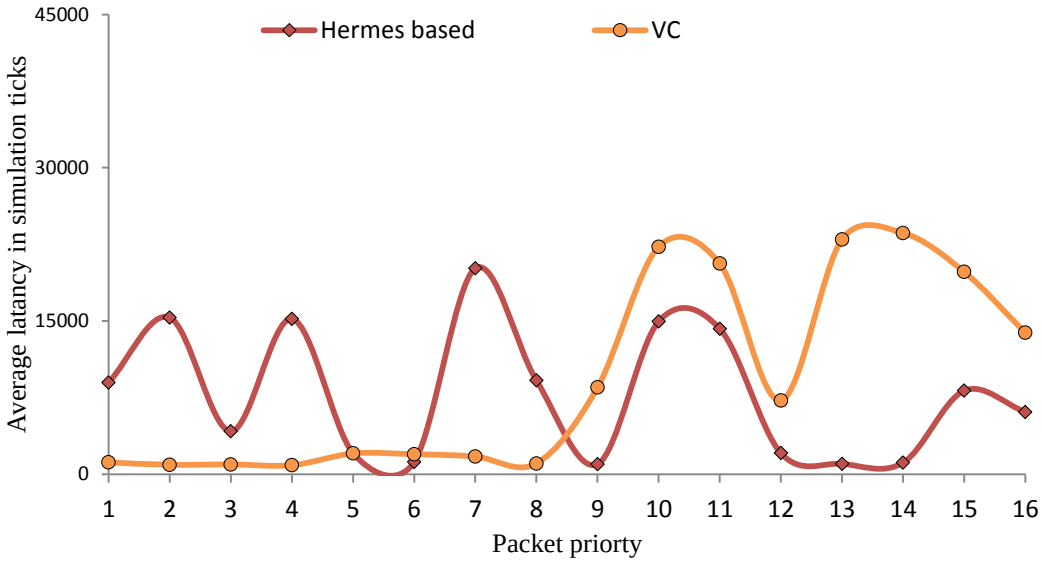


Figure 3.4: Average latency plot comparing Hermes based and VC based NoCs

Figure 3.4 shows the average latency plot of the performance of a Hermes based NoC and a VC based NoC (with 4 service levels under a random traffic scenario added as Appendix 1f) with average latency in the Y-axis and packet priority in the X-axis. The definition of a VC based NoC is as follows.

VC based NoC: *In the thesis, the NoC with preemptive arbitration enabled by Virtual Channels (explained in section 2.2.3) is referred to as the VC based NoC.*

It can be seen that with the Hermes based NoC, the high priority packets (1 to 8) does not achieve any latency advantage in magnitude (depicted by high average latency) compared to the lower priority packets. With the VC based NoC this issue is resolved as the high priority packets (1 to 8) are seen to have low average latency at the cost of the low priority packet's average latency.

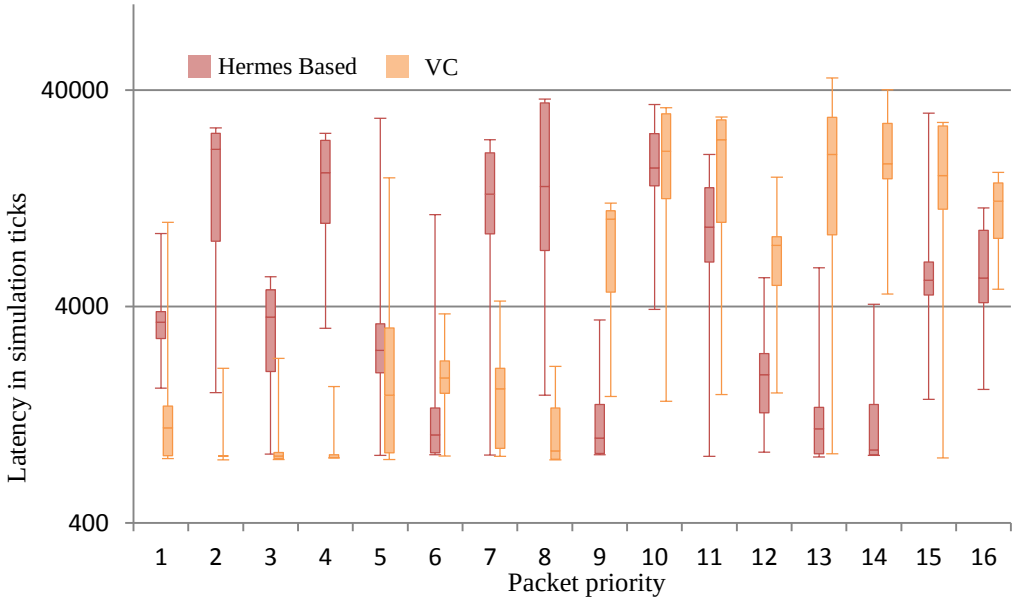


Figure 3.5: Latency box plot comparing Hermes based and VC based NoCs

The latency performance of both the NoCs are presented in Figure 3.5 and it can be seen that the high priority packets of the VC based NoC suffer lower variation in latency compared to the Hermes based NoC depicted by the shorter box and whiskers.

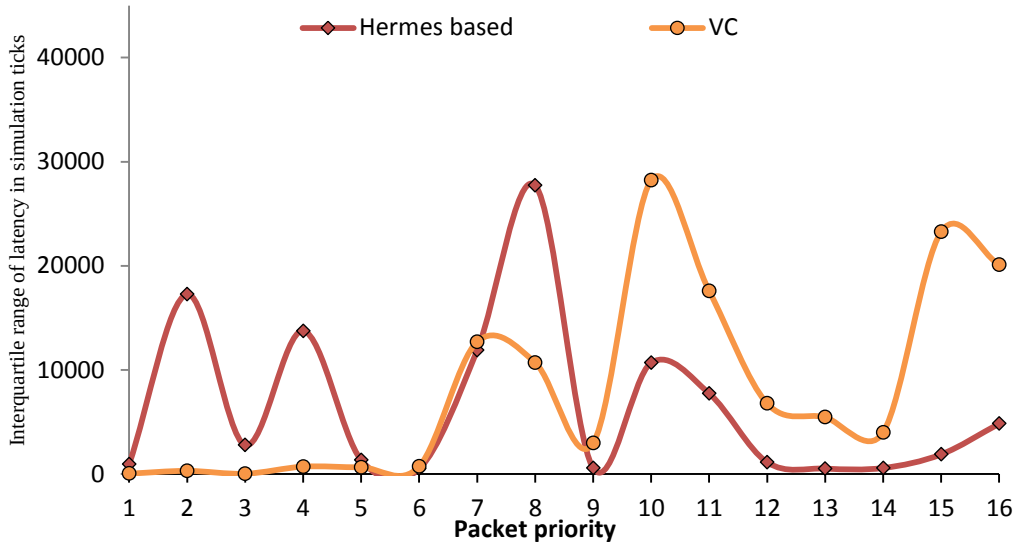


Figure 3.6: Interquartile range of latency comparing Hermes based and VC based NoCs

The latency variation in evident in Figure 3.6 where the interquartile range of latency of both the NoCs are plotted. It can be seen that with the Hermes based NoC, high priority packets 2 and 4 suffer high variation in latency. With the VC based NoC the high priority packets (1 to 9) are seen to have low variation in latency and hence have better predictability than the Hermes based NoC.

However, VC implementation results in high hardware overhead both in terms of LUTs and registers as shown in Figure 3.7 (4x4 NoC).

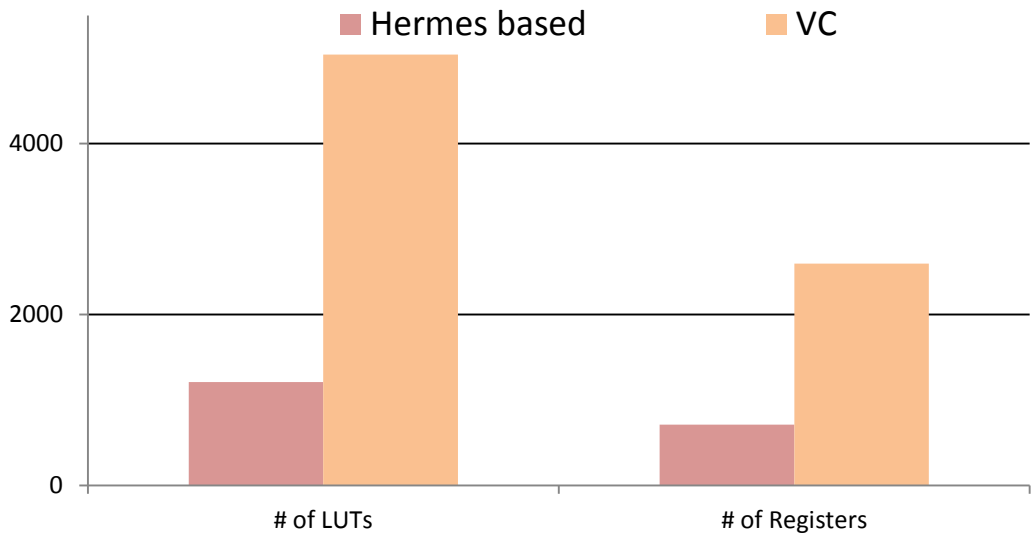


Figure 3.7: Hardware overhead comparison of the Hermes based NoC with the VC based NoC.

As VCs rely on separate buffers for each service level, an increase in size in the NoC would result in decrease in latency performance in terms of both magnitude and variability. As a result, for ensuring predictability (with increased NoC size), the number of service levels would have to be increased hence resulting in a linear increase in hardware overhead.

The research presented in the thesis aims at improving the predictability of non-preemptive NoC packets with low overhead dynamic methods that are completely scalable.

3.3. Evaluation Infrastructure

To evaluate the performance of the developed NoC models, either a hardware platform like a Field Programmable Gate Array (FPGA) can be used or the evaluation can be done in simulation. Though hardware base test infrastructure would be several times faster than simulation [115] [116], such systems provide limited support for monitoring the internals as well as functionality of the design. As a result, even though the models are done with synthesisable HDL, the performance evaluation was conducted with Bluesim [117] simulation environment.

The performance metrics defined in section 3.1.1 are obtained in the thesis in simulation and since the NoCs follow the direct modelling approach, the simula-

tion results are cycle accurate and are in terms of clock cycles. The same models were then synthesised to the chosen FPGA to get the hardware overhead figures as per the hardware overhead metrics described in section 3.1.3.

3.3.1. NoC Framework

The basic evaluation model was designed in Bluespec System Verilog [118] [119] as a configurable direct implemented router design called the R2.

Bulespec System Verilog was chosen as the implementation medium for a variety of reasons. Based around System Verilog, Bulespec System Verilog treats both architectural exploration and verification as part of HDL hence considers verification as a design problem. As a result, the verification time can be reduced significantly thus reducing overall implementation time compared to Verilog, VHDL or System Verilog [120]. Furthermore, it provides several abstraction mechanisms for simplifying the design over System Verilog thus reducing design time further. These features simplify the design process profoundly resulting in reduced number of bugs along with reduced design, verification and debugging time [120].

R2 routers were enveloped in a generic test bench that replicated and interconnected routers and data generator/receptors. The local port of each router was connected to packet generators/receptors so that packet generation and reception can be carried out and documented for analysis.

Hermes is a widely used scalable non-preemptive NoC design with minimalistic hardware overhead [15]. Furthermore, Hermes is widely used in contemporary literature on predictability [48] [52] and hence in this thesis, Hermes is treated as the baseline architecture to which the techniques presented in the thesis are compared.

The basic router follows a five port architecture based around Hermes hence employing XY-routing [57] and wormhole switching [121] for low hardware requirements. To enable scalability and for ensuring low overhead, mesh type topology is used with priority based arbitration.

The R2 design follows a uniform mesh topology and unlike Hermes, each packet header includes a priority value (application-specified priority) which is used by the arbitration unit inside the routers to resolve contention between packets over output ports.

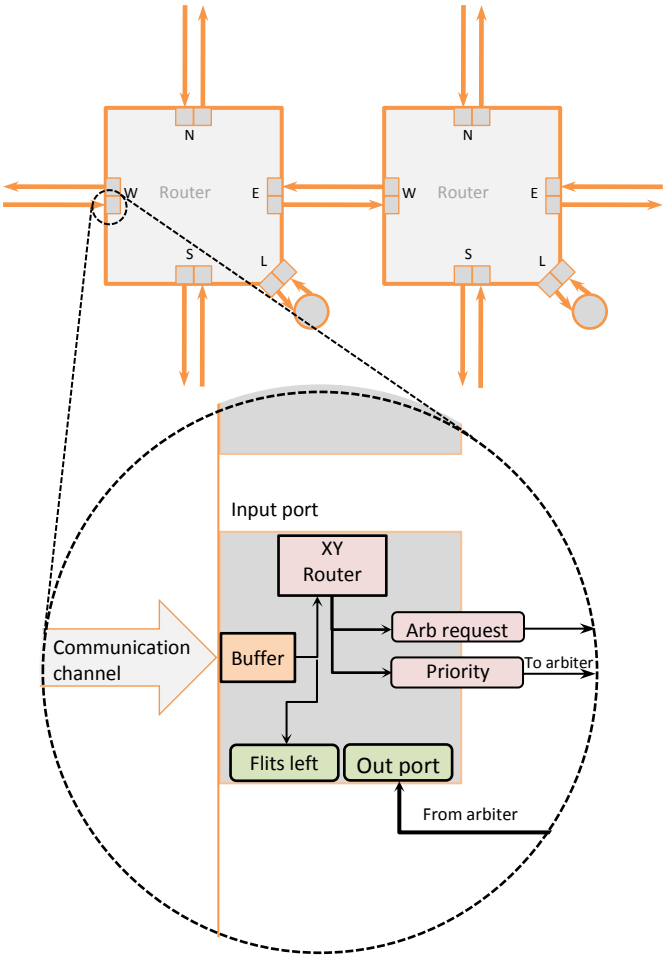


Figure 3.8: Router input port

As shown in Figure 3.8, R2 routers have buffered input ports, which on reception of a packet header employ XY-routing to set the ‘Arb_request’ register and the ‘priority’ register in accordance with the destination and priority information carried.

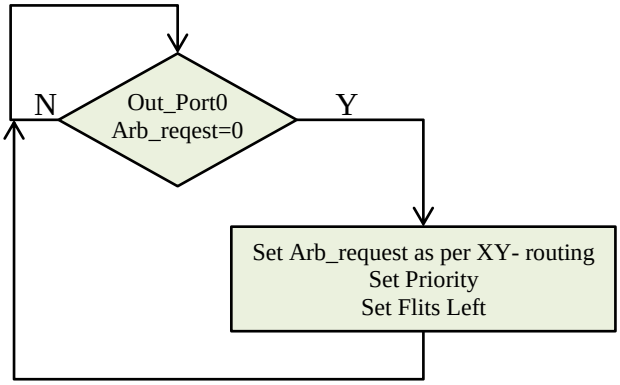


Figure 3.9: XY-routing logic operation

As shown in Figure 3.9, the arbitration unit in the router then checks ‘Arb request’ and ‘priority’ registers of all input ports to provide arbitration to the qualified ports. The arbitration logic simultaneously checks for ports requesting arbitration to each of the output ports. If the associated output port is unused, the router will grant arbitration to highest priority request by setting the ‘out port’ register on the qualifying input port. This will permit the input port to send flits to the allocated output port so that flits could be transferred away through the communication links.

As an example, the logical operation of the arbitration unit for the local port is shown in Figure 3.10.

As shown in the figure, if the output port referred in the arbitration request of the local port is unused and the arbitration request for the local port is active, the arbitration logic for the local port gets triggered.

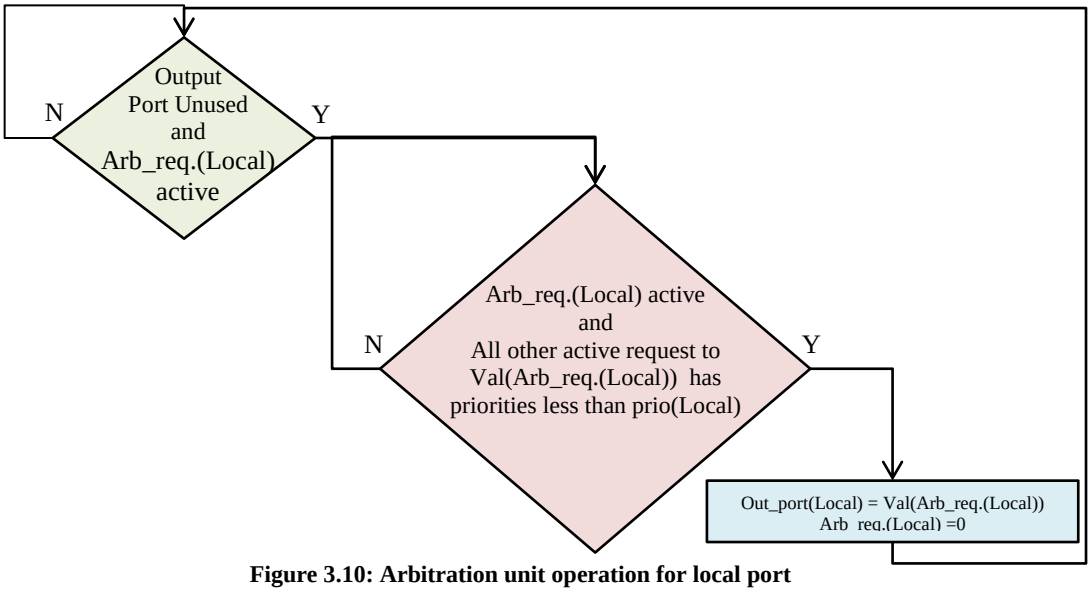


Figure 3.10: Arbitration unit operation for local port

As shown in Table 3.1, each port has an identifier and a value greater than zero inside an arbitration request register represents a valid arbitration request.

Connection	Identifier
Null	0
Local	1
North	2
East	3
South	4
West	5

Table 3.1: Port connection identifiers

The arbitration logic then checks whether all the other arbitration requests (to the same output port) are of lower priority than the current arbitration request and if they are, it provides arbitration by setting the 'out_port' register inside the native input port to the identifier value of the output port. In case there are any other requests with higher priorities than the current arbitration request, the logic waits while the logic inside the arbiter associated with that input port (hosting the highest priority request) executes.

As the flits are being transferred, the input port also decrements the value in the 'flits left' register so that when the value reaches zero, the connection can be closed by re-setting the 'out_port' register value to zero.

Details on the NoC prototypes used in the thesis and the Universal Resource Locator (URL) to the open source code is added in Appendix 2.

3.3.2. Performance Evaluation Framework

The performance evaluation framework consists of packet generator modules that inject packets into the NoC as per a pre-set parameter list. The packet generator configuration is auto generated as Bluespec source code using a custom built code generator (shown in Figure 3.11) which can either configure the generators randomly or in accordance with a series of algorithms to generate specific configuration patterns.

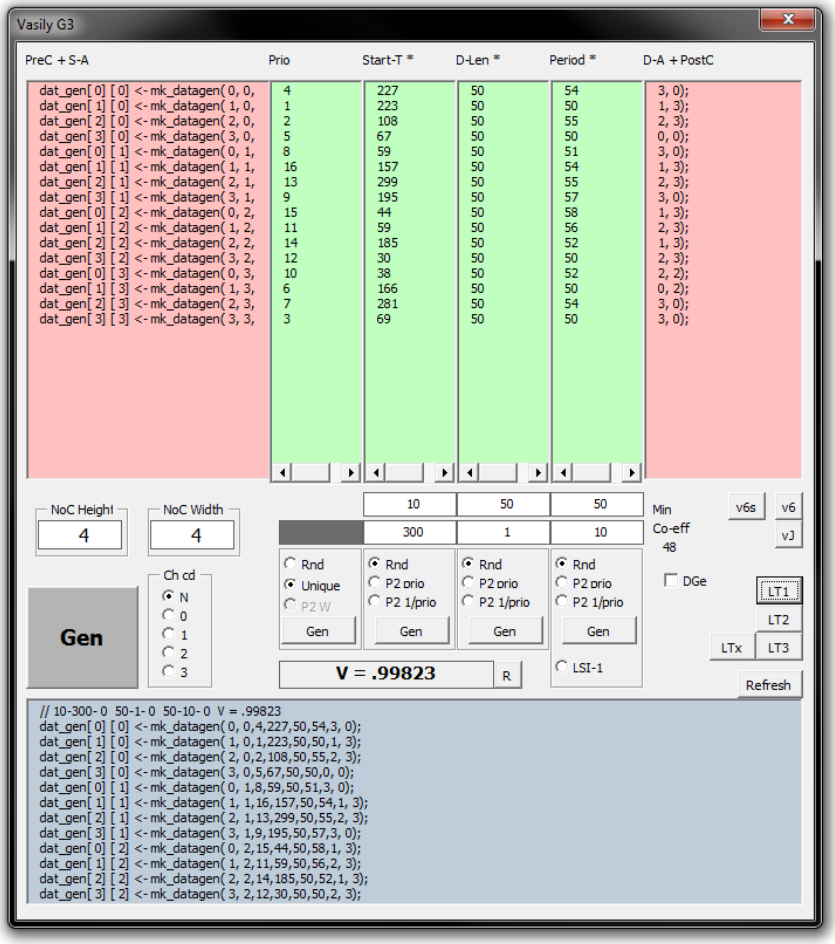


Figure 3.11: Data generator configuration generator

Depending on the parameters set, the code generator generates the source code for configuring the packet generators and this code is added into the HDL code before compilation.

A sample output of the code generator is shown in Figure 3.12. The numbers seen inside the parenthesis are the configuration parameters of each data generator and they represent X-address, Y-address, Packet Priority, Start Time, Packet Size, Period, Destination X-Address and Destination Y-Address respectively.

```
dat_gen[ 0] [ 0] <- mk_datagen( 0, 0,2,139,50,56, 0, 1);

dat_gen[ 1] [ 0] <- mk_datagen( 1, 0,3,147,50,58, 1, 0);

dat_gen[ 0] [ 1] <- mk_datagen( 0, 1,1,083,50,51, 1, 1);

dat_gen[ 1] [ 1] <- mk_datagen( 1, 1,4,220,50,53, 1, 0);
```

Figure 3.12: Snippet from 'data generator configuration' generator output

The generators are designed to send packets of a fixed size and then sleep for a time period after which the same cycle will be repeated.

To enable close simulation of a realistic system, the generators are also provided with logic to assess unforeseen waits imposed on it by the communication network so that the associated time lag can be compensated by decreasing waiting periods. To enable this, the generators have two internal counters, Counter_A and Counter_B. Counter_A is a free running counter which will increment itself irrespective of the state of packet generation logic. Counter_B is the conditional counter that is designed to increment under both of the stages of the packet generation logic; packet injection state and sleep state. So, under ideal conditions, both counters will have the same value in them thereby denoting zero time lag and under this situation, the packet injection and sleep operations will be repeated as per the pre-set parameters.

If the generator becomes unable to inject packets into the NoC due to congestion inside the NoC, the generators will not be able to increment Counter_B while Counter_A gets incremented. This value difference between the counters is used to determine the wait time the generator will have to succumb to the next time.

The generator will then try to equalise the two counters by compensating for the difference by trading the sleep time as shown in Figure 3.13.

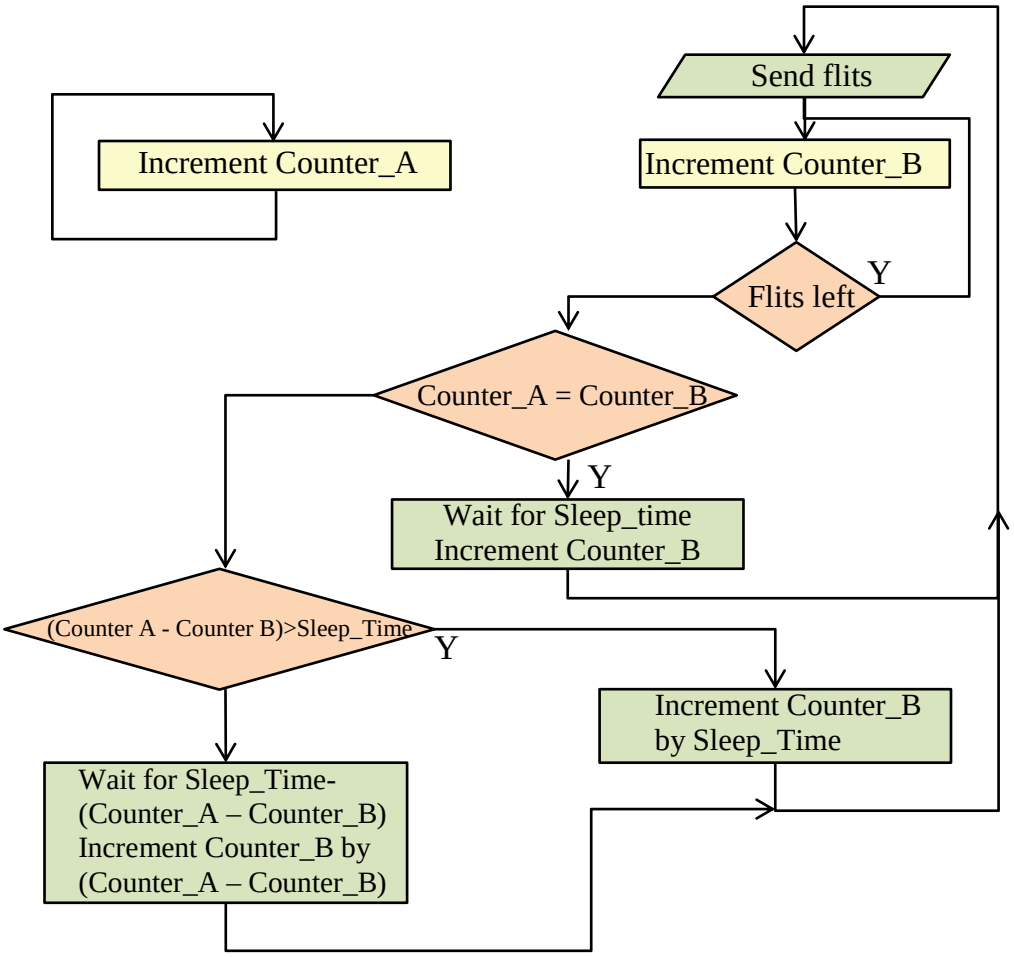


Figure 3.13: Packet generator logic

In case the lag is greater than the sleep time, the generator will refrain from going into the sleep state and will start injection of the next packet immediately after the current packet. The logic also increments Counter_B with the time saved by skipping the sleep stage to document the resultant gain in time. In case the time lag is less than the sleep time, the generator sleeps for a period equal to the difference between sleep time and time lag so that once this is performed, both counters will be running at the same values denoting zero time lag. This enables the generator to mimic the performance of a system with large buffers with minimal overhead.

Apart from conditioning the data to the required flit format and injecting into the NoC, the packet-generators also receive packets from the NoC and export evaluation figures to an external file. A snippet from one of the tests is added as Figure 3.14.

I	11	1	0	3	1	4	179	100	419	700
R	8	2	0	701	1					
I	9	1	3	4	3	0	164	100	438	704
R	25	0	4	706	1					
I	18	1	2	2	2	4	171	100	435	708
R	12	1	4	742	1					

Figure 3.14: Snippet from an exported text file detaining simulation milestones

The lines starting with 'I' depict the timestamp with characteristics of a packet injection while lines starting with 'R' depict the particulars of a packet reception along with its timestamp. These details are then analysed by a custom built macro code running inside the spread sheet software to generate performance statistics and graphs. The detailed information on the injection timestamp and reception timestamp is shown in Figure 3.15 and Figure 3.16 respectively.

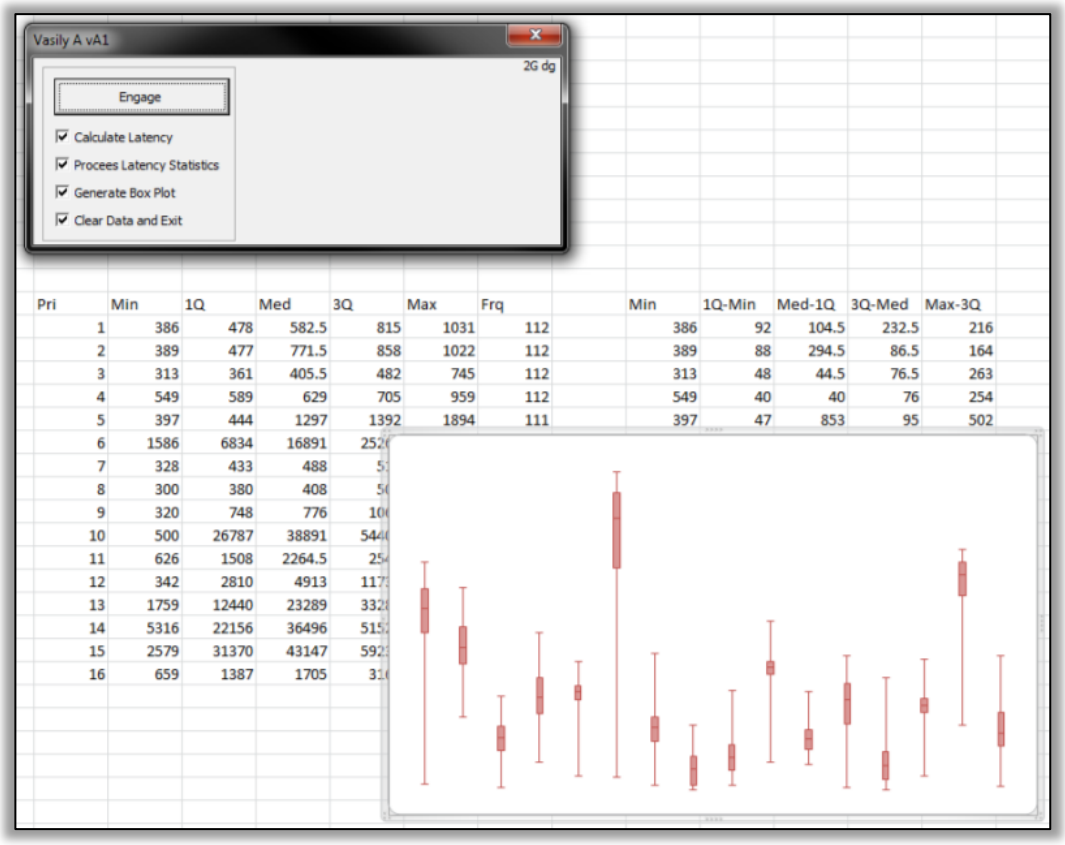
Type	Packet Priority	Source X Address	Source Y Address	Destination X Address	Destination Y Address	Packet ID	Start Time	Packet Size	Sleep Time	Time stamp
I	11	1	0	3	1	4	179	100	419	700

Figure 3.15: Packet injection timestamp details

Type	Packet Priority	Receptor X Address	Receptor Y Address	Reception Timestamp	Packet ID
R	8	2	0	701	1

Figure 3.16: Packet reception timestamp details

The packet latency performance is then estimated by analysing the data file using a VB Script coded analysis macro (shown in Figure 3.17) developed to run inside the spreadsheet software used.



Pri	Min	1Q	Med	3Q	Max	Frq		Min	1Q-Min	Med-1Q	3Q-Med	Max-3Q
1	386	478	582.5	815	1031	112		386	92	104.5	232.5	216
2	389	477	771.5	858	1022	112		389	88	294.5	86.5	164
3	313	361	405.5	482	745	112		313	48	44.5	76.5	263
4	549	589	629	705	959	112		549	40	40	76	254
5	397	444	1297	1392	1894	111		397	47	853	95	502
6	1586	6834	16891	252								
7	328	433	488	5								
8	300	380	408	5								
9	320	748	776	10								
10	500	26787	38891	544								
11	626	1508	2264.5	25								
12	342	2810	4913	117								
13	1759	12440	23289	332								
14	5316	22156	36496	515								
15	2579	31370	43147	592								
16	659	1387	1705	31								

Figure 3.17: Performance evaluation macro

3.4. Summary

This chapter initially covered the metrics the thesis is looking into, latency for performance and LUT and register utilisation for hardware overhead. The chapter continued with the problem statement followed by details on the evaluation infrastructure.

In the problem statement section, the predictability issues in latency suffered by non-preemptive NoC designs were discussed. The literature shows that non-preemptive NoCs can have starvation of packets regardless of the priority value and that the magnitude of latency can vary widely thus rendering application-supplied priory pointless.

The section also briefly looked into how VCs can resolve such issues at the cost of scalability limitation and hardware overhead. This motivates the research pre-

sented in the thesis as the techniques presented aim at improving packet predictability with simple and scalable techniques than the VC approach.

The last section of the chapter detailed the basic evaluation infrastructure used and it acts as a prologue to the specifics of implementation details of the techniques presented in the subsequent three chapters.

Chapter 4

Starvation Resolution by Priority Manipulation

In order to resolve HOL blocking situations, this chapter presents the Priority Forwarding and Tunnelling (PFT) [25] technique that is designed to neutralise HOL blocking scenarios by enabling the blocked packet up the line to remotely manipulate the routers and priority of the packets blocked down the line. Typically, HOL blocking is resolved with preemptive arbitration. As seen in the section 2.2.3, preemption implementation is hardware intensive whereas PFT aims to resolve HOL blocking without the use of Virtual Channels for scalability and reduced hardware overhead.

4.1. Priority Forwarding and Tunnelling

In the PFT technique, when a packet is blocked by another blocked packet of lower priority, the priority and destination information of the packet blocked up the line will be extracted by the respective router and forwarded through the blocked path to the header of the blocked packet down the line as a PFT-flit. Once the PFT-flit reaches the router with the blocked header, the priority value contained in the PFT-flit is compared with the blocked header's priority. If the priority specified in the PFT-flit is higher than the priority of the packet header blocked down the line, the header's priority is boosted to that of the information in the PFT-flit to resolve the block. This part of the technique is called Priority Forwarding and it

permits HOL blocking to be neutralised, hence starvation of packets (due to lower priority packets) can be avoided.

To prevent further HOL blocking on the routers through which the PFT-flits flow, the routers also perform a process called Priority Tunnelling by which the output ports that will be used by the packet blocked up the line in the near future will be locked with its priority value. This prevents other packets with lower priority values from getting arbitration to those output ports temporarily until the blocked packet is transmitted through them.

Since the PFT-flit flows occur only on paths that are completely blocked, the same data lines can theoretically be used by PFT-flits by using a multiplexing logic hence reducing additional overhead of extra connection lines as shown in Figure 4.1.

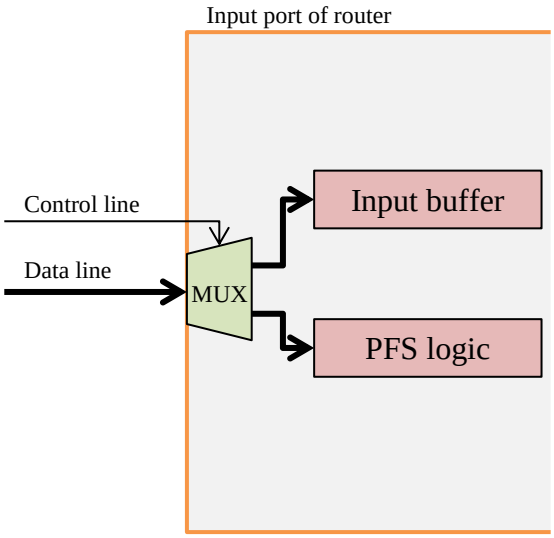


Figure 4.1: PFT-flit transmission

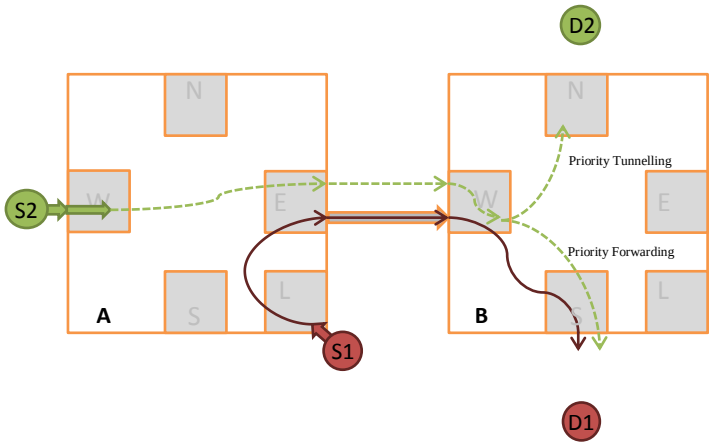


Figure 4.2: Detailed PFT functionality

To understand the technique further, consider the scenario depicted in Figure 4.2 where we can see routers A and B with two packet flows.

Packet flow 1 originates from the local port of router A (shown as S1) and is blocked somewhere south of router B. Packet flow 2 originates from the west port of router A and has destination north of router B.

As packet 2 is blocked by the blocked packet 1, PFT logic will be initiated and the blocking information will be forwarded to the west input port of router B. Now, it can be seen that there is a conflict of interest. As the destination of packet 1 is somewhere south of router B, the PFT-flit has to be forwarded towards the south to do Priority Forwarding. But the port on router B that will be used by packet 2 in future is the north port and hence that is the port that has to be tunnelled.

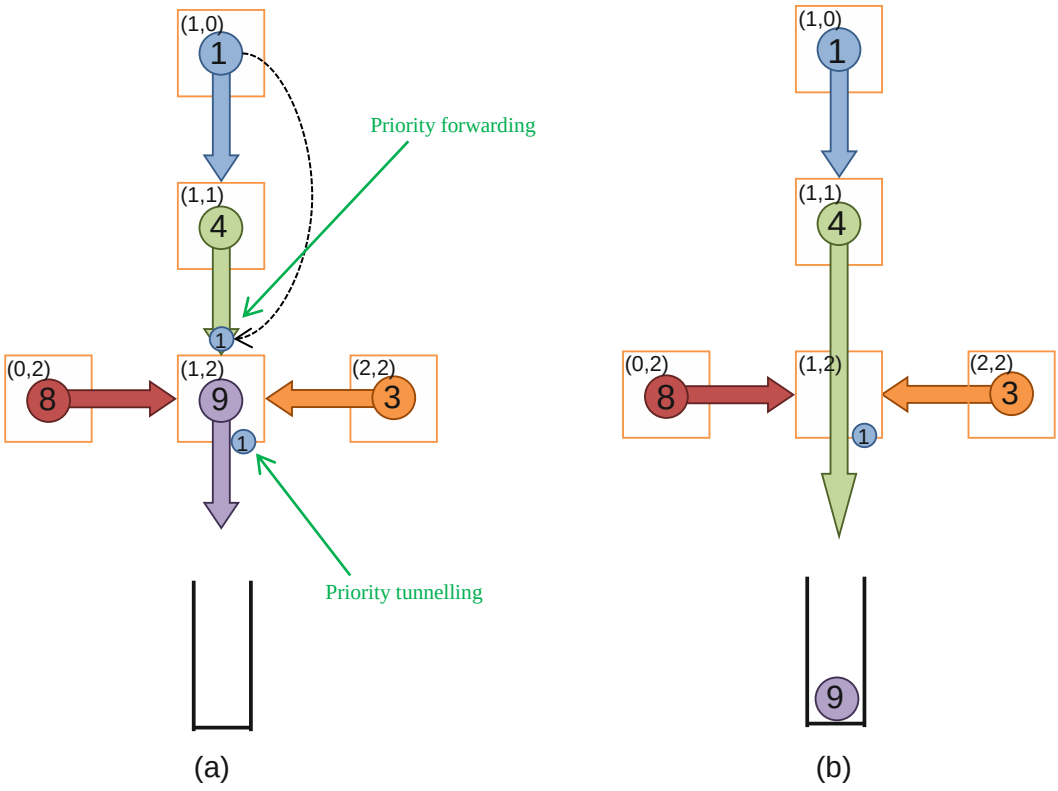
To enable the routers to calculate the correct ports to tunnel, the PFT-flit consists of three components; destination information, packet priority and tunnelling flag. Every time Priority Forwarding is done, the future output port (of the packet behind the line) is calculated by using the destination information contained in the PFT-flit and the appropriate port is tunnelled. If the future port is the same as the one towards which the PFT-flit has to be forwarded, the PFT-flit is send to the next router as such. On the other hand if the future port is different to that towards which the PFT-flit is to be forwarded as seen in the example, the tunnelling flag in

the PFT-flit is disabled and is send to the next router so that the routers down the line do not preform unnecessary tunnelling.

In the example, the router tunnels the north port of router B and then forwards the PFT-flit through the south port with the Tunnelling flag disabled so that the routers down the line does not do further Tunnelling.

The effect of PFT on the HOL blocking scenario explained in Section 2.2 is shown in Figure 4.3. As stated before, it is assumed that packet priorities decrease with the increase in the numeric value of priority (i.e. Packet priority 1 > 2 > 3).

So, ordinarily under the situation in Figure 4.3a, packet 3 will get arbitration to the south port of router (1,2) (after packet 9 is transmitted) ahead of packet 4, hence forcing packet 1 to wait behind the line further. However, with the application of Priority Forwarding, packet 4 is forwarded with the priority value 1 hence allowing packet 4 to secure arbitration ahead of packet 3 (as depicted in Figure 4.3a and Figure 4.3b).



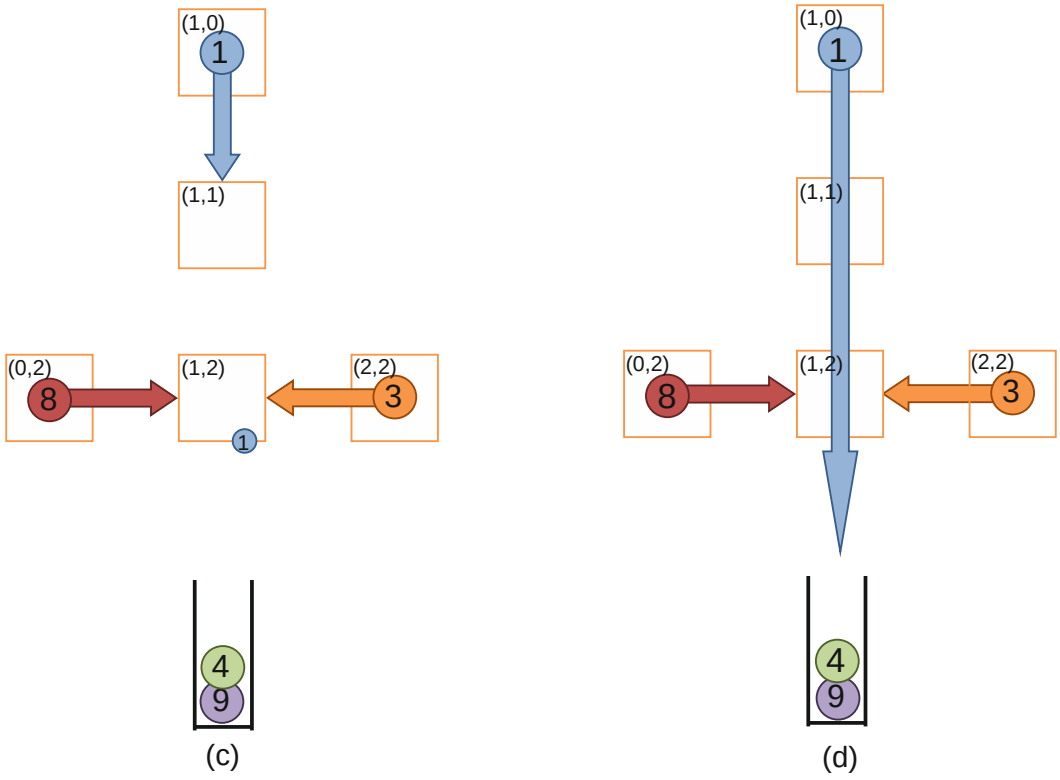


Figure 4.3: Priority Forwarding and Tunnelling operation

Since Priority Tunnelling was performed to the south port of router (1,2), none of the lower priority packets (like 3 or 8) will be granted arbitration to the port and packet 1 will be able to secure arbitration next (as seen in Figure 4.3c and Figure 4.3d). As from the example, it can be seen that PFT neutralises HOL blocking and allows packet 1 to get transmitted ahead of packet 3 and packet 8. As the PFT logic will be active for the packets that were transmitted ahead of packet 1, those will not suffer HOL blocking for long so as to increase the latency of packet 1 further.

PFT therefore allows the packets to be transmitted in the order 9-4-1-3-8, rather than in the order 9-3-4-8-1 as it would occur without PFT. This results in a latency reduction for the highest priority packet 1. Since packet 9 was transmitted ahead of packet 1, HOL blocking can reoccur as packet 9 could be blocked down the line due to its lower priority value. In such a situation, PFT will again be triggered to resolve that block.

Assume the situation where an input port already Priority forwarded is being Priority forwarded by a different packet. In such a situation, the routers would check whether the new Priority forwarded priority value is higher than the current one and if it is higher the value is updated. If the value is lower, it will be trashed but this will not affect the performance as the next time the packet is blocked, Priority Forwarding will again be initiated nevertheless.

4.2. PFT Implementation on the R3 NoC

To evaluate the performance merits of PFT, it was implemented as a NoC model designated the R3 (URL to the source code added in Appendix 2) which was an advancement over the R2 NoC (Hermes based).

The major challenge in PFT implementation in the R3 NoC was dealing with the contention between PFT-flits. At any point of time, there can be new PFT-flits generated at each input port due to the blocking of packets locally simultaneously with remote blocking PFT-flits arriving from nearby routers contenting for the same output port. Furthermore, the PFT-flits arriving from nearby routers will already be causing tailback on the routers up the line elevating the problem further.

To deal with such eventualities, each input port is provided with a local blocking info register called α-register to store the PFT-flit generated at the local input port. To store the PFT-flits arriving from nearby routers, each input port other than the local port is provided with a remote blocking register called β-register.

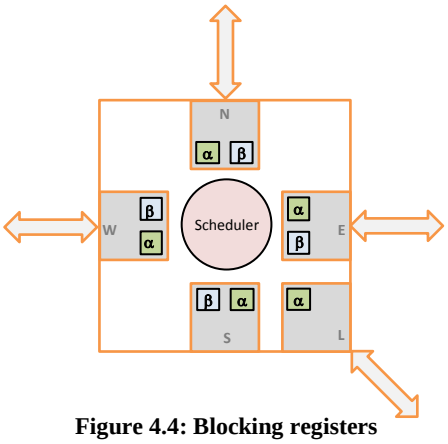


Figure 4.4: Blocking registers

Every time a packet is newly blocked by another blocked packet, a PFT-flit is generated and is stored into the respective α-register. The router is provided with a scheduling logic which uses TDM logic to select an active α-register or β-register one at a time to service. As the data inside the α-register is the local blocking info, every time it is serviced, the data inside it is forwarded to the next router towards which the packet that is blocking the local packet is blocked. The internal functionality of the routers (1,1) and (1,2) from the example is depicted in Figure 4.5.

In Figure 4.5 it can be seen that as packet 1 is blocked by the already blocked packet 4, the blocking information of packet 1 is extracted and stored into an α-register (seen inside the north port of router (1,1)).

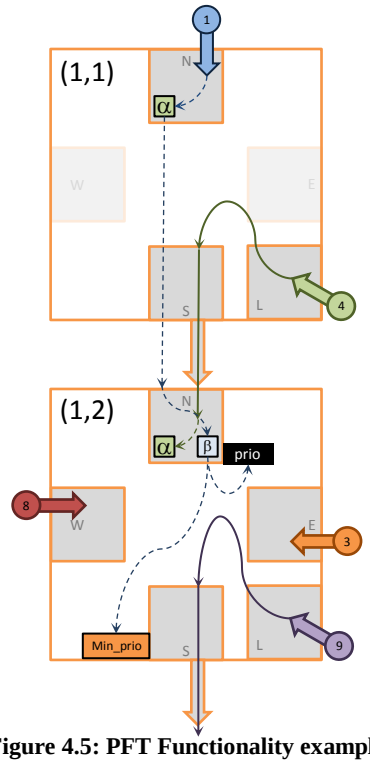


Figure 4.5: PFT Functionality example

To trigger the logic to load the α-register every time when a packet is blocked by a blocked packet, there are two Boolean flags named A and B inside the input ports of each router. The router is provided with logic to store the value of A into B and every time a flit is transmitted, the Boolean value stored in A is inverted. By using such a system whenever transmission is in progress, B will be a clock cycle lagging behind A and both the value will be the same only when the link is idle, hence identifying a blocked output port (registers A and B are not shown in Figure 4.5 for enabling better readability).

As the header of packet 4 is toward the south of the router in the example, when the PFT scheduler services the α-register in the north port, the info is forwarded towards the south to router (1,2) where it is stored into its remote blocking register the β-register (as seen inside the north port of router (1,2)).

When the scheduler services the data inside β-registers, if the blocked header is not at that router, Priority Tunnelling is done and the data is forwarded to the next router and this is continued until the PFT-flit reaches the router with the blocked

header. Once the blocked header is found, Priority Forwarding is performed to resolve the block.

To prevent tailbacking of headers inside FIFO buffers, the prototype also include logic to verify whether a flit is a header before it is injected into a FIFO. This enables the router to update the priority register inside the input port even if the header is not in the head of the queue thus engaging PFT logic even before the header initiates an arbitration request.

Even though theoretically it is possible to share the data lines for both data flits and PFT-flits, the current prototype utilises separate connection lines for ease of implementation.

The pseudo code for the logic operation is added below

Pseudo code

PROCEDURE PFT_loader (At each input port)

// to load blocking data into α-register

IF α-register is unused **AND** output_port is unassigned **AND** port_request is active **AND** requested_output_port is blocked **THEN**

    LOAD arbitration_request_priority, tunnelling_flag and destination_address into the local α-register

// to load blocking data received from the network to β-register

IF input flit is blocking data **AND** β-register is unused **THEN**

    LOAD input flit into local β-register

END PROCEDURE

PROCEDURE PFT_Scheduler

Using a scheduling algorithm

    SET the scheduling pointer to an active α-register or active β-register to be serviced

END PROCEDURE

PROCEDURE Process_α-register_data(for the α-register currently pointed by Scheduler)

// to send α-register data to next router

IF requested_output_port is blocked **THEN**

    SEND α-register data through the requested_output_port

ELSE

DROP α-register data

END PROCEDURE

PROCEDURE Process_β-register_data (for the β-register currently pointed by Scheduler)

// to find the blocked header and update priority

IF output_port is unassigned **AND** port_request is active **THEN**

IF arbitration_request_priority less than β-register_data_priority **THEN**

SET arbitration_request_priority to β-register_data_priority

ELSE

DROP β-register data

// to tunnel output port and to send β-register data to next router

IF tunnelling_flag =1 in β-register data **AND** output_port is blocked **THEN**

FIND future_output_port using XY-routing logic

IF output_port = future_output_port **THEN**

TUNNEL current_output_port

SEND β-register data through current_output_port

ELSE

TUNNEL future_output_port

SET tunnelling_flag to 0 in the β-register data

SEND β-register data it through current_output_port

// for β-register data when tunnelling is disabled (as tunnelling already done)

IF tunnelling_flag == 0 in β-register data **AND** output_port is blocked **THEN**

SEND β-register data through current_output_port

ELSE IF current_output_port is not blocked **THEN**

DROP β-register data

END PROCEDURE

4.3. Experimental Work

As this chapter deals with the resolution of packet starvation (and not latency directly), the chapter quantifies the performance of NoC designs based on the cumulative count of packet reception numbers (thus measuring starvation).

Tests were conducted using a 4x4 size mesh type NoC with a traffic scenario (added as Appendix 1a) where high priority packets 1 and 2 share a heavily congested route.

4.3.1. Varying Load Due to the Increase in Payload Flits

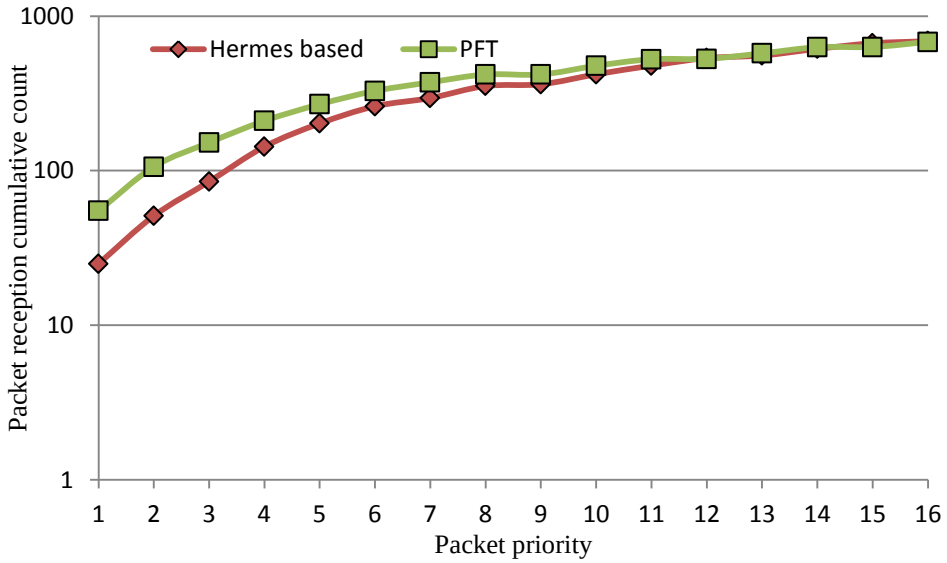


Figure 4.6: Cumulative count of received packets at load V=0.3

Figure 4.6 shows the cumulative count plot for the number of received packets at load V=0.3. As evident from the plot, the reception numbers of the high priority packets are seen higher with PFT compared to the Hermes based NoC. This is due to the resolution of HOL blocking by PFT while there will be high priority packets still waiting for transmission inside the packet generators of the Hermes based NoC.

The load level on the NoC was then increased to V=0.5 by varying the packet size (thus increasing the number of payload flits in the NoC at any point of time). The resultant plot is added as Figure 4.7.

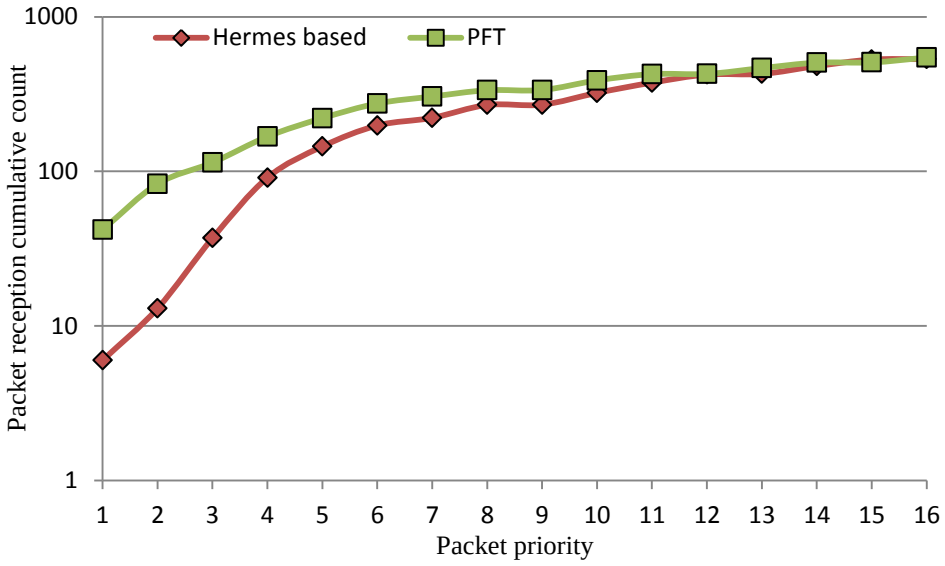


Figure 4.7: Cumulative count of received packets at load V=0.5

The effect of the increased load is evident from the plot as the number of packets successfully received with priority 1 dropped from 25 (in Figure 4.7) to six. However with PFT, the packet reception number is seen almost constant.

With further increase in load to V=0.7, packets 1 and 2 are seen to get blocked completely with the basic NoC due to HOL blocking as seen in Figure 4.8.

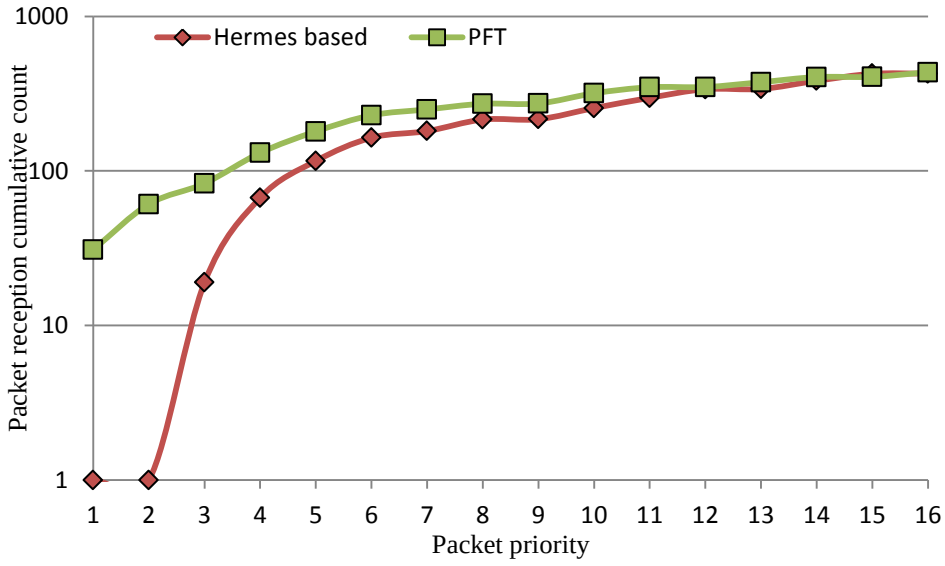


Figure 4.8: Cumulative count of received packets at load V=0.7

The effect of PFT on the traffic is quite evident as the reception numbers for the high priority packets are seen to drop only slightly from V=0.5.

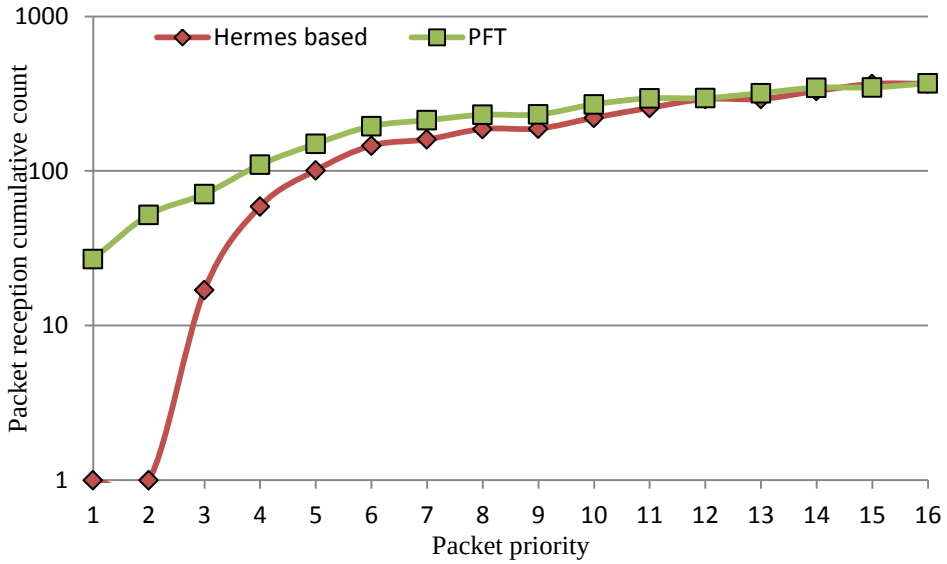


Figure 4.9: Cumulative count of received packets at load V=1

As seen in Figure 4.9, the situation is similar with the increase in load to V=1 as the packet reception numbers show only minor variation with PFT.

4.3.2. Varying Load Due to the Increase in Packet Numbers

To verify the effect of the increase in load on the NoC due to the increase in packet numbers, a 4x4 NoC was tested with increasing load by varying the packet period. The cumulative count of packet reception with load V=0.3, 0.5, 0.7 and 1 are presented in Figure 4.10, Figure 4.11, Figure 4.12 and Figure 4.13 respectively.

Although the tests reveal similar performance characteristics, the notable difference between the results of this section is that despite an increase in the load on the NoC the reception numbers of the high priority packets are seen to be unaltered with PFT.

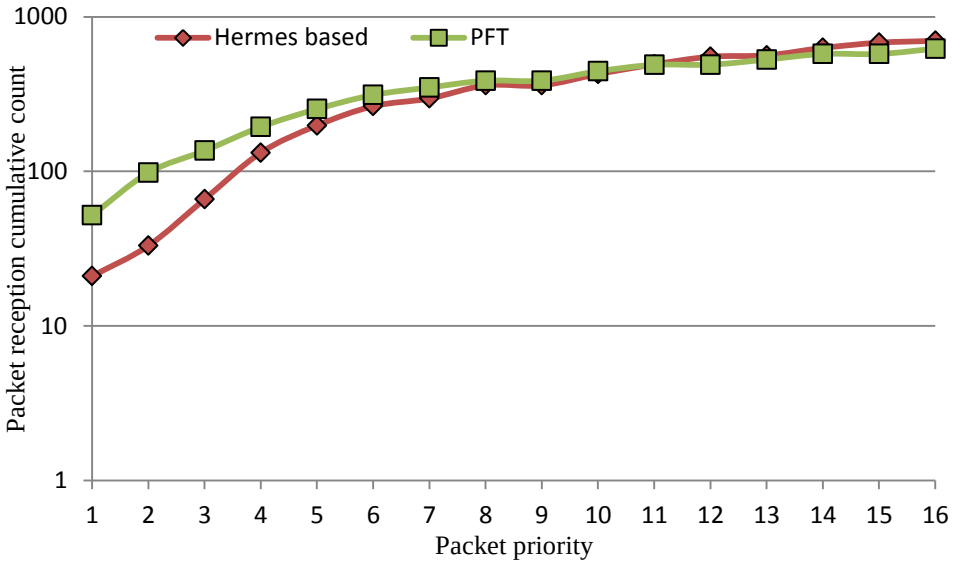


Figure 4.10: Cumulative count of received packets at load V=0.3

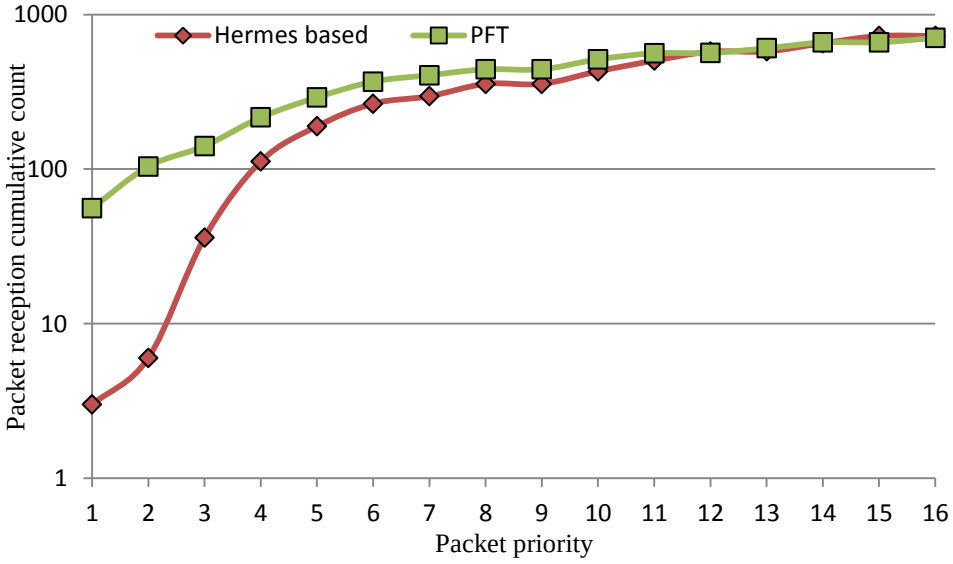


Figure 4.11: Cumulative count of received packets at load V=0.5

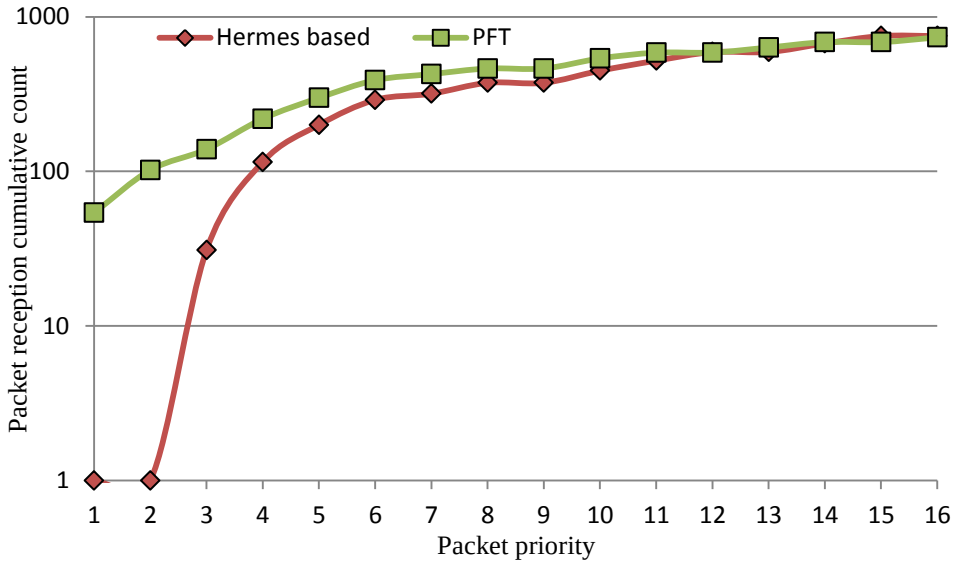


Figure 4.12: Cumulative count of received packets at load V=0.7

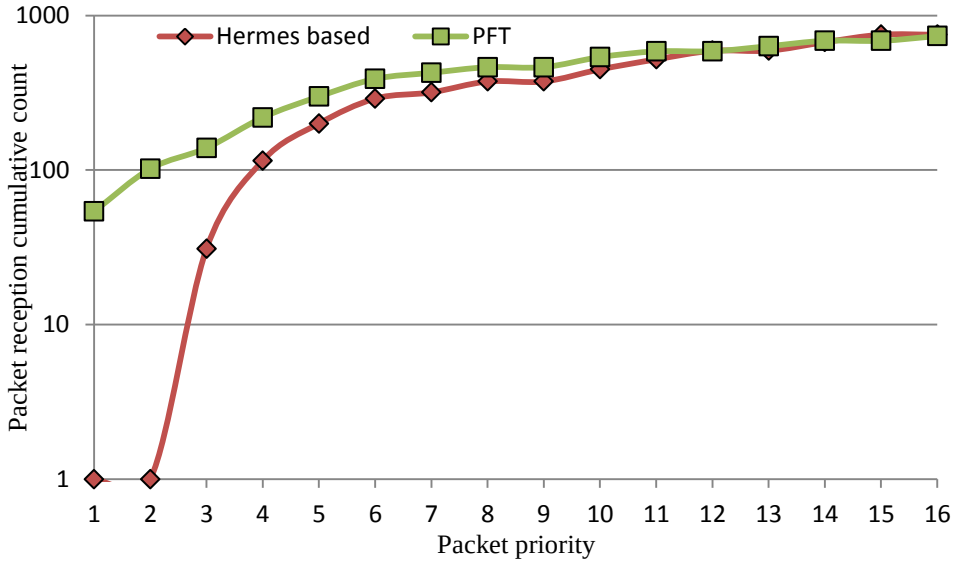


Figure 4.13: Cumulative count of received packets at load V=1

PFT does not need additional hardware with increase in size of the NoC hence ensuring scalability. For example, if the size of the NoC was increased resulting in increased number of packet priorities, the plot will be similar to the one seen before with the difference that the two lines (depicting Hermes based and PFT) would overlap further down the priority range. In Figure 4.13, it can be seen that the cumulative count of the Hermes based NoC and PFT overlap at packet priority 12. If the same test was conducted with a bigger NoC resulting in more number of

packet priorities, this overlap would have happened further down the line depending of the number of packet priorities.

The performance of PFT at different load levels can be seen in Figure 4.14 and Figure 4.15 where the cumulative count of received packets are plotted. Figure 4.14 shows the plot where the load was increased by increasing the number of payload flits in the NoC and Figure 4.15 depict the comparison when the load was increased by increasing the packet numbers.

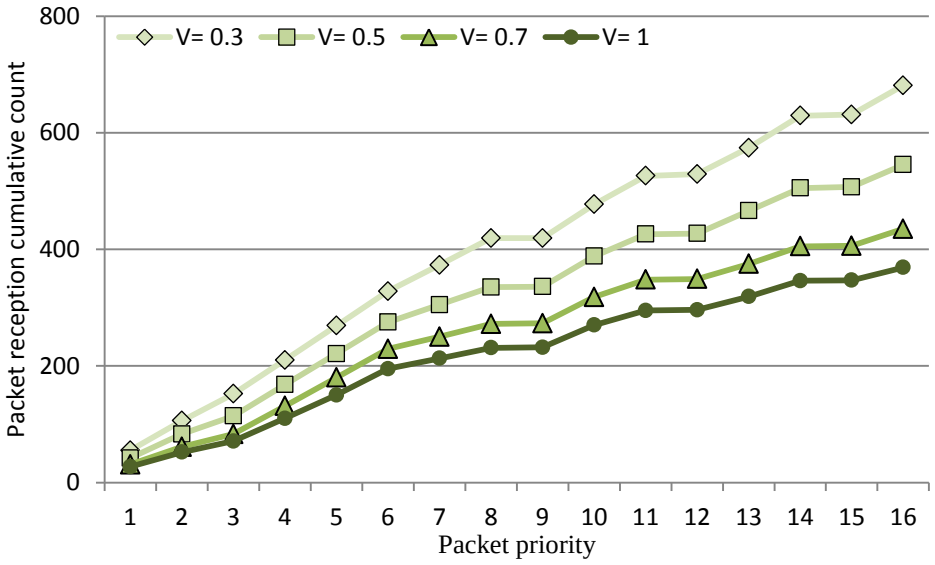


Figure 4.14: Packet reception cumulative count with PFT due to the increase in payload flits

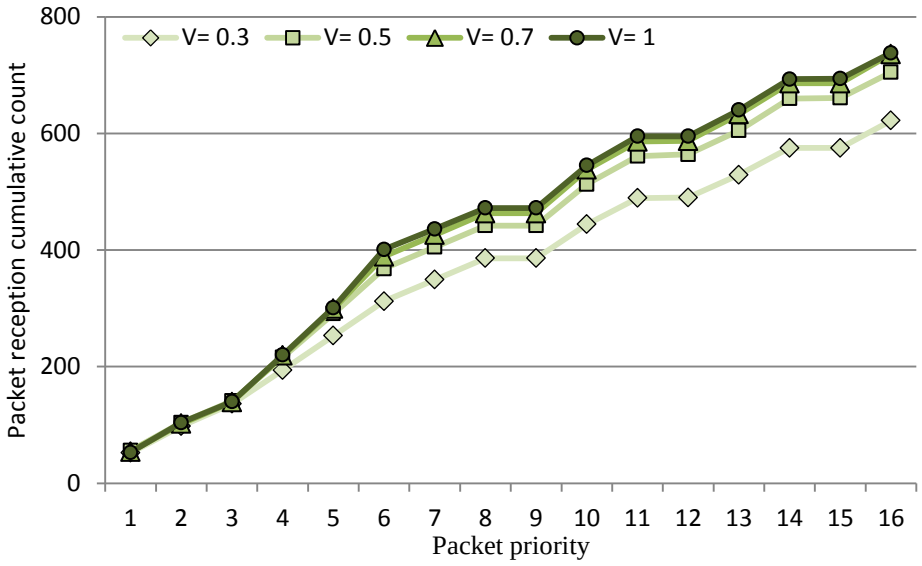


Figure 4.15: Packet reception cumulative count with PFT due to the increase in packet numbers

It can be noted that the performance variation of the high priority packets is considerably less due to the increase in packet numbers than increase in payload flit numbers. Also, with the increase in load on the NoC due to the increase in headers the reception numbers for low priority packets are seen to increase contrary to the previous approach.

With Hermes based NoC, the increase of load on the NoC due to payload flits seems to have a linearly scaled effect on packet reception numbers (lines at different load levels are seen almost parallel to each other) as seen in Figure 4.16.

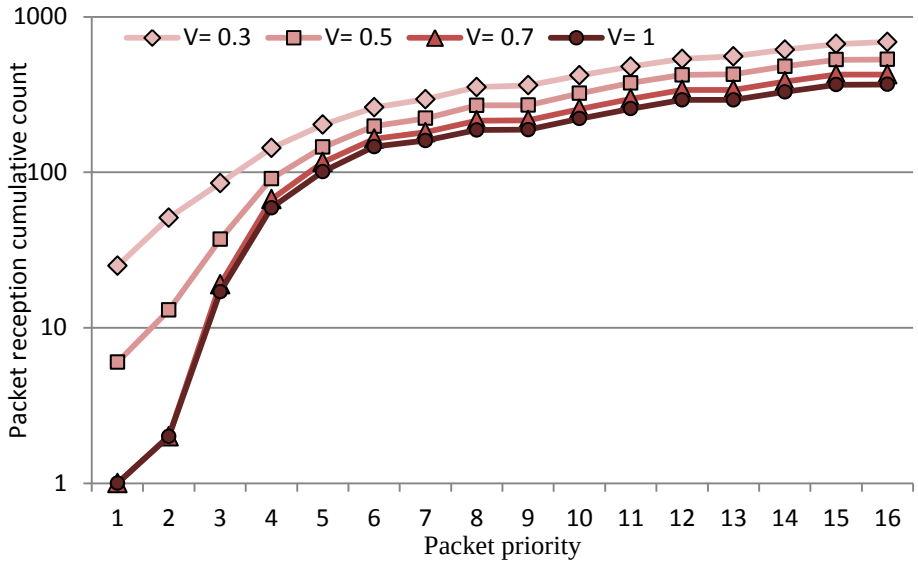


Figure 4.16: Packet reception cumulative count with Hermes based NoC due to the increase in payload flits

However with the increase on load due to the increase in packet numbers, the performance variation gets magnified as seen in Figure 4.17.

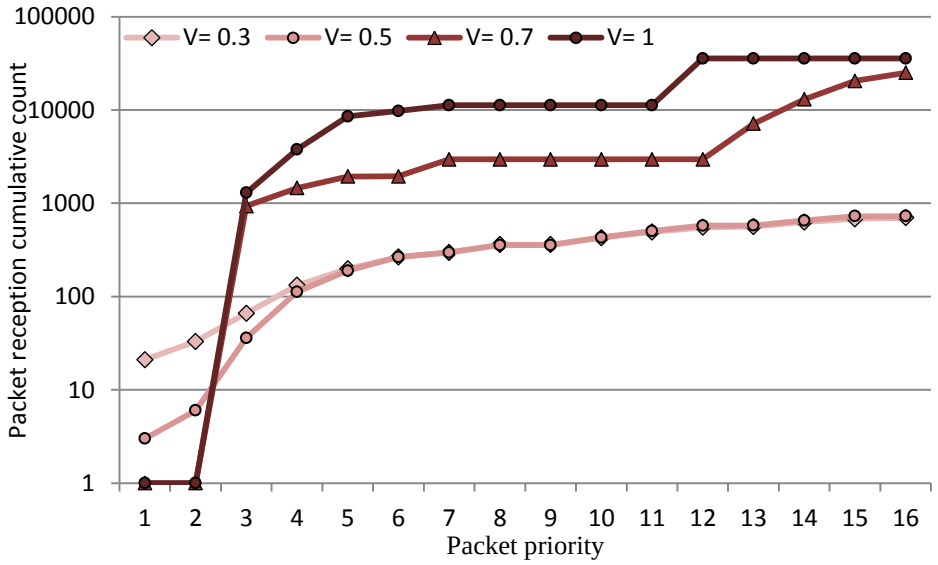


Figure 4.17: Packet reception count with Hermes based NoC due to the increase in packet numbers

Although, PFT was shown to improve latency of packets (shown in Figure 4.18), the advantage was seen to be limited due to tailbacking of packets (explained in Chapter 5).

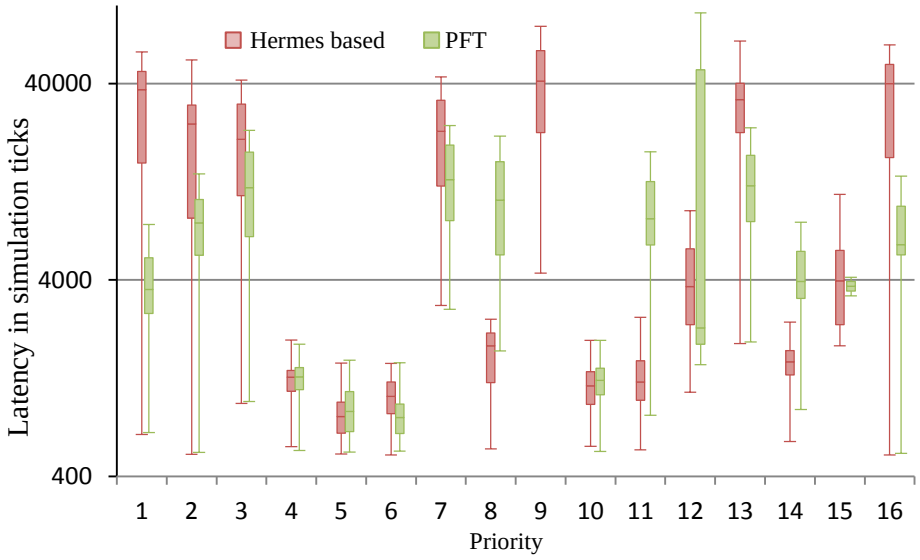


Figure 4.18: Packet reception latency boxplot

In the figure it can be seen that packets like 1,2,3 and 7 has lower and shorter box and whiskers depicting lower magnitude and variation in latency.

4.3.3. Performance Variation with Packet Size

To evaluate how packet size variation (with respect to priority) affect the performance, a 4x4 NoC was evaluated with packet size increasing proportionally and inversely proportionately with priority. The resultant cumulative count plots at a low load of V=0.4 are added as Figure 4.19 and Figure 4.20.

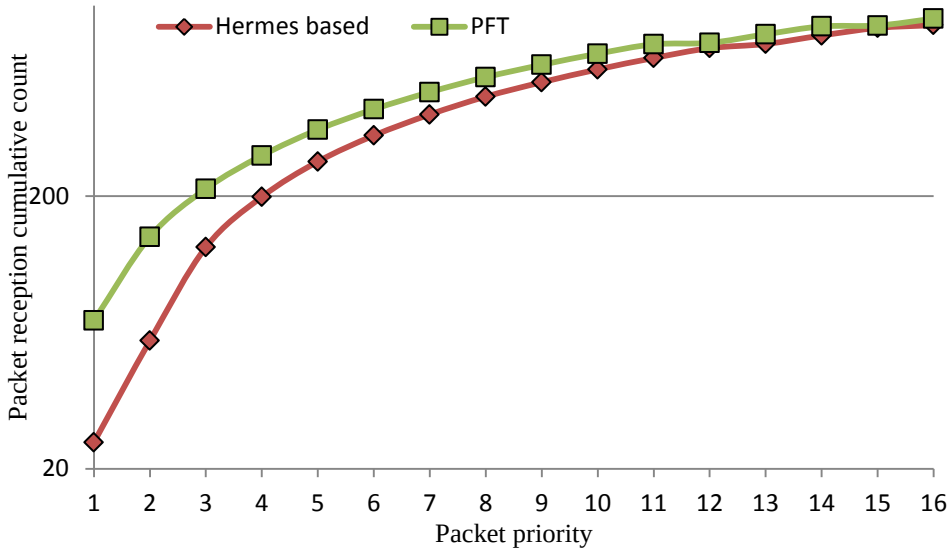


Figure 4.19: Cumulative count plot with packet size scaled proportionately to priority with V=0.4

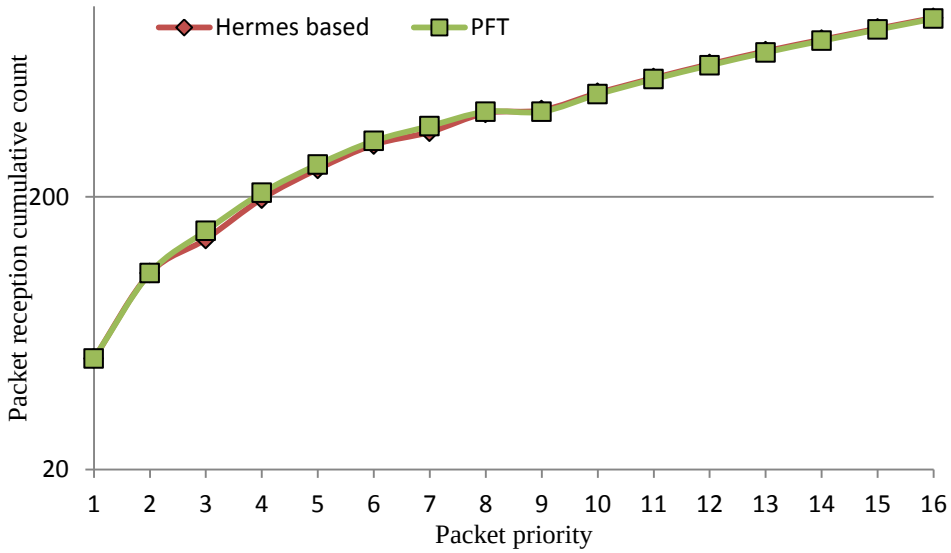


Figure 4.20: Cumulative count plot with packet size scaled inversely proportionately to priority with V=0.4

From the results it is evident that PFT is more effective when the high priority packets are shorter than lower priority packets. When the high priority packets are longer than low priority packets, both Hermes based NoC and PFT had similar performance under low load as HOL scenarios would have been very rare (Figure 4.20).

As the load was increased to V=0.8, starvation of packets were encountered with the Hermes based NoC as in previous experiments (evident from Figure 4.21).

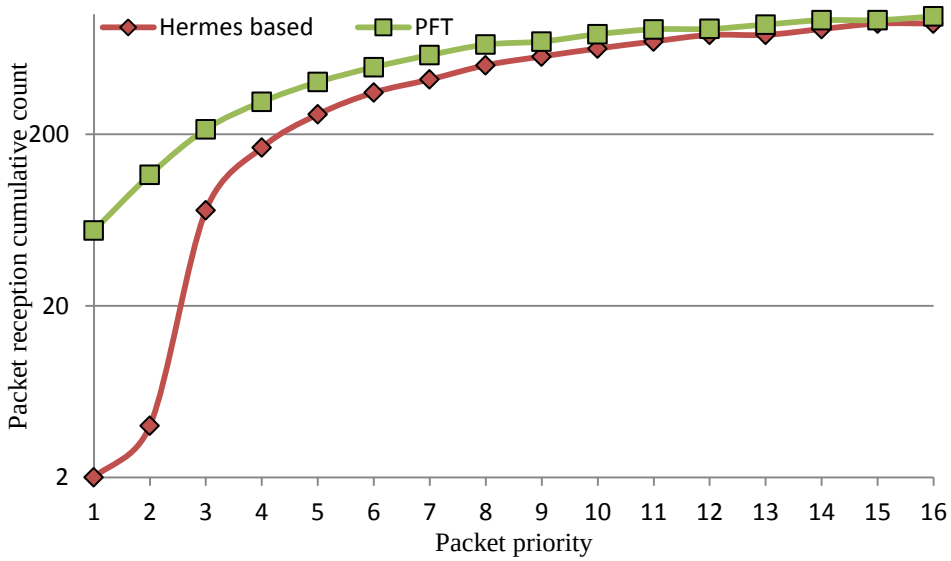


Figure 4.21: Cumulative count plot with packet size scaled proportionately to priority with V=0.8

Figure 4.21 shows the performance of the system where the high priority packets where shorter than the low priority ones. Similar to the previous experiment, the effectiveness of PFT seem to be similar to Hermes based NoC when the low priority packets were shorter than the high priority ones as seen in Figure 4.22.

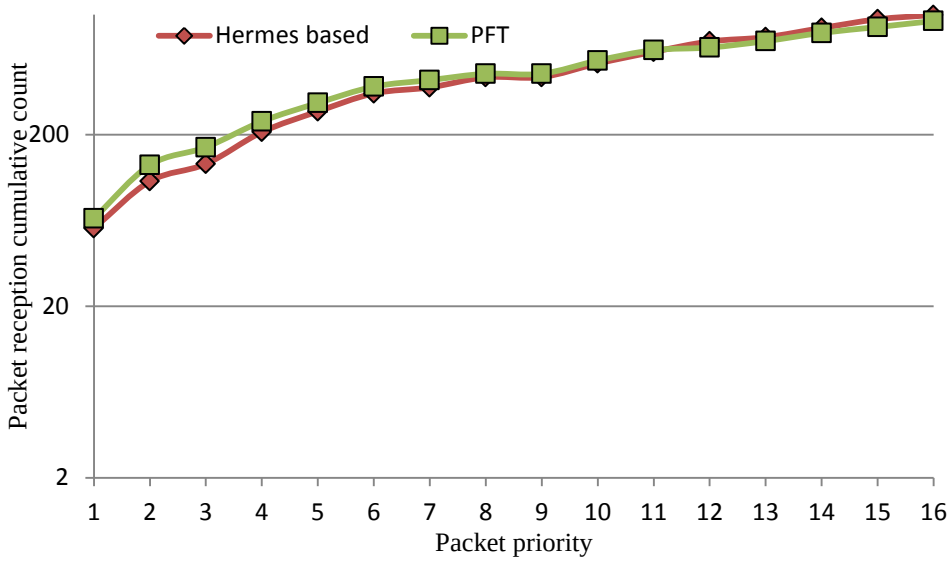


Figure 4.22: Cumulative count plot with packet size scaled inversely proportionately to priority with V=0.8

4.3.4. Limitations

Although PFT improved packet reception numbers by resolving starvation caused by HOL blocking, the technique can be ineffective with random traffic without hotspots. A hotspot is defined in the thesis as follows.

> ***Traffic hotspot:*** *Connection link on the NoC which is shared by a significantly higher number of packets than the average case.*

With random traffic without any hotspots, the occurrence of HOL blocking becomes a less common phenomenon thus decreasing the instances where the technique will be triggered. Also, due to the occurrence of tailbacking, the advantages in the magnitude and variation of latency is limited.

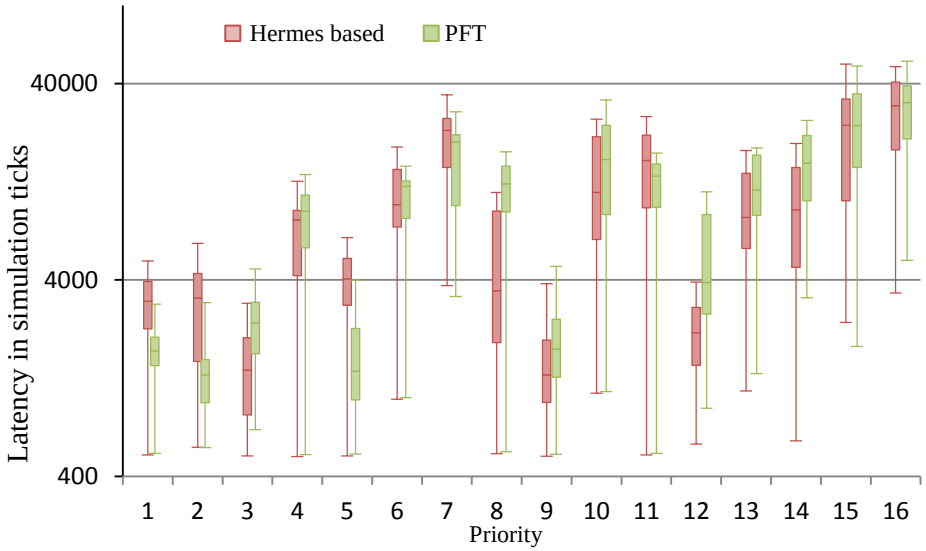


Figure 4.23: Latency plot

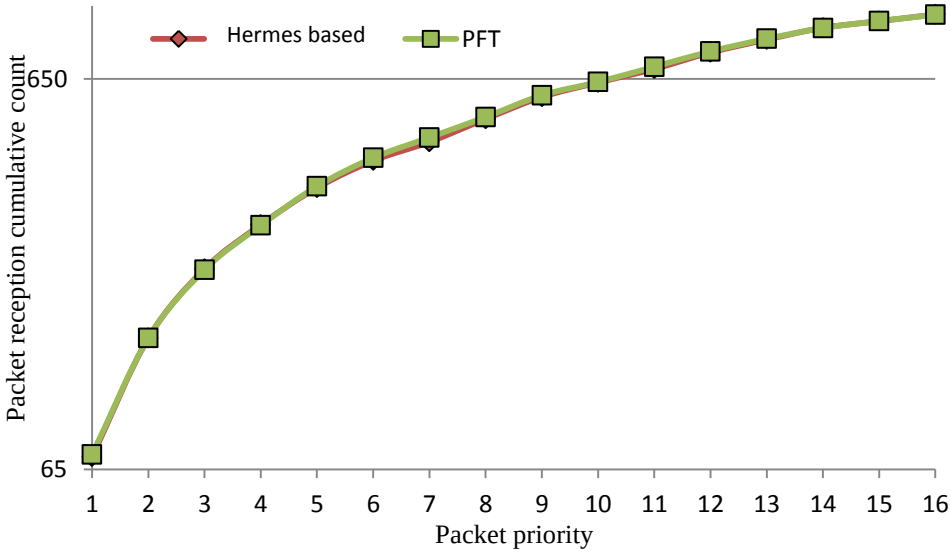


Figure 4.24: Cumulative count plot

For example, the latency box plot of a traffic scenario without hotspots is shown in Figure 4.23. It can be seen that the packets 1, 2, 5 and 6 receive minor advantages with PFT however the packet reception numbers remain identical (Figure 4.24).

4.4. Hardware Overhead

The hardware overhead evaluation show that the baseline router (2-position input buffers) based on Hermes with non-preemptive priority based arbitration utilised 1209 LUTs and 710 Slice registers of the chosen FPGA.

However the PFT enabled NoC (2-position input buffers) utilised 2096 LUTs and 1236 Slice registers. Detailed hardware overhead details is added in Appendix 3 in sections R2 and R3.

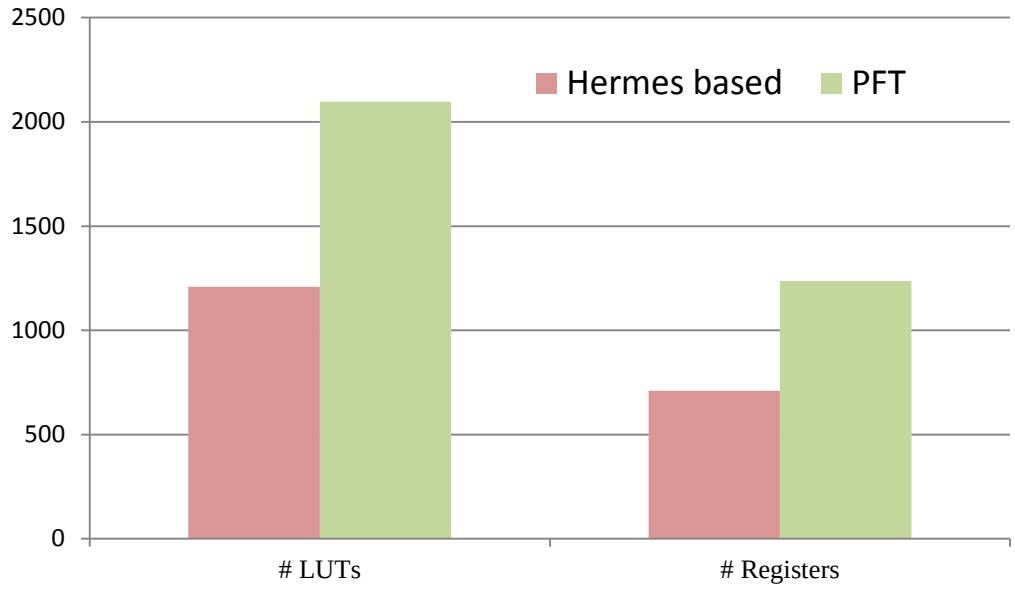


Figure 4.25: Hardware overhead

4.5. Summary

The chapter introduced the PFT technique using which HOL blocking could be neutralised, thus countering starvation of packets due to lower priority packets. This was achieved by forwarding the priority of the HOL blocked packet from behind the line to the router where HOL blocking is occurring using dedicated connection links to transmit PFT flits (control messages). This enabled the router to update the priority of the arbitration request of the low priority packet with the priority of the higher priority packet blocked up the line and hence resolve the block.

While transmitting the PFT flit from router to router, the routers also would lock the output ports that will be used by the packet blocked up the line in future so that other packets of lower priority would not get arbitration to those ports until the packet is transmitted.

Tests using an HDL coded model reveal advantages in packet reception numbers under HOL blocking. Test were conducted with increased load levels by increasing the number of payload flits as well as header flits in the NoC at any point of time. Both tests revealed improvement in packet reception numbers depending on packet priority compared to a Hermes based NoC with the latter case revealing lower variation in reception numbers with the increase in load.

As seen in the literature review, HOL blocking is typically resolved using techniques like VCs or LDM which provides spatial isolation or by using TDM which provide temporal isolation. While spatial isolation based techniques have high hardware requirements, temporal isolation limits the dynamic behaviour and scalability of the NoC.

PFT however uses a dynamic approach which modifies the arbitration policy of routers to achieve the goal. As a result, PFT requires low hardware resources (unlike VC or LDM) and is dynamic and scalable (unlike TDM). In typical scenarios with hotspots, PFT will be effective due to the occurrences of HOL-blocking however in uniform random traffic, the advantages is limited due to the limited number of HOL blocking scenarios.

This motivated the research described in the next chapter where tailbacking is resolved in non-preemptive routers by manipulating packets and thus enhancing predictability.

Chapter 5

Predictability Enhancement by Packet Splitting

With non-preemptive NoCs with packet priorities, tailbacking (defined in Section 3.2) of high priority packets by low priority packets will be a frequent phenomenon, thus causing variation and increase in magnitude of latency. Typically, tailbacking is resolved in dynamic systems with preemptive arbitration using Virtual Channels, which are hardware expensive and have scalability issues. This chapter aim to resolve tailbacking situations by splitting data packets thereby realising a low hardware overhead emulation of a preemption functionality.

5.1. Selective Packet Splitting

Selective Packet Splitting (SPS) is aimed to resolve tailbacking scenarios by splitting lower priority packet flows and thereby provide better predictability to high priority packets than regular non-preemptive NoCs. As SPS emulates preemption functionality by splitting packets, the hardware requirements are comparatively less than the Virtual Channel approach.

Consider the HOL blocking example in Figure 2.4 in page 15 where packet 9 is tailbacking (or blocking) packet 3 despite the higher priority value. Similarly, there is tailbacking of packet 1 due to packet 4. As a result packet 3 will have to wait until packet 9 gets transmitted even at the best of times hence increasing its latency. As packet 1 is being tailbacked by packet 4 which is tailbacked by packet

3 which itself is tailbacked by packet 9, packet 1 will have to wait until the packets 9, 3 and 4 get transmitted completely despite having the highest possible priority value.

With Virtual Channels and priority preemption, there will be separate service levels and a higher priority service level will be able to preempt and transmit flits through a link even if the link is being used by a lower priority service level packet.

Instead of using expensive Virtual Channel hardware, in SPS the logic splits the lower priority communication into two so that once the initial part of the lower priority packet (which is tailbacking the higher priority packet) is transmitted, the high priority packet can be transmitted followed by the remaining part of the low priority packet. As a result, the system does not need extra connection lines for communicating between routers as they utilise the arbitration logic already present in the router.

To enable splitting, the most significant bit (MSB) of every flit is designated as the tail flit indicator so that the router can terminate a communication by splitting the flow using a tail flit indicator. Thus by splitting a lower priority packet (which is tailbacking a higher priority packet), the higher priority packet will be able to secure arbitration before the lower priority packet is transmitted completely. The router also issues a new arbitration request and header for the rest of the low priority packet so that once the high priority packet is transmitted; the remaining part of the low priority packet can be transmitted. The flowchart of SPS operation is shown in Figure 5.1.

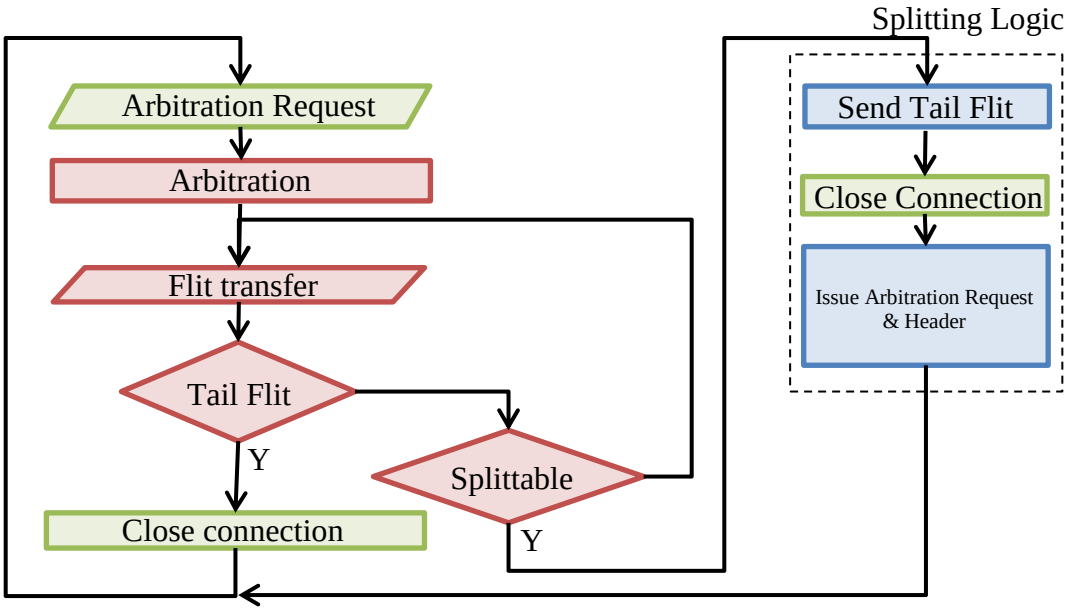


Figure 5.1: Operational flowchart

To denote a splitted communication, each input port will have a boolean register called ‘split flag’. Once the tail flit for the low priority packet is sent, the output port is released by closing the communication and the split flag is set denoting a split communication. The router will also issue a new arbitration request for the splitted low priority packet so that the rest of the packet will be transmitted when the output port becomes free.

The input port also will have registers to store the destination address of the split packet so that when the splitted packet gets arbitration next, a new header can be formulated and send followed by the payload. The split flag register is hence used by the state machine to identify a splitted communication from an intact packet so that a new header can be issued and send if it is a splitted packet.

As an example, consider packet 9 and packet 3 in the scenario depicted in Figure 2.4 in page 15.

With packet splitting enabled in the routers; router (1,2) will stop transmission of packet 9 and then send a tail flit down the line so that the routers down the line terminates the connection automatically, hence eliminating the need for control lines or control flits. Then the router releases the south output port so that packet 3

will be able to secure arbitration and hence get transmitted through that port. Simultaneously, the router also issues a new arbitration request and header for the rest of packet 9 so that once the port is free, the remaining flits of packet 9 can be transmitted. In this example, this will allow packet 4 to get arbitration after packet 3 while the remaining part of packet 9 waits for arbitration. As packet 1 will already have caused splitting of packet 4 at router (1,1), once the initial part of packet 4 gets transmitted, packet 1 will be able to secure arbitration while the final section of packet 9 and packet 4 waits for arbitration.

5.2. Priority Forwarded Packet Splitting

Even though in the example in Figure 2.4 in page 15, SPS enabled packet 1 to get transmitted before packet 9 and 4, it still had to wait until packet 3 gets transmitted completely as packet 1 was HOL blocked by packet 3. This prevented packet 1 from getting to router (1,2) hence preventing it from splitting packet 3.

As seen in Chapter 4, under intense load, HOL blocking can cause total starvation of high priority packets and to resolve this, the Priority Forwarding technique (previously seen in Chapter 4) was combined with SPS to eliminate both tailbacking and HOL blocking.

With Priority Forwarded Packet Splitting (PFS), both Priority Forwarding and packet splitting occurs when a packet is blocked by a lower priority packet so that both HOL blocking and tailbacking could be neutralised. Consider the example in in Figure 2.4 in page 15. With PFS, packet 4 will be forwarded with the priority value 1 hence enabling it to split packet 3 which will have already secured arbitration by splitting packet 9. This will allow complete transmission of packet 1 before all of the other packets.

5.3. PFS Implementation on the R7 NoC

To test PFS, the technique was implemented as a Bluespec System Verilog coded model designated as the R7 NoC (URL to the source code added in Appendix 2).

The basic architecture and functionality of R7 routers is similar to the R2 router explained in Figure 3.8 in page 52.

Upon reception of a packet header (similar to R2), the input port evaluates the destination information in the header and sets the ‘arb_request’ register with the required output port id using XY-routing algorithm. The input port also stores the priority of the header into the priority register so that the arbitration unit will be able to evaluate all arbitration request to each output port and then provide arbitration to the qualifying input port by setting the appropriate ‘out_port’ register.

This enables the input port to send flits through the communication link and unlike the R2, the last flit of each R7 data packet has a bit reserved designating the tail flit. So, the R7 routers are designed to terminate communication upon reception of a tail flit.

The functionality of R7 NoC routers that enable SPS is shown in Figure 5.2.

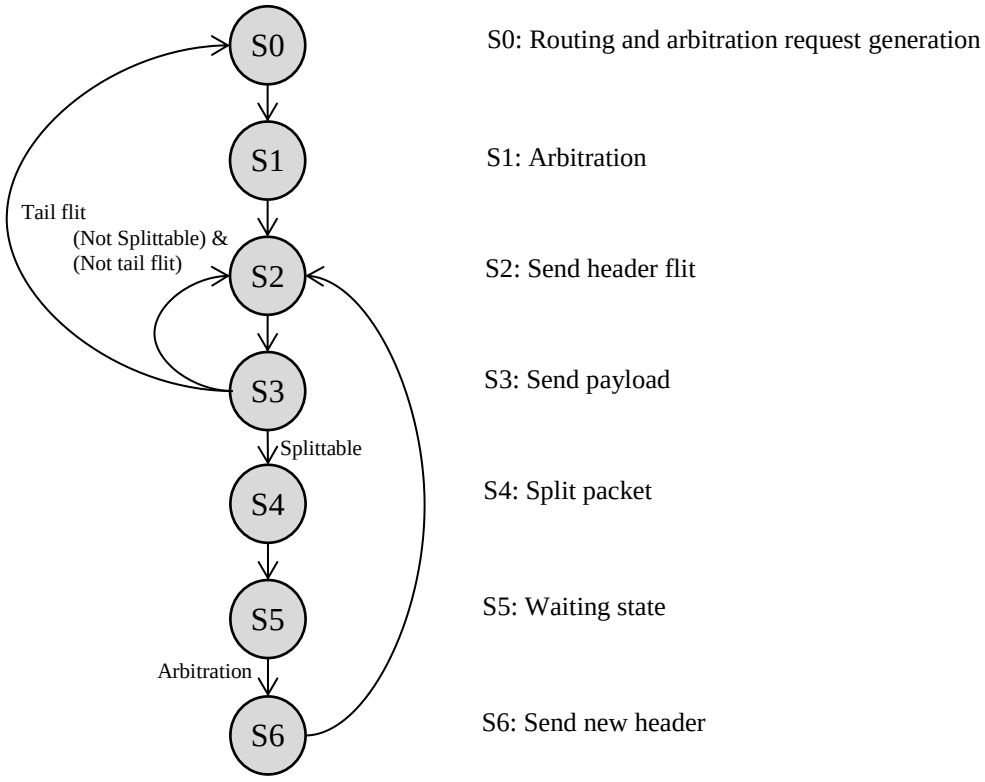


Figure 5.2: SPS implementation on the R7 NoC.

State 0 is responsible for routing and arbitration request generation and State 1 is for issuing arbitration. State 2 deals with sending of the header flit and State 3 deals with sending of the payload flits and this continues until the tail flit is encountered or when the port detects a splittable scenario.

If the port detects a splittable scenario, the port will go into the splitting state (State 4) where the 'out_port' register will be reset and 'arb_request' register will be set along with splitted flag. Then the port will go into the waiting state (State 5) waiting for arbitration similar to how it would in State 1. When the required output port gets available and on getting arbitration, the port will go into State 6 where a new header will be formulated and sent following which it returns to State 3 for payload delivery.

To enable the packet generator/receptor module to identify the original tail flit, along with the tail flit denoting flag which denote the tail of a packets (even for split ones), a bit is reserved in the tail flit denoting the final flit of the whole packet.

The pseudo code for the logic is added below.

Pseudo code

```
PROCEDURE Arbiter (for each input port [on state S1 or S5])
IF qualified_for_arbitration THEN
        ARBITRATE (by setting out_port register)
        IF NOT(splitted flag) THEN
                SET STATE to S2  // to send header
        ELSE
                SET STATE to S6 // to formulate and send a new header
        END IF
END IF
END PROCEDURE

PROCEDURE send_flit (at each input port [on state S2 or S3 or S4 or S6])
IF tail_flit THEN
        END CONNECTION
ELSE
        IF state = S2 THEN SEND header
```

```
        ELSE IF state = S6 THEN FORMULATE new header and SEND
        ELSE IF state = S3 THEN SEND payload_flits
        ELSE IF state = S4 THEN
                SET STATE to S5 // waiting state
                SEND tail_flit
                ENABLE splitted_flag
        END IF
END IF
END PROCEDURE
```

5.4. Experimental Work

This section presents the experimental results of the PFS based NoC (R7) compared to a basic Hermes based (R2) NoC. Both the NoCs are tested for its latency variability using box plots with four random traffic scenarios and are tested with varying load levels both due to the increase in payload flits in the NoC and due to increase in packet numbers (header flits) in NoC.

The NoC were also tested with varying packet sizes and finally the chapter presents the hardware requirement evaluation for the NoCs.

5.4.1. Random Traffic

The latency plot of a 4x4 NoC with four random traffic scenarios (described in Appendix 1b to 1e) is presented in Figure 5.3, Figure 5.4, Figure 5.5 and Figure 5.6.

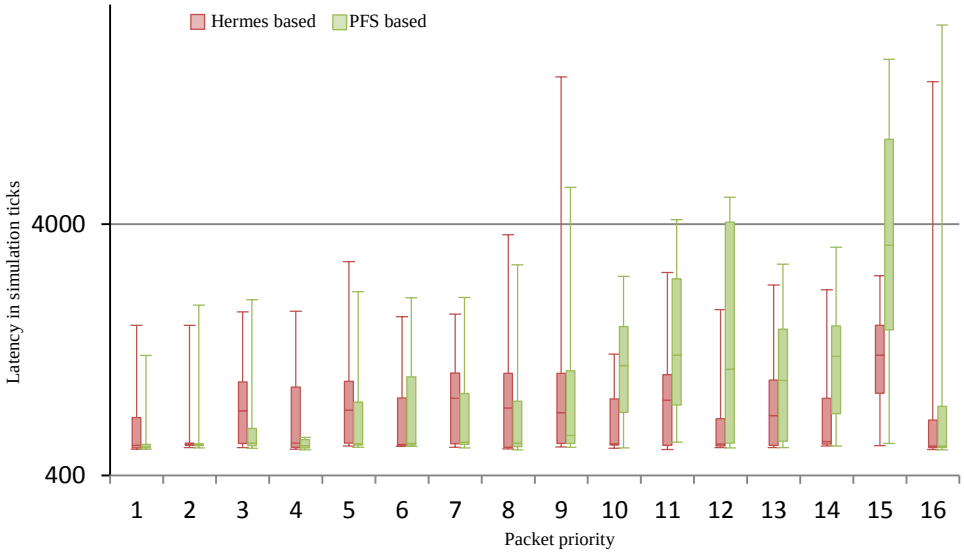


Figure 5.3: Latency performance with random traffic 1

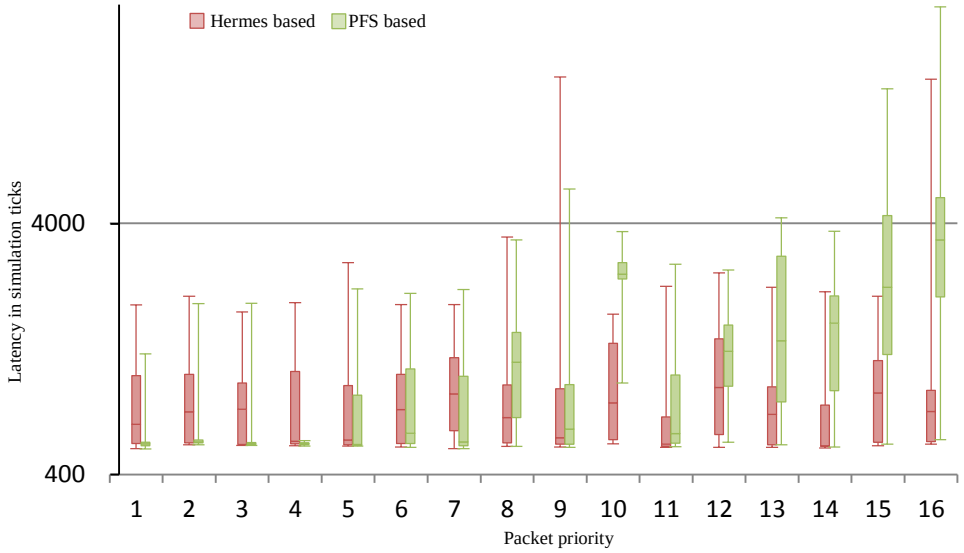


Figure 5.4: Latency performance with random traffic 2

From the plots, the effect of PFS is evident as the high priority packets (like 1 to 8) are seen to suffer lower magnitude and variation in latency depicted by the lower and shorter box and whiskers.

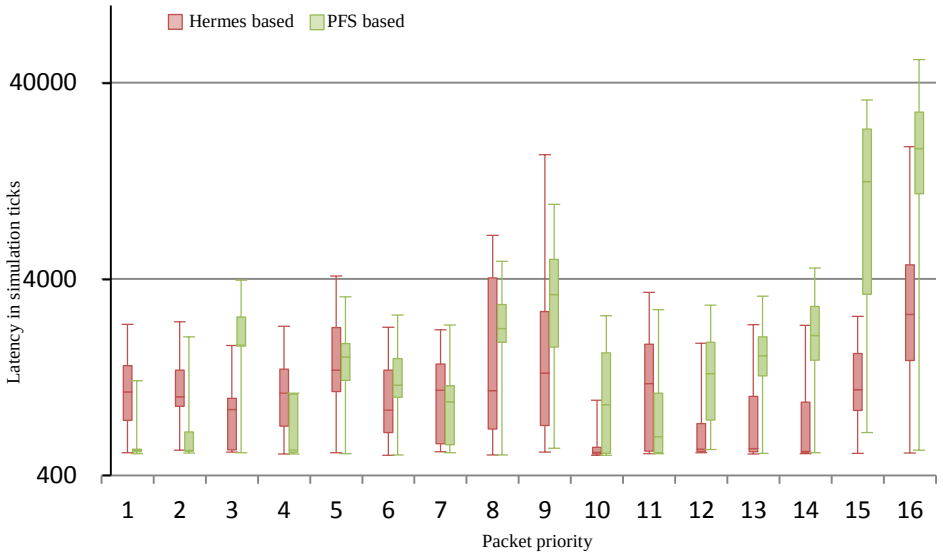


Figure 5.5: Latency performance with random traffic 3

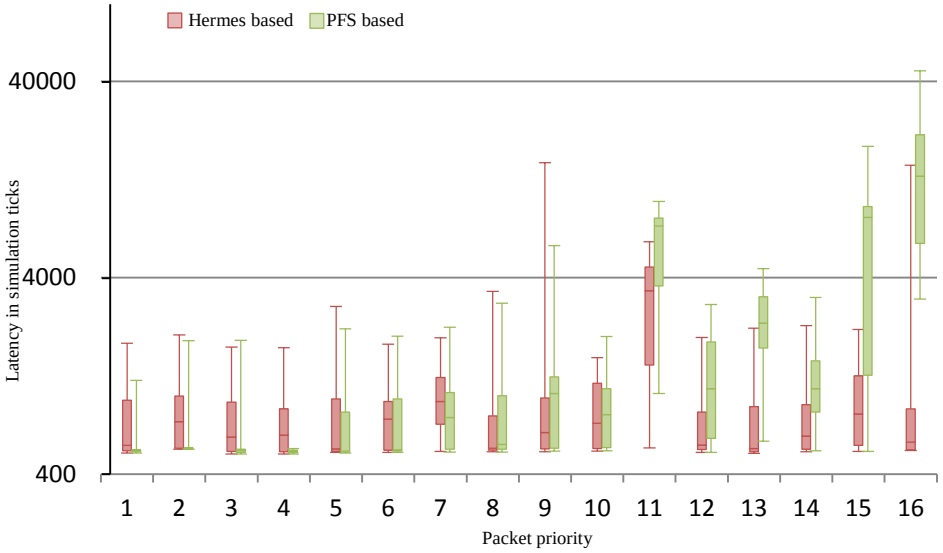


Figure 5.6: Latency performance with random traffic 4

Due to the latency improvement brought about to the high priority packets, the latency performance of the packets in the lower spectrum of the priority range (priorities 12 to 16) are seen to get worse than the Hermes based NoC. Although this is acceptable due to the lower priority range of the associated packets, further research will look into moderating those by trading residual slack from higher priority packets. The details of the associated research is presented in Chapter 6.

5.4.2. Varying Load Due to the Increase in Payload Flits

To verify the performance of PFS due to the increase in the load due to payload flits, a 4x4 NoC was tested with increased load level by increasing the packet sizes. The resultant latency statistics at load V= 0.4, 0.6, 0.8 and 1 are presented in Figure 5.7, Figure 5.8, Figure 5.9 and Figure 5.10.

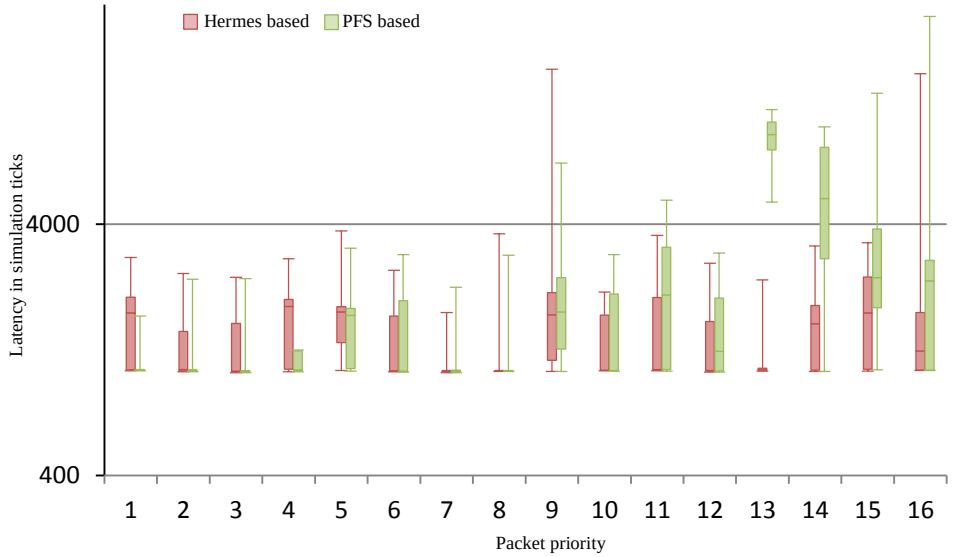


Figure 5.7: Latency performance with random traffic at V=0.4

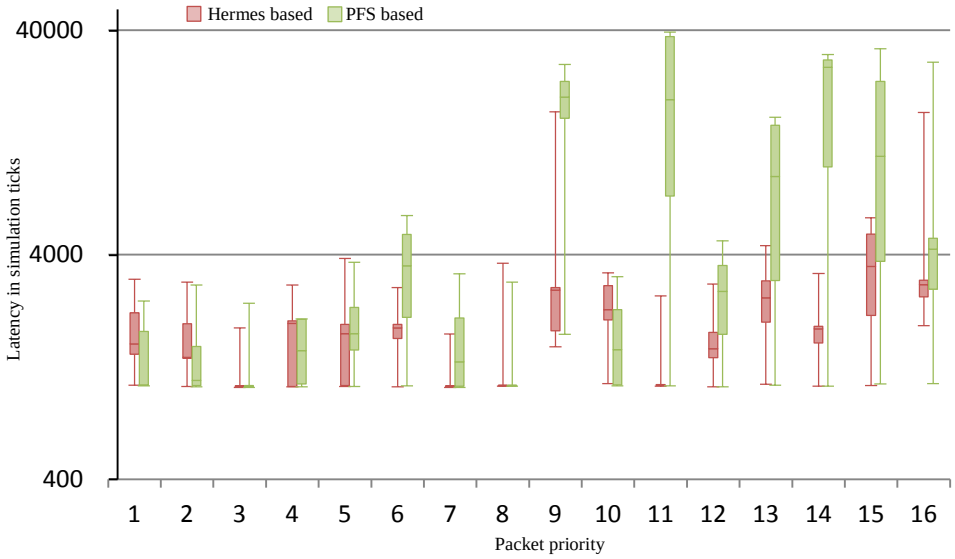


Figure 5.8: Latency performance with random traffic at V=0.6

It can be seen that with increase in load in the NoC from 0.4 to 0.6 and 0.8, the variation in latency of higher priority packets (like 1 to 4) are less with PFS compared to the Hermes based NoC. However this results in higher magnitude and variation in latency of the lower spectrum of the priority range (13 to 16).

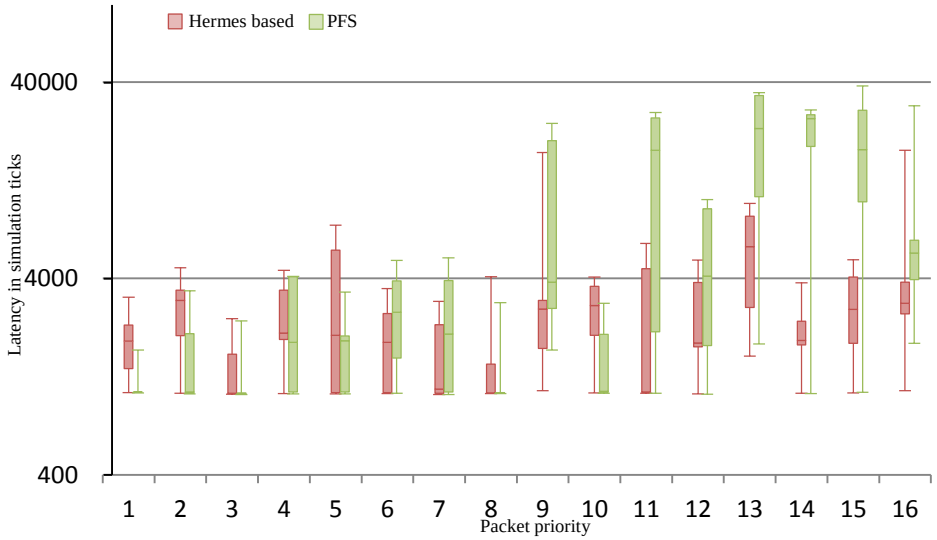


Figure 5.9: Latency performance with random traffic at V=0.8

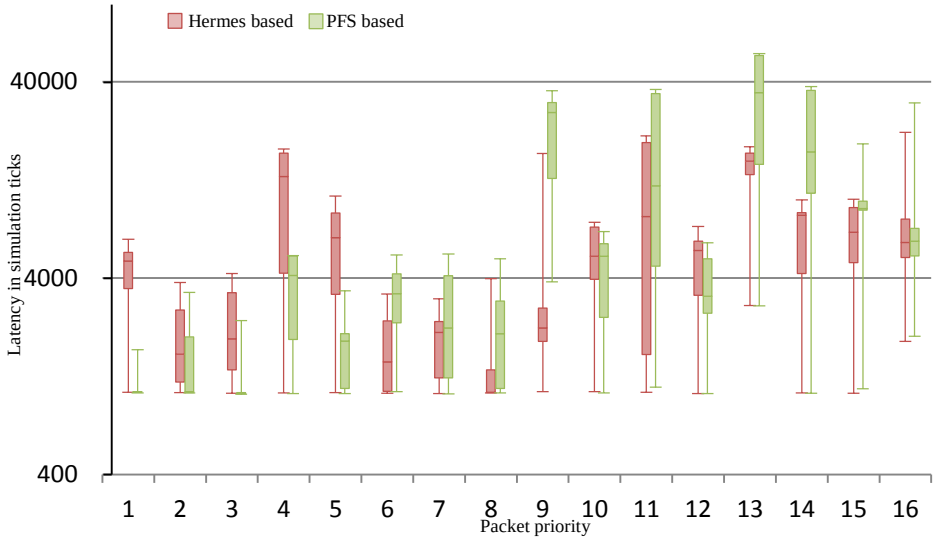


Figure 5.10: Latency performance with random traffic at V=1

Similar to the experiments that were conducted in the previous section, PFS was seen to improve the magnitude and variation in latency of high priority packets depicted by the lower and shorter box and whiskers. Despite the increase in load on the NoC due to the increase in payload flits, the latency improvement to high priority packets is visible.

5.4.3. Varying Load Due to the Increase in Header Flits

The system was also tested with increased load levels by increasing the number of packets numbers (header flits) in the NoC. This was done by reducing the packet periods and the resultant latency performance at load V= 0.4, 0.6, 0.8 and 1 are presented in Figure 5.11, Figure 5.12, Figure 5.13 and Figure 5.14.

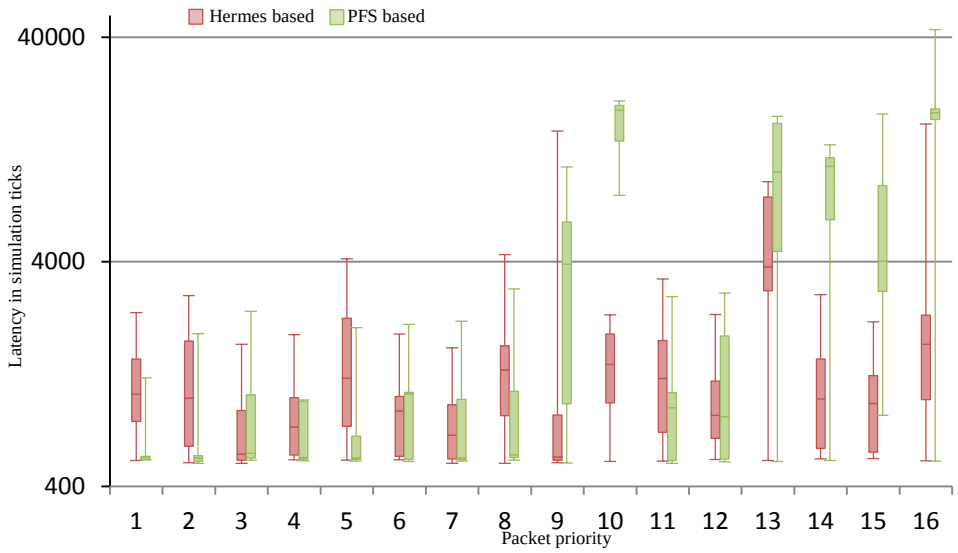


Figure 5.11: Latency performance with random traffic at V=0.4

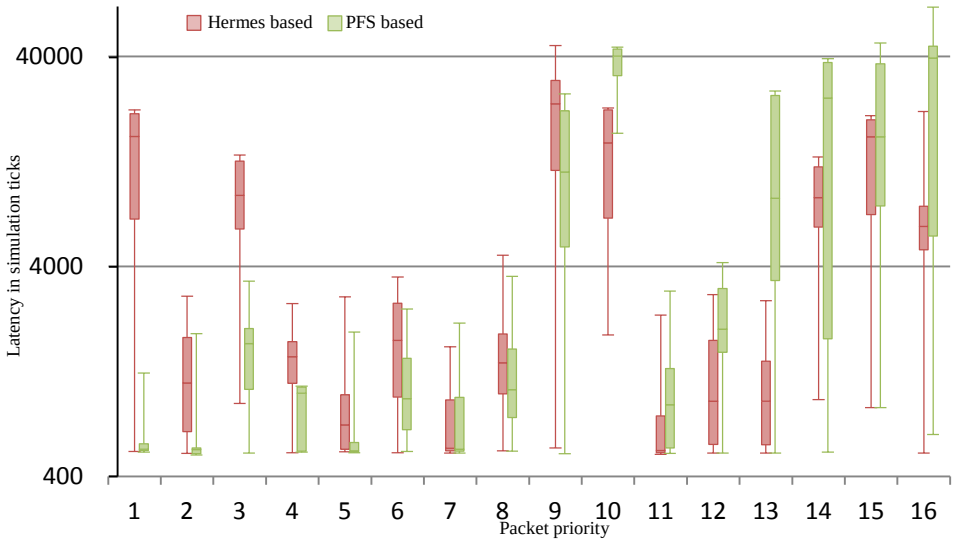


Figure 5.12: Latency performance with random traffic at V=0.6

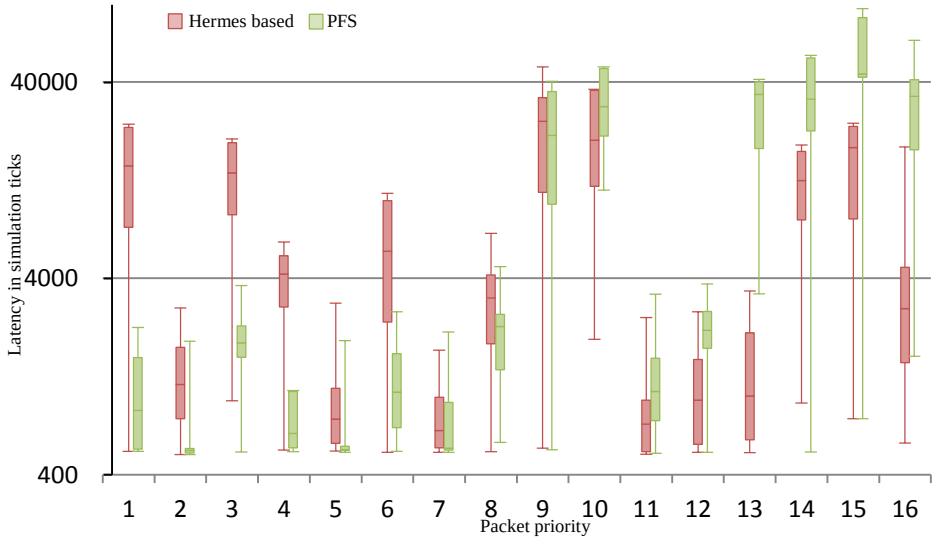


Figure 5.13: Latency performance with random traffic at V=0.8

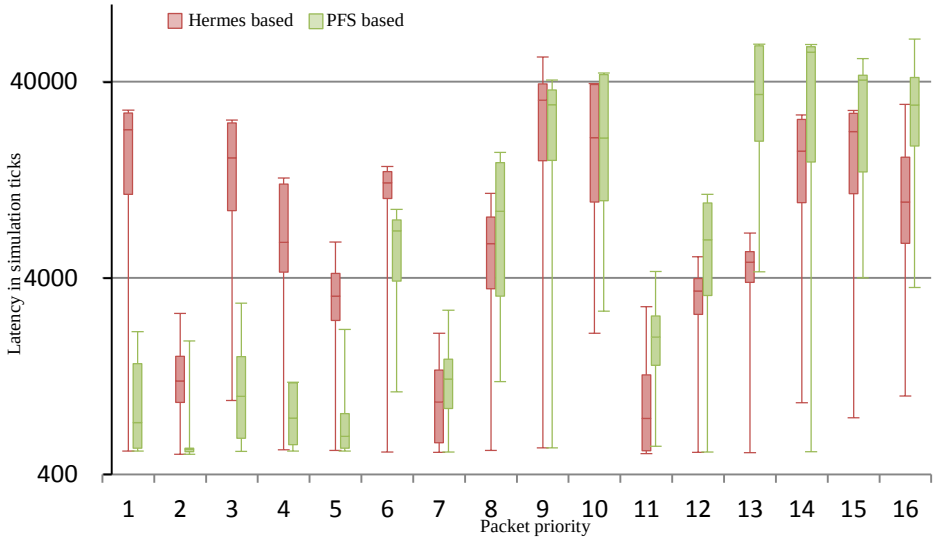


Figure 5.14: Latency performance with random traffic at V=1.0

As evident from the figures, PFS is seen to reduce the magnitude and variation in latency of high priority packets as with varying load levels due to the increase in headers in the NoC.

With these tests however, the effect of PFS is seen to be more prolific with the increase in load due to the increase in packet numbers as the Hermes based NoC suffer poor latency figures. With the high load of 0.8 and 1.0, the high priority packets (1 to 6) are seen with very high magnitude and variation in latency with the Hermes based NoC but with PFS the variation and magnitude is confined to low levels.

5.4.4. Performance Variation with Packet Size

Figure 5.15 and Figure 5.16 depict the performance of the NoC with packet size scaled proportionally and inversely proportionally to packet priority.

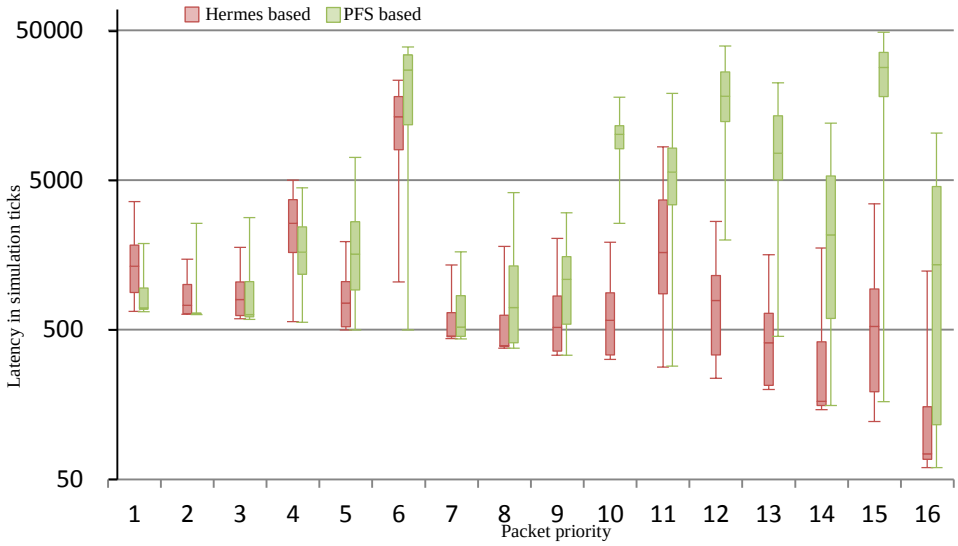


Figure 5.15: Latency performance with packet size scaled proportionately to packet priority

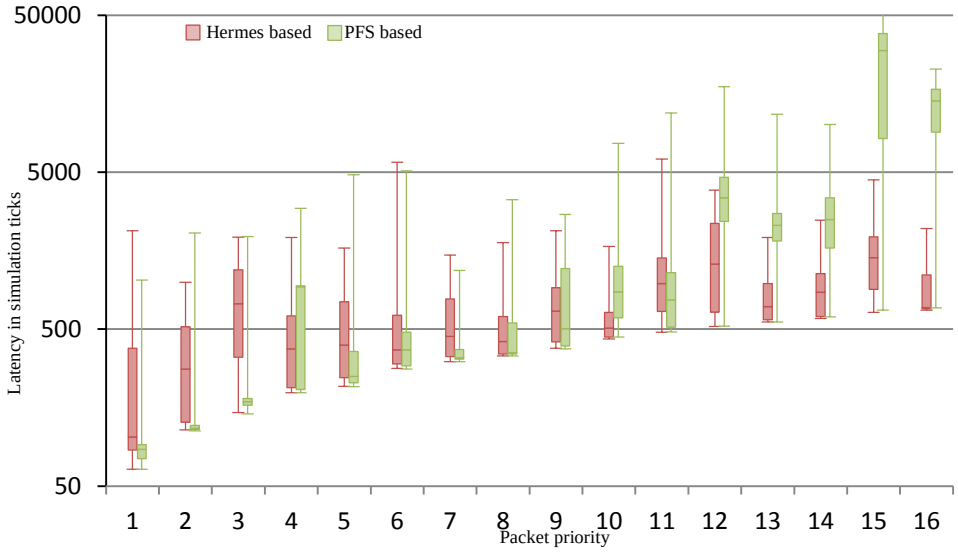


Figure 5.16: Latency performance with packet size scaled inversely proportionately to packet priority

From the figures, it is evident that although the advantages of PFS are clear, the effectiveness of PFS is seen to be more when the high priority packets are shorter compared to lower priority packets (Figure 5.16) as seen with PFT in Chapter 4. This is due to the fact that when the low priority packets are longer than the high priority packets, the short high priority packets would get more of an advantage

splitting them whereas the Hermes based high priority packets would have to wait.

5.5. Hardware Overhead

The hardware evaluation show that the baseline router (2-position input buffers) based on Hermes with priority based arbitration utilised 1209 Look Up Tables (LUTs) and 710 Slice Registers of the chosen FPGA. Compared to that, the PFS enabled NoC (2-position input buffers) however utilised 2382 LUTs and 1050 Slice registers.

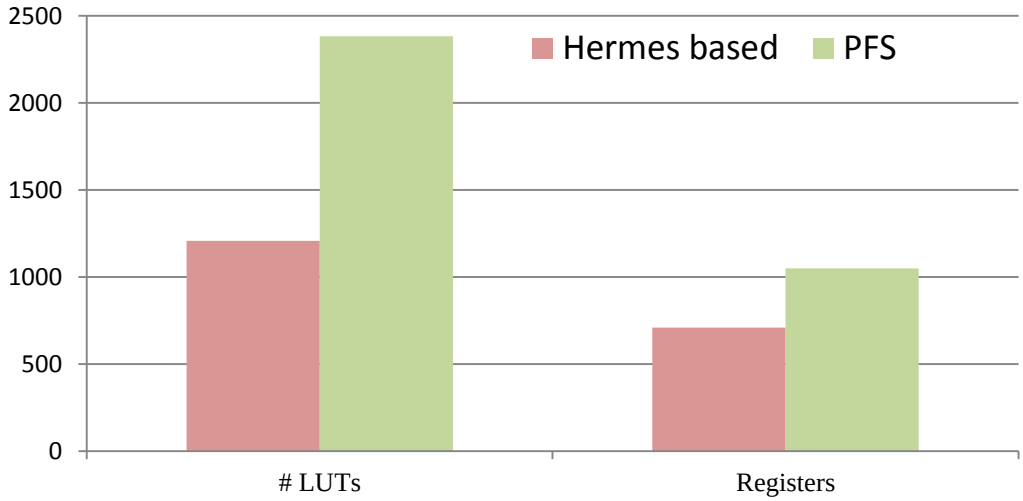


Figure 5.17: Hardware overhead

Detailed hardware overhead details is added in Appendix 3 in sections R2 and R7-F.

5.6. Summary

This chapter initially introduced the SPS techniques aimed at emulating preemption functionality by splitting packets. With the use of VC as in QNoC, packet pre-emption requires buffers and additional hardware which will increase the overall hardware requirements significantly. It also limits the scalability of the NoC as the size of the NoC or the number of packet priorities increase, the effectiveness of the pre-emption will decrease unless more buffers are added.

SPS however avoids the used of the classical pre-emption technique to reduce hardware requirements and to ensure scalability. With SPS, the routers are modified to split a low priority packet if a higher priority packet requires arbitration to the same port. The routers are designed to split the low priority communication by sending a tail flit down the line so that the routers down the line will know of the end of the packet and will close communication automatically after receiving the tail flit. By using such a system, SPS does not require any additional connection lines to communicate with downstream routers, thus making the design simple.

In SPS, custom logic is added to the routers to issue a new header and arbitration request to the split packet so that once the transmission of the high priority packet is completed, the rest of the split lower priority packet can be transmitted. As the rest of the split packet is an intact packet with header, payload and a tail flit, the routers down the line will treat them as normal packets thus eliminating the need for any additional control signals or links further simplifying the design.

The chapter followed with the details of the PFS technique which is a hybrid between SPS and Priority Forwarding technique [25] from Chapter 4. The implementation details of the PFS based model was presented then followed by the test results. PFS was tested with four random traffic scenarios as well as with varying load levels both due to the increase in payload flits in the NoC and due to the increase in header flits in the NoC. In all the tests, PFS was seen to reduce the magnitude and variation in latency depending on packet priority (better predictability for higher priority packets) compared to Hermes based NoC.

Although acceptable, the improvements in latency performance of higher priority packets are brought about by trading the performance of the lower priority packets. To moderate this effect when possible, further research was conducted to add a timeliness parameter along with packet priority to enable routers to provide better QoS to lower priority packets if the high priority packets have residual slack in latency. This is the concept explored in the next chapter.

Chapter 6

Predictability Enhancement Through Dynamic Slack Awareness

Typically, predictability enhancement techniques like VCs and LDM employ application supplied priority as the decisive parameter. Thus, the routers favour high priority packets over low priority packets while implementing arbitration, preemption or other predictability enhancement measures despite its timeliness. This means that ordinarily, a high priority packet with residual slack (hence early in time compared to its soft deadline) will be preferred over a low priority packet which has no residual slack (hence is late). This will delay the late packet even more while the high priority packet get transmitted even though it can afford to get delayed. The term residual slack is defined as follows.

> ***Residual slack:*** *The time in simulation ticks a packet can be delayed without missing its soft deadline.*

This chapter presents the technique that will allow the routers to improve the predictability of packets with lower priority when possible by trading the residual slack associated with competing higher priority packets.

In real-time systems in which the application structure and system workload is known ahead of time, static analysis can be used to determine suitable packet pri-

orities and mappings. However, in open applications the workload which the platform must handle can be unknown at design time. This can be because tasks or data flows may arrive dynamically requesting immediate transmission, but nevertheless requiring a certain quality of service (QoS). Alternatively, in a heterogeneous architecture, known applications may have to coexist with dynamically admitted traffic. These situations require additional flexibility in arbitration decisions beyond static priorities.

This chapter introduces an approach by which the packets can be added with an additional parameter that notifies the routers of the timeliness of packets so that predictability enhancement measures can be employed by evaluating both its timeliness and application supplied priority.

Typically, the notion of timeliness is realised on multicore systems using time stamps like in [122] and [123]. The use of time stamping however requires access to a global time to compare with. With NoC routers, the addition of a timekeeper unit employing large counters or time synchronisation mechanisms would be hardware expensive and impractical. This chapter presents the technique called Dynamic slack Hard-line Aware Router Architecture (DHARA) [28] using which a notion of timeliness can be introduced into NoC packets.

With DHARA, NoC routers will be able to estimate the residual slack of a packet (thus denoting its earliness) at any instant so that the routers can be equipped with logic to provide preference to packets evaluating both its timeliness and its application supplied priority. This will allow the routers to trade time (residual slack) from early packets to improve the QoS of lower priority packets.

As a practical application for the system, the chapter also details the design and performance implementation results of a PFS enabled prototype which was equipped to utilise the slack information in arbitration decisions.

With DHARA enabled in the PFS prototype, arbitration decisions are made on the basis of a dynamically computed priority value. Packet headers are augmented with an additional slack value and this slack value is decremented by intermediate arbiters while the packet is blocked and forced to wait. During arbitration deci-

sions, an instantaneous priority is computed from this slack value and the application supplied priority value. This dynamic priority adjustment allows lower priority packets (which have been waiting) to be serviced trading residual slack available on higher priority packets.

6.1. Motivational Example

Consider two PFS enabled routers (previously seen in Chapter 5) in a scenario shown in Figure 6.1.

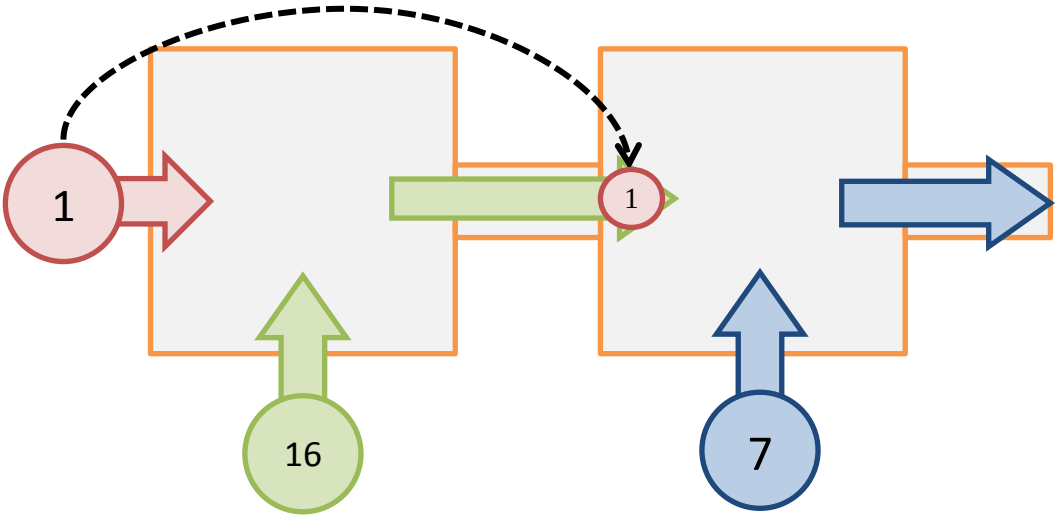


Figure 6.1: Motivation example

In Figure 6.1, packet 1 is blocked by packet 16 which is itself blocked by packet 7. As the routers are equipped with PFS, packet 1 will be able to split packet 16 and packet 7. As a result, once the initial part of packet 7 is transmitted, the initial part of 16 will be transmitted followed by packet 1. It is only after the transmission of packet 1 that packet 7 and packet 16 will be transmitted, which is desirable under a normal situation.

Assume the situation where packet 1 is early in time compared to its soft deadline. In such a situation, forcing packet 7 and 16 to wait is unnecessary and inefficient. After packet 1 is transmitted, packet 7 will get transmitted forcing packet 16 to wait further. Assume the situation where packet 7 is too late to be useful. In such a situation, transmitting packet 7 will be a waste of resource as it would result in

unnecessary traffic further along the route of packet 7, and would increase the latency of packet 16 for no reason.

In this example, the introduction of the notion of time and equipping the routers to perform PFS evaluating both priority and timeliness will resolve such issues. This will allow the router to trade the residual slack (expendable time on higher priority packets) for latency enhancement in lower priority packets and thus provide an overall improvement in QoS.

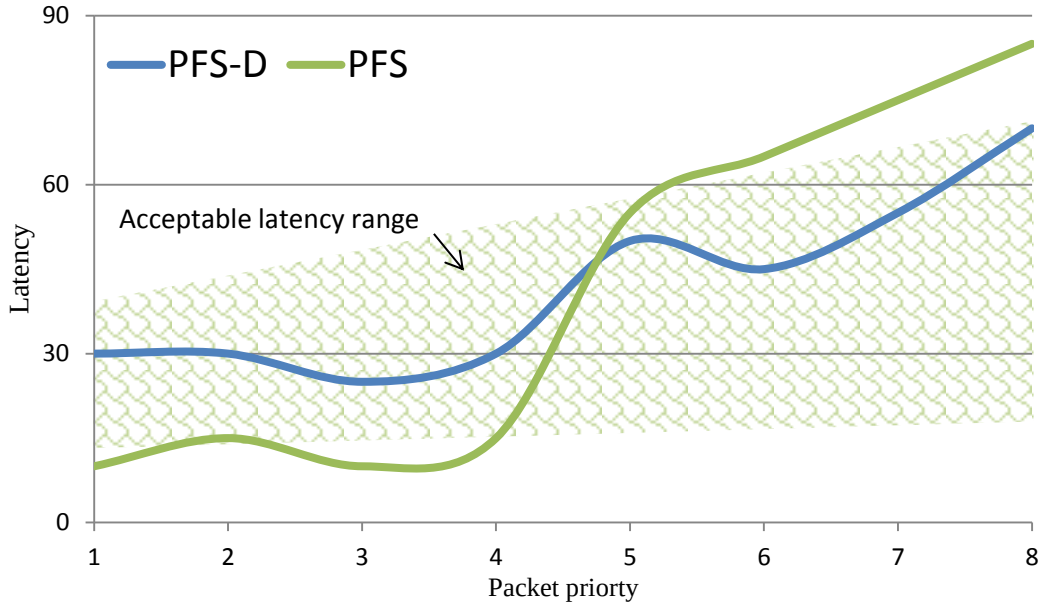


Figure 6.2: DHARA functionality

To understand the expected functionality in detail, consider the performance of a PFS based NoC and a PFS based NoC with DHARA enabled (denoted as PFS-D) in Figure 6.2 (hypothetical example). It can be seen that the PFS based NoC has very low latency for the high priority packets (packets 1 to 4) at the cost of the lower priority packets (packets 5 to 8). As a result the low priority packets suffer high latency which are outside the acceptable latency range with respect to its soft deadline. The idea with DHARA is to moderate this negative effect by trading the slack the high priority packets have (inside the acceptable latency range) to improve the performance of the lower priority packets. In the plot, it can be seen that with PFS-D, the latency of the high priority packets are increased (still inside the

acceptable range) thus enabling the routers to improve the latency performance of the lower priority packets.

6.2. Residual Slack as the Notion of Timeliness

DHARA enables the packet generator/IP or the Network Interface to provide an additional parameter to packets (apart from priority and destination information) that will notify routers of the residual slack the packet has. Packet headers are augmented with an additional slack value, which represents the latency the packet can endure to its destination without adverse effects. This slack value is decremented by intermediate arbiters while the packet is blocked and forced to wait. During arbitration decisions, an instantaneous priority value is computed from this slack value and the application-supplied priority value. This dynamic priority adjustment allows lower priority packets which have been waiting for longer to be serviced, while trading off some residual slack available on early high priority packets.

Every time a packet header is injected into an input port, the slack is stored into a register. If the packet gets arbitration immediately, slack along with the rest of the parameters will be sent to the next router. On the other hand, if the packet gets delayed, the value inside the slack register will get decremented every time a slack-interrupt is encountered. To generate slack-interrupt, the router is added with an incrementing counter that will produce a slack-interrupt every time it overflows.

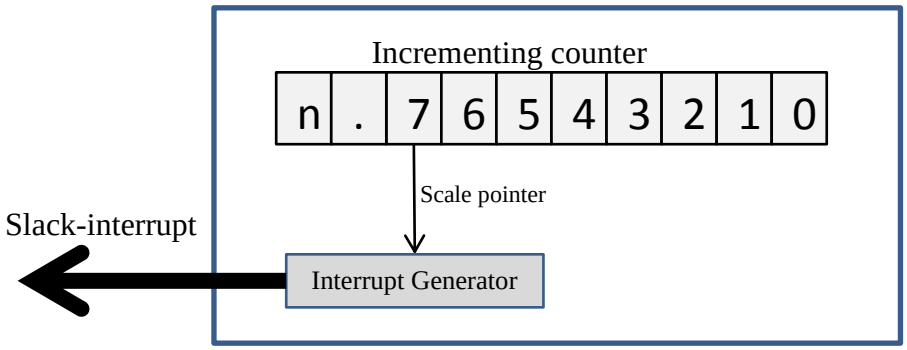


Figure 6.3: Slack-interrupt generator

As there would be packets with different ranges of residual slack, there is also provision to vary the granularity of the wait time upon which the slack-interrupt generation occurs. As shown in Figure 6.3, slack-interrupt generator has an adjustable scale pointer using which the granularity of timeliness can be varied. For example, if the scale pointer is set at zero, the system will provide an interrupt every two clock cycles and so, the slack value will be decremented every two clock cycles the packet is forced to wait. The granularity will be equal to $2^{\text{`scale pointer value`}+1}$ and hence if the pointer is set to 7 as in the figure, the value inside the slack-left register will be decremented every $2^8 = 512$ clock cycles the packet is forced to wait.

The header is also provided with an expendable flag to denote the data which is deemed useless after its slack is exhausted. As a result, if a router encounters a packet with slack set to zero and expendable flag set, the input port will remove the flits (as it has exhausted its slack and hence deemed useless) rather than transmit them and elevate congestion. As this system does not require access to a global time, the hardware requirement is relatively low thus enhancing its practicality.

In larger NoCs, the delay that has to be encountered by packets which has to travel a longer route will be significantly greater than a packet that has to travel a shorter route. As a result, it is sensible to add slack value to packets taking into account the route length of the packet as well. For instance, a packet that has to travel a longer route should be added with lower slack value than a packet taking a short route thus enabling it to reach the destination without excessive delays.

Future work on the topic will involve testing the system with packets of different route length to determine the effects. Future work will also involve integrating a route length component into the instantaneous priority equation so that the routers will be able to account for the position of the packet in the NoC with respect to the destination (instantaneously) while estimating its instantaneous priority.

6.3. Application with PFS Based NoC

To evaluate the performance of the system, a PFS enabled model (R7-F NoC used in the previous chapter) was modified to encompass DHARA (URL to the source code added in Appendix 2).

6.3.1. DHARA Based Slack Awareness

The model used the R7-F NoC as the starting point with additional logic for DHARA implementation. In this case, all computational units including the arbiter, Priority Forwarding logic and packet splitting logic were modified to make decisions based on instantaneous priority rather than the priority information in the packet header (application-supplied priority).

The instantaneous priority is estimated using equation (6.1) which employs an addition and a right shift (>>) operation thus enabling efficient realisation in hardware.

$$P_I = P_P + (S \gg D) \quad \textbf{(6.1)}$$

(P_I – Instantaneous priority, P_P – Packet priority, S – Slack value, D- Divider index)

As seen in the equation, the instantaneous priority is estimated by summing the packet priority and the slack value shifted to the right D number of times. Practically, D can be set to 0, 1 or 2 hence realising S, S/2 and S/4 respectively. Thus by varying the value of D, the weightage of the slack component on the instantaneous priority can be varied.

6.3.2. Implementation Details

DHARA

As a starting point, the PFS enabled (R7-F) router used in Chapter 5 was modified to encompass DHARA. As the slack value was set at seven bits, the highest value possible i.e. 127 is treated as packets with the notion of lateness disabled (where PFS will never be enabled). With slack values less than 127, the routers will decrement the slack value as determined by the scale pointer, described in Section 6.2.

Consider the situation where a header is inside the input buffer of an input port behind some flits of some other packet which is blocked. As the slack register in the input port will be updated with only when the header is in the head of the buffer (typically) thus initiating an arbitration request, such a situation will allow headers to wait for arbitration unaccounted for.

For example, consider the situation in Figure 6.4 where an input port and its input buffer is depicted. In the figure, it can be seen that as the header of a packet is behind the flits from another blocked packet in the FIFO, the system will not be able to update the slack value even though the packet is waiting for arbitration.

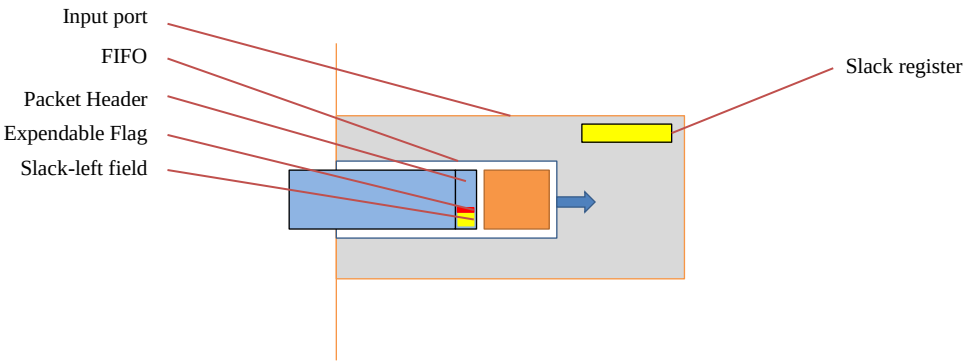


Figure 6.4: HOL blocking of slack-left value

To resolve such issues, the buffers could be modified so that every time a flit is injected into the buffer, the newly added logic will verify whether it is a header and if it is; the slack-left register will be updated. As this will happen before the

packet gets to the front of the queue the routers will be able to decrement the slack value irrespective of the position of the header in the FIFO and hence provide an accurate awareness of residual slack.

The work assumes that packet size will be longer than the buffer size so that only one header would be in a buffer at any point of time. In case the packet size is less than the buffer size, there is possibility of multiple headers getting injected into a buffer and under such a situation, the slack value of the header injected last would overwrite the value inside the slack-register.

Packet generators

To enable slack awareness in packets, the packet generators were modified so that the packet headers produced will include a seven bit (configurable) slack value and a single bit expendable flag. The slack value and expendable flag value are hard coded during design time by the configuration generator (previously explained in Section 3.3).

Type J packet generators

To enable performance testing with complex realistic traffic, an advanced version of the packet generator was developed which would support four sets of communication with an internal preemption mechanism. As a result, each generator can be configured with four in-depended packet flow information.

Internally, each Type J packet generator acts as four individual packet generators and the internal preemption mechanism (SPS based) allows transmission of a higher priority communication even when there is an active lower priority communication.

6.4. Experimental Work

To evaluate the performance benefits, NoC designs were tested for their magnitude and variation in latency using the metrics presented in Section 3.1. Magnitude of latency is evaluated using a variety of plots like box-plots, average latency

plots, maximum latency plots and plots depicting the cumulative count of late packets. Variability of latency is evaluated with plots like box-plots, interquartile range plots as well as the variability metric S-index.

6.4.1. Performance with Random Traffic

Under a random traffic scenario (described in Appendix 1f), the latency box plot and average latency plot of a Hermes based NoC compared to a PFS based NoC and a PFS based NoC with DHARA (PFS-D) are presented in Figure 6.5 and Figure 6.6. For the experiments the Scale pointer was set at 7 and slack value set at 20 for all packets.

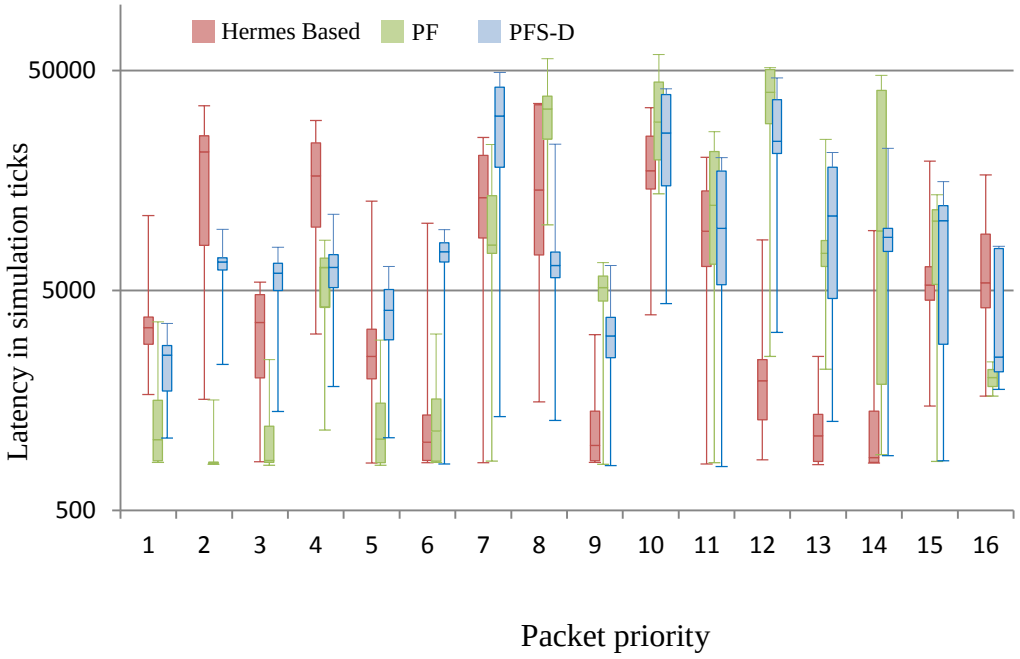


Figure 6.5: Latency comparison with random traffic 1

In Figure 6.5, it can be seen that the Hermes Based NoC suffers high magnitude and variation in latency despite its priority values (like packets with priority 2 and 4) due to HOL blocking and tailbacking.

With the PFS based router, the PFS logic counters HOL blocking and tailbacking thus improving the latency of high priority packets. As this performance improvement is achieved at the cost of the performance of low priority communica-

tion, low priority packets can have increased magnitude and variation in latency (like packet 12 and 14).

With DHARA enabled in PFS routers, this ill effect is moderated by trading the residual slack in higher priority packets and the effect is quite evident in the figure. As a result the PFS-D based system is seen to moderate the extreme cases of latency (like with packet 14).

Even though the best performance with DHARA is achieved by custom allocation of slack value depending on requirements, the tests in the research were conducted with slack values assigned equally to all packets.

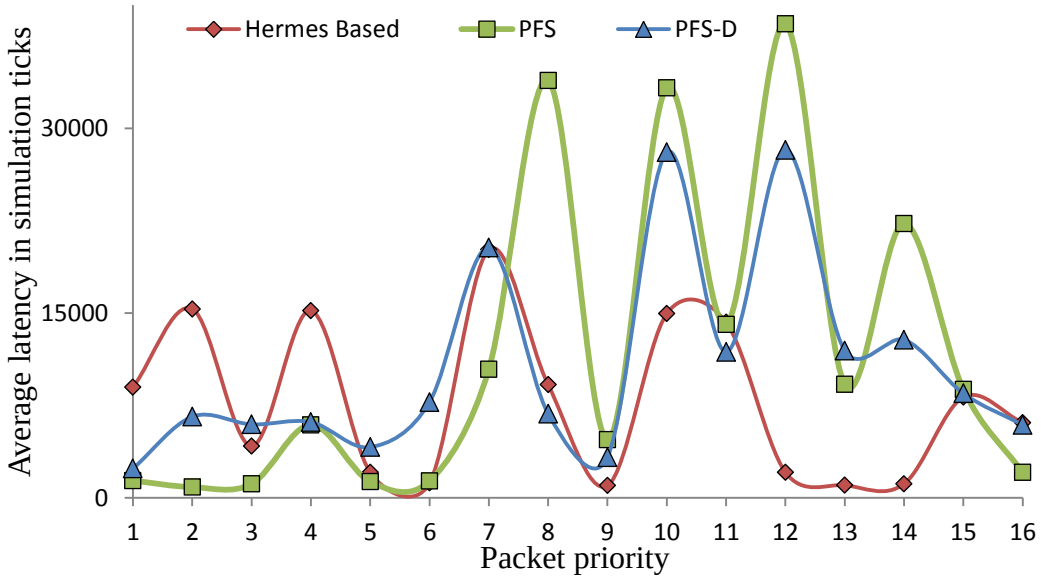


Figure 6.6: Average latency plot for random traffic 1

This effect of DHARA is clearer in Figure 6.6 where the average latency is plotted. It can be seen that there are irregular peaks in average latency with Hermes based NoC regardless of the priority value (like packet 2 and 4). With PFS however these issues are resolved but as a result there are peaks in average latency of lower priority packets (like packets 8, 10 and 12).

The plot corresponding to PFS-D is seen to be more refined than both of those as the high peaks in latencies are seen to be moderated by trading residual slack.

Assuming sum of the allocated slack and the basic latency to be the optimal arrival time, the number of late packets was also evaluated with the traffic scenario and the cumulative count of number of late packets can be seen in Figure 6.7. With the Hermes based NoC, it can be seen that there are late packets regardless of the priority value (like packets 3, 8 and 9). With PFS based router, late packets are not encountered until packet 11 thus showing the effectiveness of the technique.

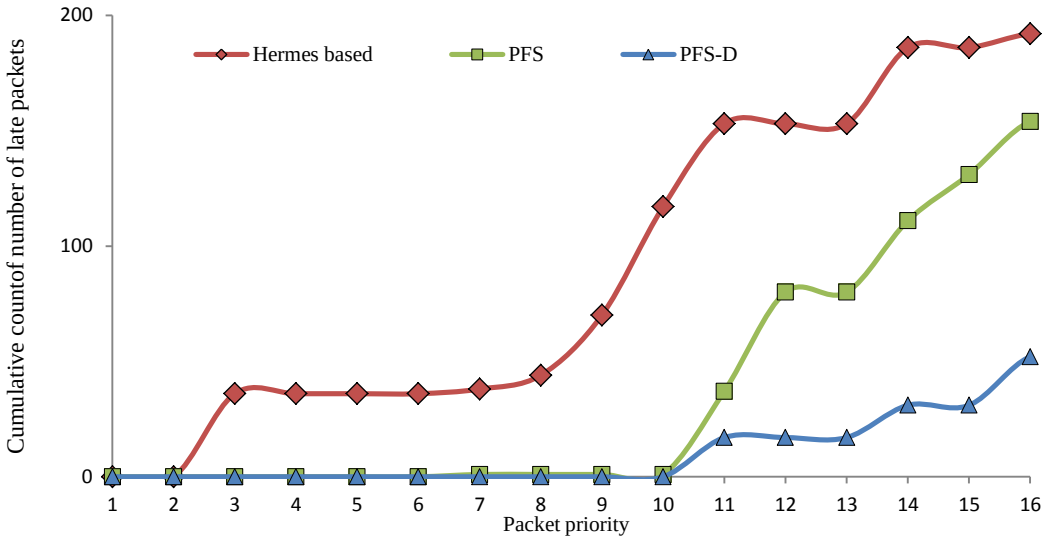
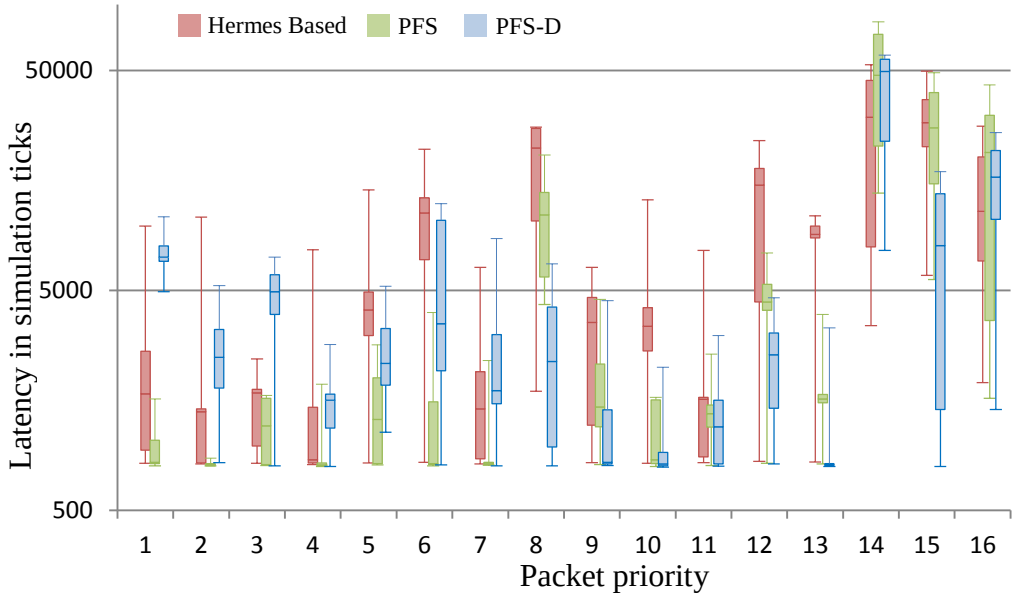


Figure 6.7: Cumulative count of late packets with random traffic 1

Even though the effect is similar with PFS-D, it can be noted that the PFS-D based approach produces a lower number of late packets compared to PFS.

Similarly, the latency plot of another random traffic scenario (Appendix 1g) is depicted in Figure 6.8.



Similar to what was seen with random traffic 1, the effect of PFS and PFS-D is quite evident from the plot. The average latency plot for the scenario is added as Figure 6.9.

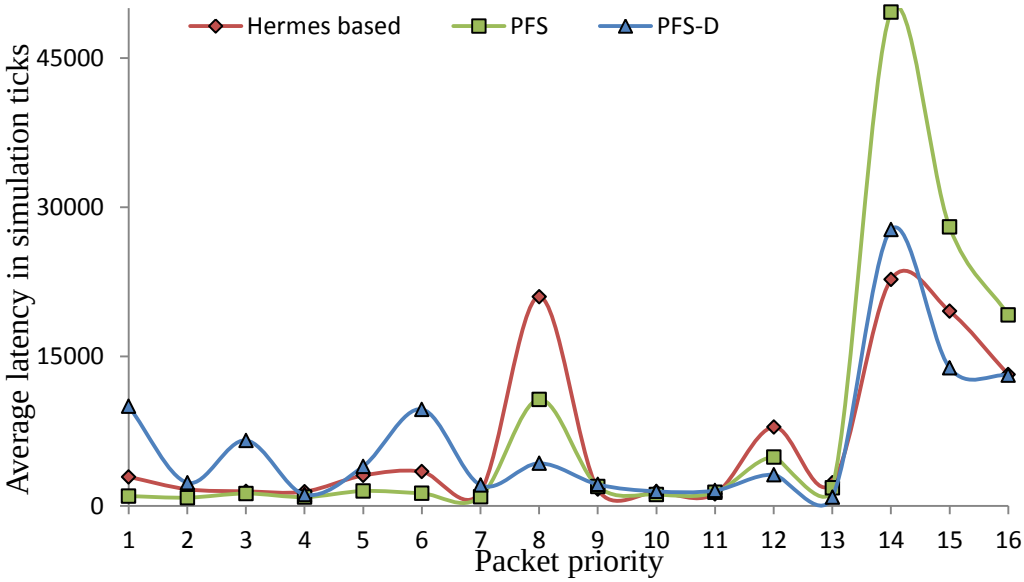


Figure 6.9: Average latency plot for random traffic 2

As seen with random traffic 1, it can be seen that there are peaks in average latency with Hermes based NoC (like packet 8) along with peaks with PFS based NoC (like packet 14). These are seen to be moderated with PFS-D by trading residual slack.

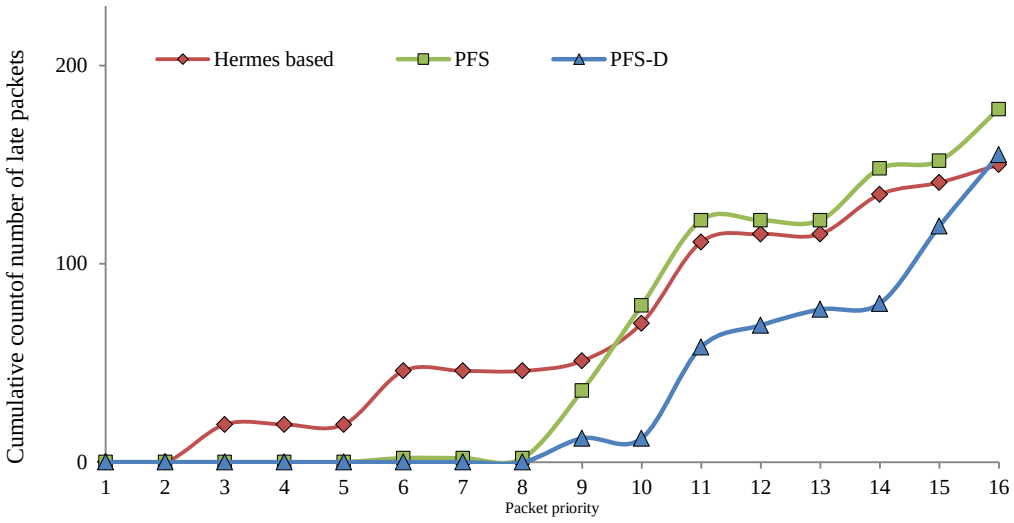


Figure 6.10: Cumulative count of late packets with random traffic 2

In Figure 6.10, the cumulative count of the number of late packets with random traffic 2 can be seen. It can be seen that with the Hermes based NoC, there are late packets of high priority values (like 3 and 6) whereas with PFS and PFS-D, late packets are not encountered until packet 9. With this traffic scenario, the total number of late packets with PFS and PFS-D are seen to be more than the Hermes based NoC. As the late packets in both accounts are from the lower priority spectrum of packets, this phenomenon is justifiable.

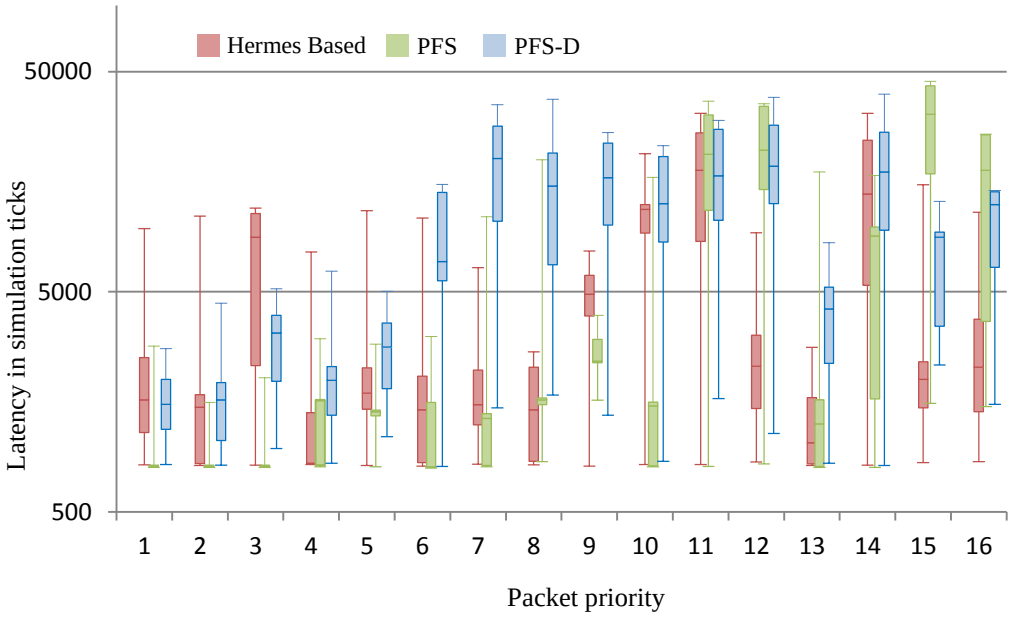


Figure 6.11: Latency comparison with random traffic 3

Similar effect can be seen in Figure 6.11 where the latency of another random traffic (Appendix 1h) is presented as boxplots. Similarly, with the average latency plot of random traffic 3 (Figure 6.12), PFS-D is seen to reduce peaks in latency which were encountered with the Hermes based NoC and the PFS based NoC.

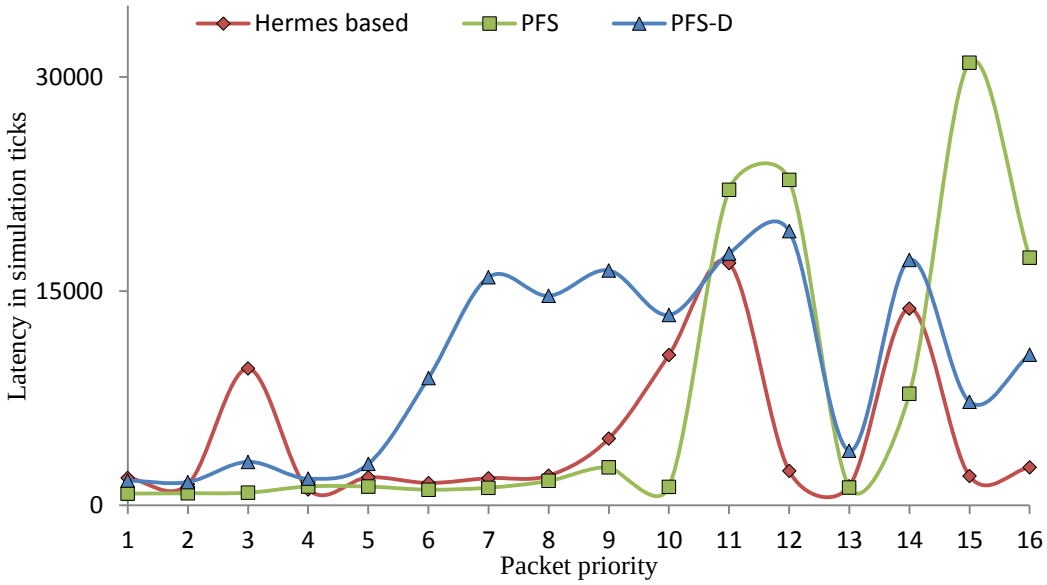


Figure 6.12: Average latency plot for random traffic 3

The cumulative count of the number of late packets is shown in Figure 6.13 and similar to the previous experiments the PFS and PFS-D produce a lower number of late high priority packets compared to the Hermes based NoC. PFS-D however is seen to have a slight increase in late packet numbers with packets 9 and 10, although the total number of late packets is lower than both the other cases.

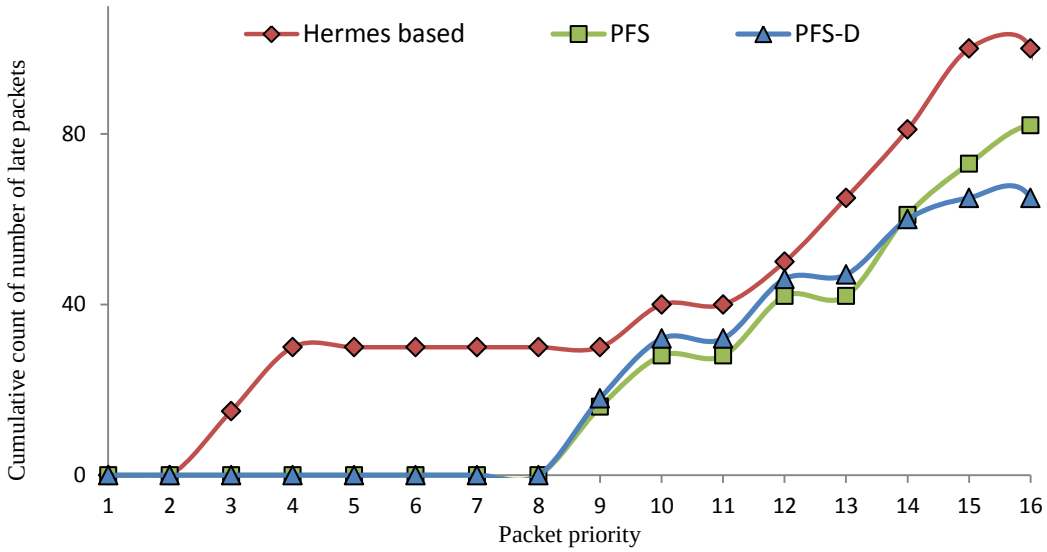


Figure 6.13: Cumulative count of late packets with random traffic 3

6.4.2. Performance with Varying Load

The average latency plots of random traffic 2 at load levels V= 0.6 ,0.8 and 1 are presented in Figure 6.14, Figure 6.15 and Figure 6.16 respectively.

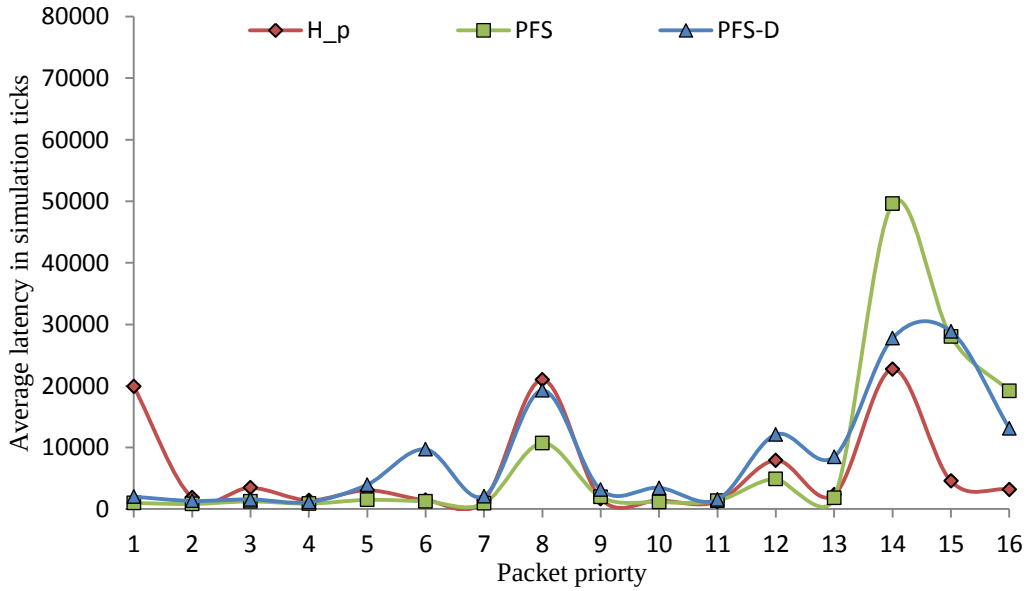


Figure 6.14: Average latency plot for traffic 3 with V=0.6

In Figure 6.14, it can be seen that with the Hermes based NoC, packet 1 is having a high average latency despite possessing the highest possible priority value. With the PFS based router, this is resolved however the lower priority packet; packet 14 has a high average latency.

With PFS-D, both these occurrences are seen to be moderated by trading the residual slack.

With the increase in load to 0.8, the average latency of packet 1 with the Hermes based NoC and packet 14 with PFS NoC is seen to be magnified further (Figure 6.15). However, the PFS-D plot is seen to be having minor variation despite increase in load.

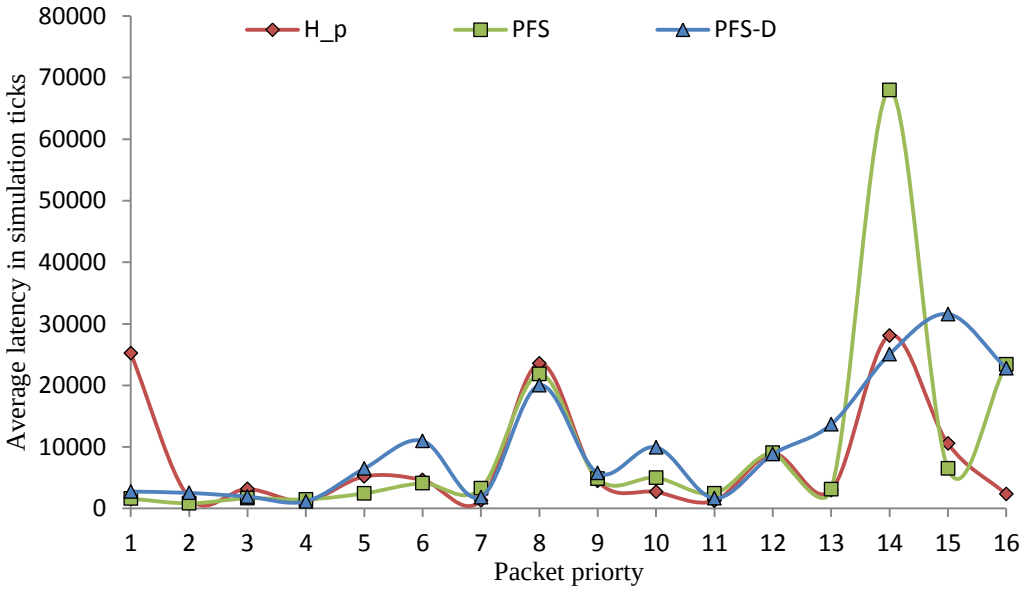


Figure 6.15: Average latency plot for traffic 3 with V=0.8

Similarly, with the increase in load to 1, high variation in latency can be seen with both the Hermes based NoC and the PFS based NoC however the PFS-D based NoC displays lower variation (Figure 6.16) than both cases.

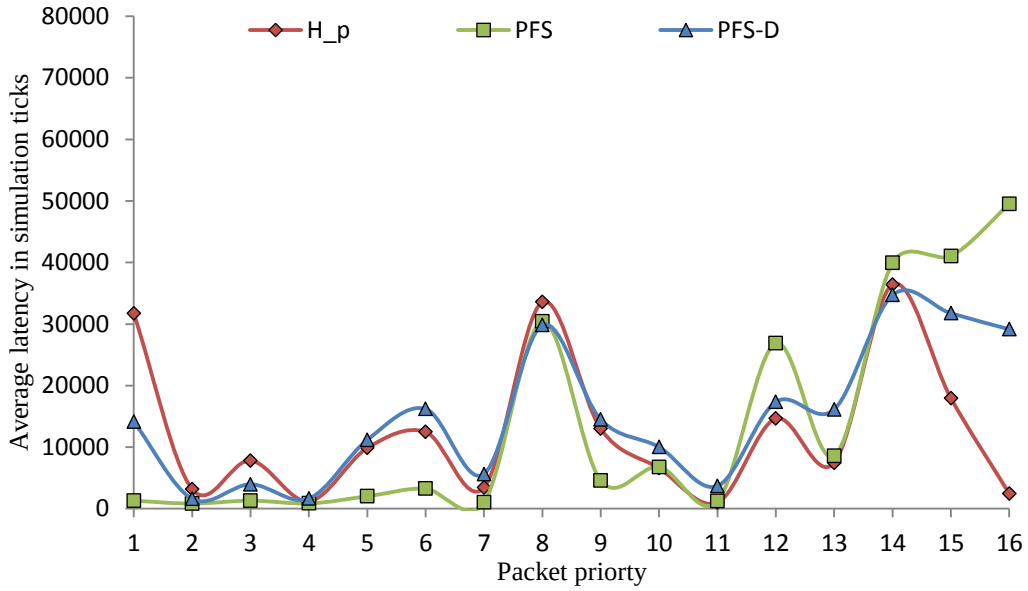


Figure 6.16: Average latency plot for traffic 3 with V=1

The variation of maximum latency in the Hermes based NoC at the three load levels is presented in Figure 6.17. It can be seen that there are peaks in maximum latency regardless of the priority value due to HOL blocking and tailbacking.

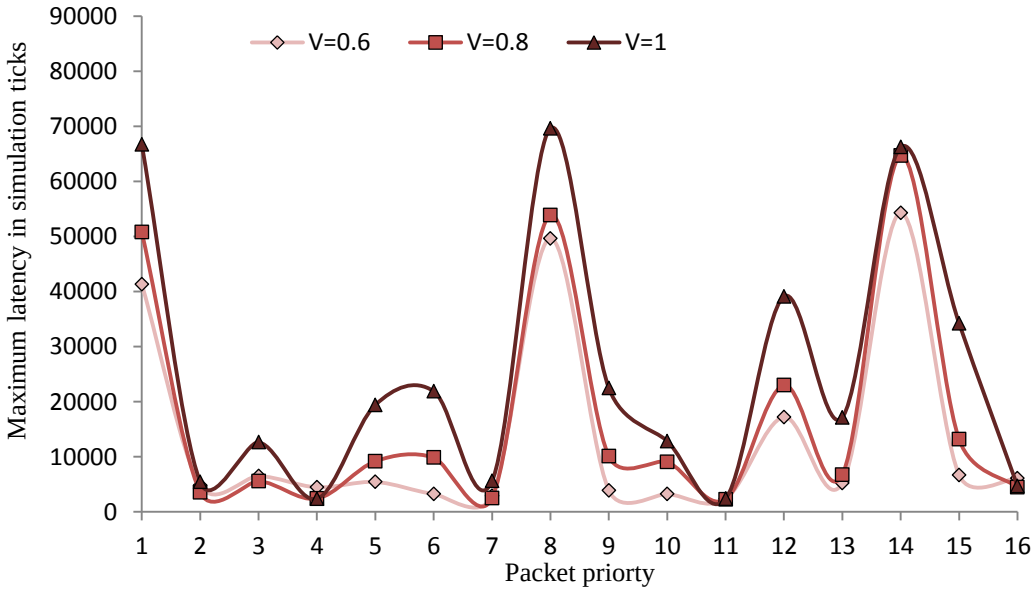


Figure 6.17: Maximum latency variation with Hermes based NoC

The performance of the PFS based NoC is shown in Figure 6.18 and it can be seen that with the use of PFS the high priority packets (1 to 7) suffer very low maximum latency under the three load levels. However this results in high magnitude and variation in low priority packet's maximum latency plots.

For example, with packets like 8, 12 and 15, variations of high magnitudes can be seen with the increase in load.

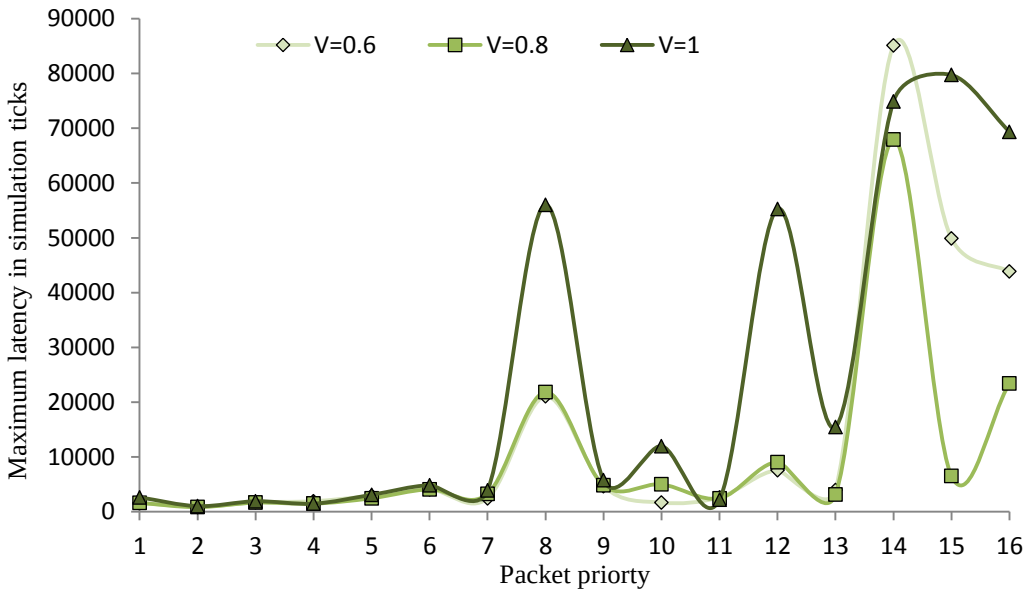


Figure 6.18: Maximum latency variation with PFS based NoC

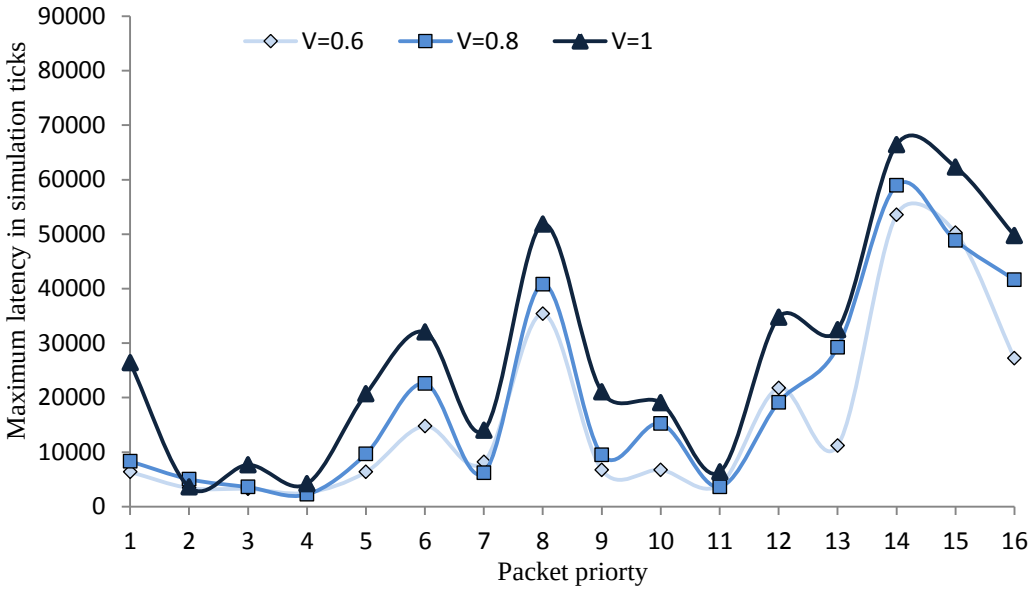


Figure 6.19: Maximum latency variation with PFS-D based NoC

With PFS-D (Figure 6.19), the maximum latency of packets is seen to be lower through the lower priority range and the effect of the increase in the load is seen to follow a pattern and is seen not to produce large variation in maximum latency unlike PFS or Hermes based NoC.

Figure 6.20 shows the average value of slack left on packets upon final reception with the three load levels.

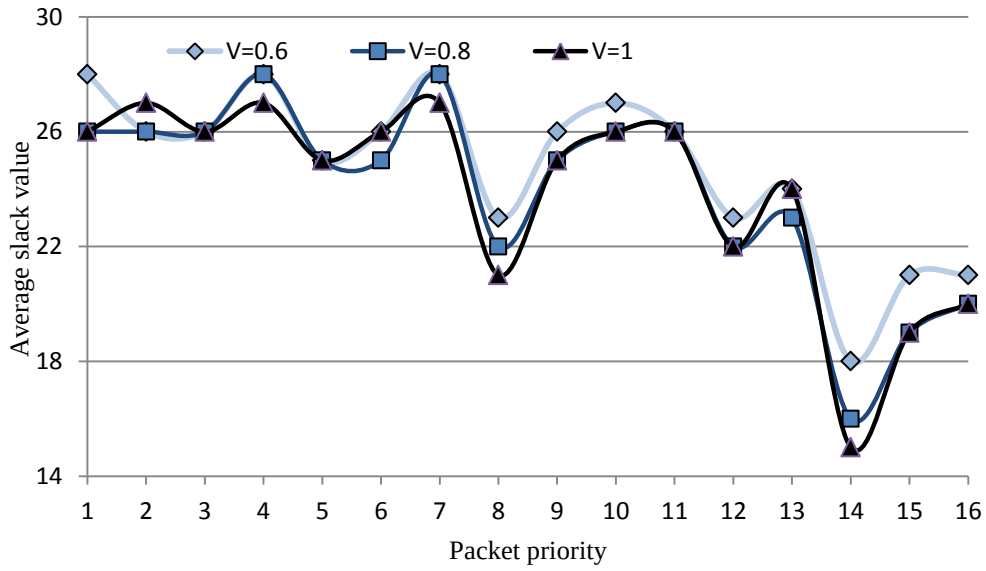


Figure 6.20: Average remaining slack

It can be seen that the low priority packets (8 to 16) suffers more decrease in average remaining slack (from initial value of 30) than the higher priority spectrum of packets and this effect is seen to intensify with the increase in load on the NoC. As per design, this reduction in slack is what enables them to achieve arbitration ahead on higher priority packets occasionally to improve their quality of service. The higher priority packets (1 to 7) seem to show minor reduction in slack value thus confirming that they are not subjected to long waiting periods for arbitration that could negatively affect is QoS.

6.4.3. Performance with Divider Index Variation

As the divider index defines the weightage of slack component in computing the instantaneous priority, its impact upon the NoC using PFS-D was tested by setting divider index at 0, 1 and 2. The results from the tests are presented as a boxplot in Figure 6.21.

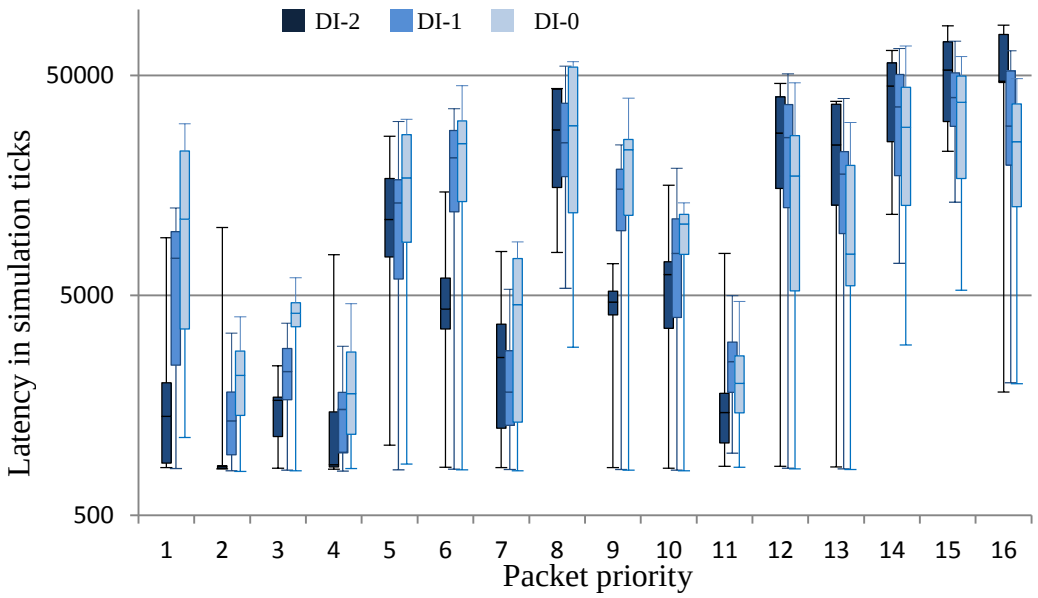


Figure 6.21: Latency variation with divider index

It can be seen that with divider index 2 (DI-2), the weightage of slack component in computing the instantaneous priory is $1/4^{th}$ hence the high priority packet are seen with lowest magnitude and variation in latency at the cost of the lower priority packets. With the change in divider index to 1 and 0 (DI-1 and DI-0) the

weightage of slack component gets altered to ½ and 1 thus scaling the performance of the system.

With the tests with DI-1 and DI-0, the improvement in low priority packet latency is visible (like with packets 13, 14, 15 and 16); which is achieved by trading the performance of higher priority packets (like 1, 2 and 3).

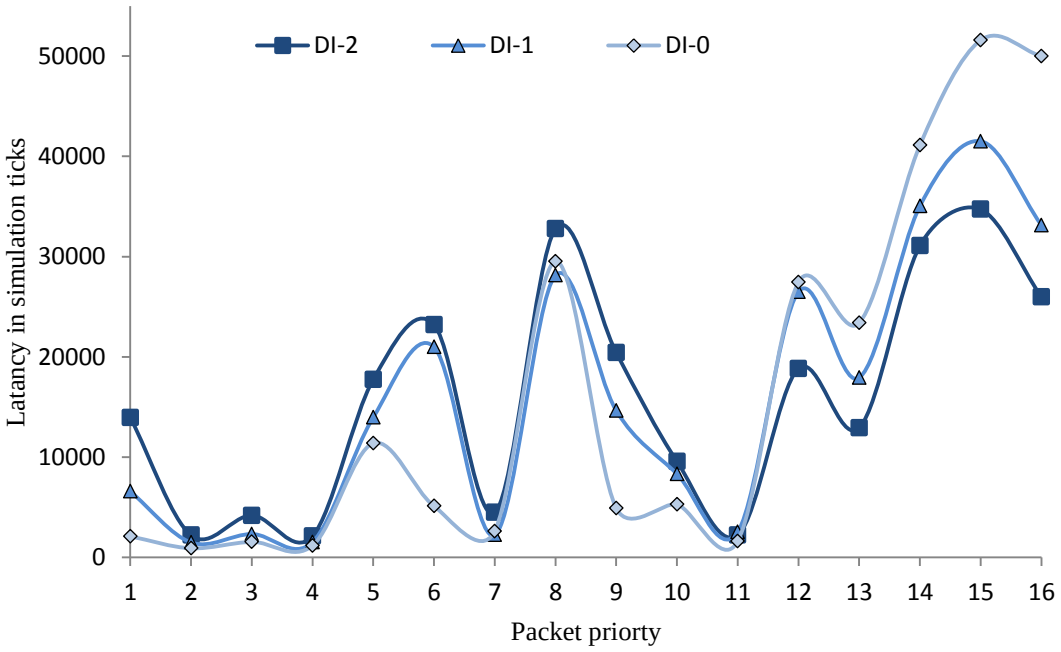


Figure 6.22: Average latency variation with divider index

Similarly, the same effect can be seen in Figure 6.22 where the average latency for the three conditions is plotted.

6.4.4. Performance with Realistic Traffic

To evaluate the performance with realistic traffic, the system was tested with a traffic scenario based on the application used in [16] (configuration added in Appendix 1i) and Figure 6.23 shows the cumulative count plot of the number of late packets under the scenario. For this experiment, a classical round robin based Hermes router based NoC (without packet prioritisation) designated as H was also tested. It can be seen that the Hermes based NoC packets suffer high magnitudes

of late high priority packets. The addition of packet priorities (priority based arbitration) does improve the situation marginally as evident from the plot for the Hermes based NoC with packet priorities depicted as H_P. This scenario is improved with PFS and by using PFS-D, the number of late packets are seen to decrease even further.

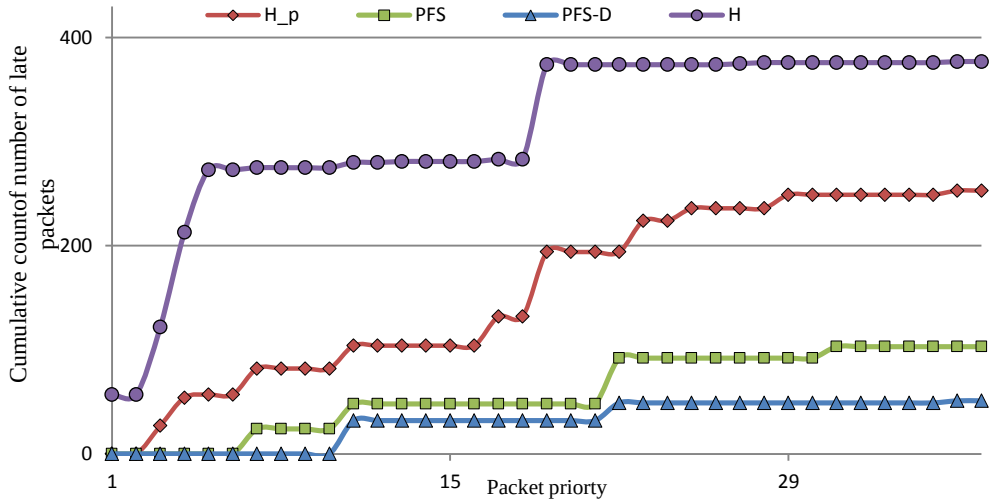


Figure 6.23: Performance with realistic traffic

To evaluate the effect of additional packet flows, ten more packet flows (with lowest priorities) were added.

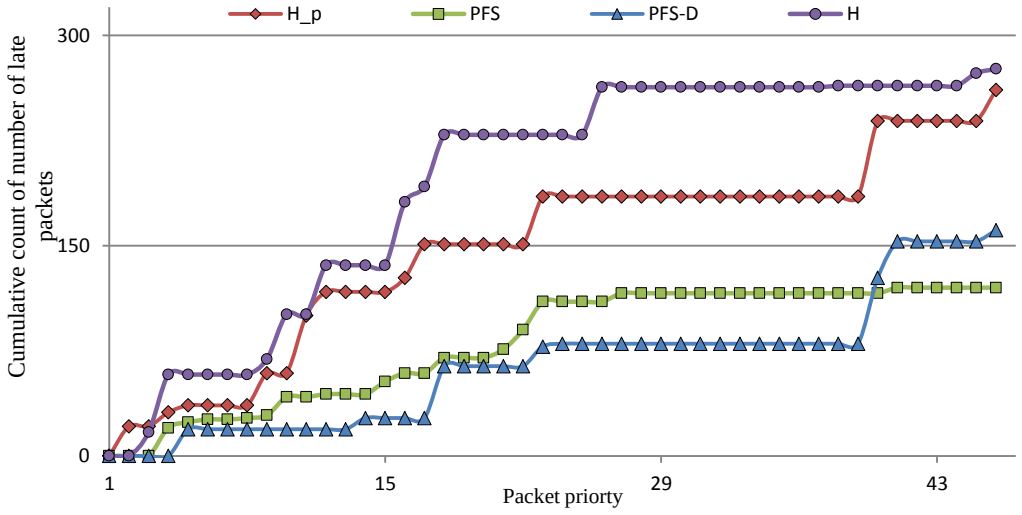


Figure 6.24: Performance with hybrid traffic

The result of the experiment is presented in Figure 6.24 and it proves comparable to the previous experiment however; a few of the lowest priority packets (packets 40 to 42) are seen to have increased number of late packets with PFS-D than PFS.

6.4.5. Scalability of Priority Levels

As PFT, PFS and PFS-D does not rely on VCs or slot tables and instead depend on dynamic alterations to packets and arbitration polices, the NoCs employing the techniques are totally scalable (both in NoC sizes and packet priority numbers). Unlike VCs which need additional hardware with increase in NoC size, the techniques presented in the thesis require a fixed number of components regardless of the size of the NoC.

Though the majority of the performance tests in the thesis were carried out with 4x4 NoCs, to demonstrate the scalability of the system, the latency performance of a 6x6 NoC with a random traffic scenario (Appendix 1j) is depicted in Figure 6.25.

As with the random traffic based tests with the 4x4 NoC, the tests show advantages in magnitude and variation in latency of higher priority packets with PFS compared to the Hermes based NoC. This can be seen resulting in increased magnitude and variation in latency of the low priority packets. This is seen to be moderated with PFS-D by trading the residual slack associated with the higher priority packets.

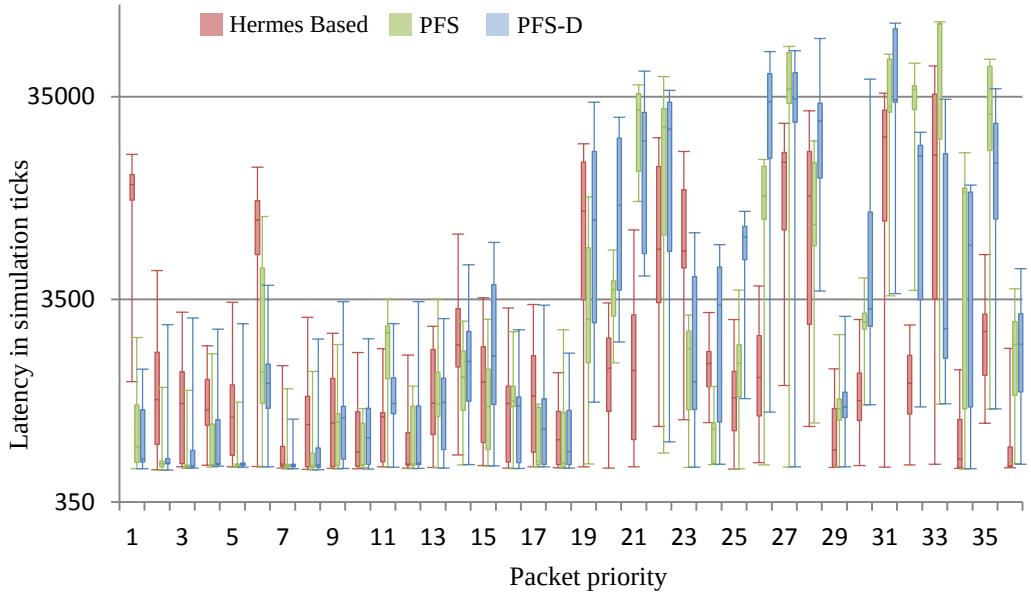


Figure 6.25: Latency performance of a 6x6 NoC with random traffic

The average latency plot from the test is added as Figure 6.26. As with the previous tests, PFS-D is seen to resolve the peaks in average latency of high priority packets with the Hermes based NoC as well as the peaks in low priority packets as seen with PFS based NoC.

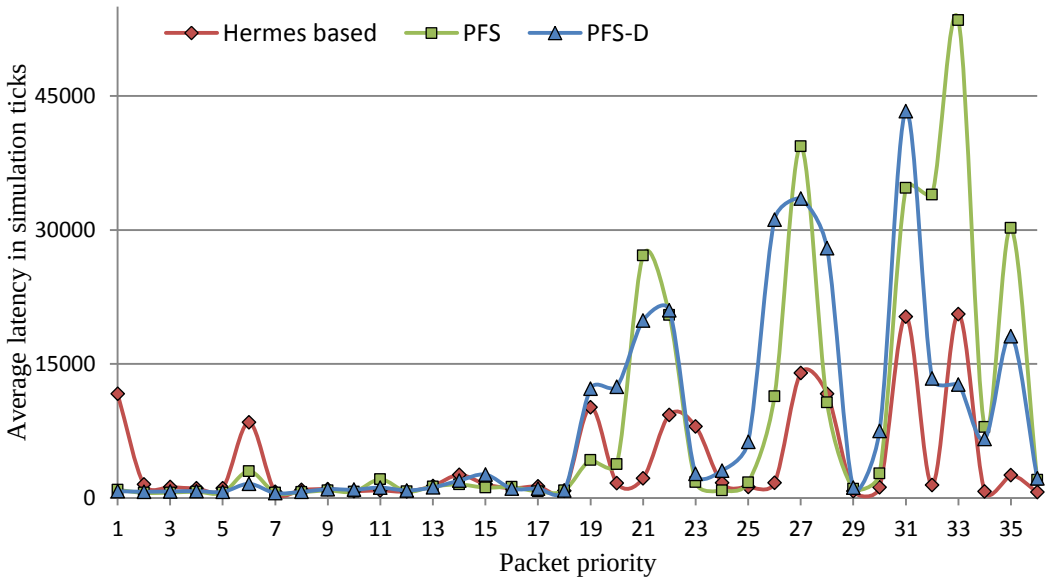


Figure 6.26: Average latency of a 6x6 NoC with random traffic

As seen before, the hardware overhead for VC based NoC is much higher than the techniques presented in the thesis. As a result, NoCs bigger than 4x4 had high magnitudes of overhead that the Bluesim simulation environment was unable to simulate its performance.

Similarly, the average latency plot of a 8x8 NoC is shown in Figure 6.27. It can be seen that the performance of PFS and PFS-D is unaffected with the increase in size of the NoC and increase in packet priorities as in previous tests. This is because the techniques introduced in the thesis use a fixed number of components regardless of the size of the NoC and the number of priority levels. However, the biggest NoC size that could be tested with the tools was 8x8 (average latency plot added as Figure 6.27).

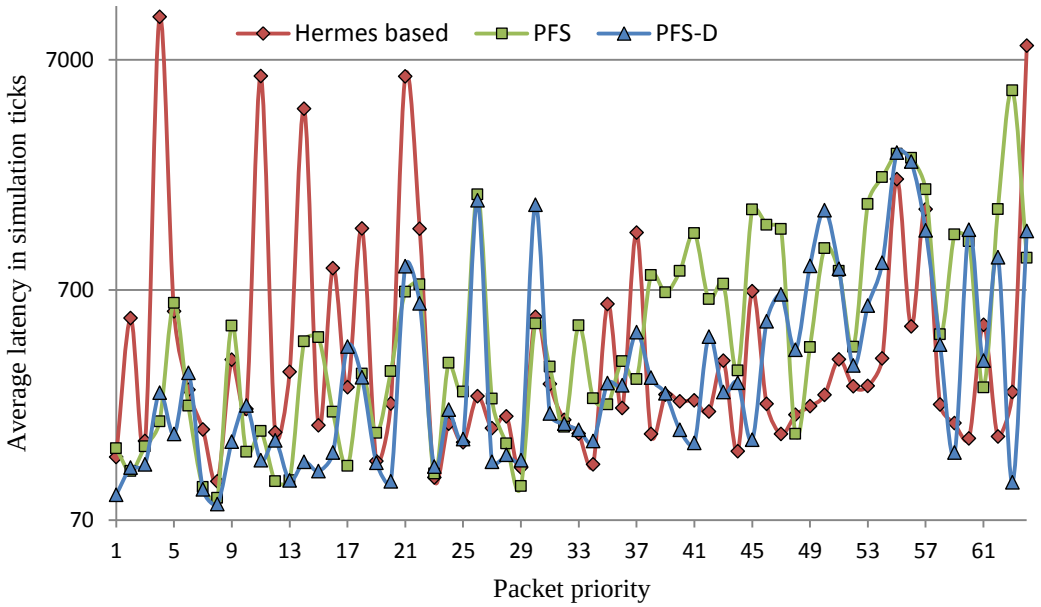


Figure 6.27: Average latency of a 8x8 NoC with random traffic

6.4.6. Comparison with VC Based NoCs

To test the performance of the techniques compared to the VC approach, a VC based NoC prototype was implemented designated the R8 (URL to the source code added in Appendix 2). The R8 NoC use a variation of the R2 NoC (Hermes Based) with added feedback lines as components and it is replicated (to a number of times equal to the number of VCs needed) to enable multiple service levels.

The access to the output ports have a priority based system so that if multiple VC packets need access to the same output port, only the highest priority VC (which is not blocked) will be allowed access.

The latency box plot of a Hermes based NoC, PFS, PFS-D and a VC based NoC with 4 channels under three random traffic scenarios are presented in Figure 6.28, Figure 6.32 and Figure 6.36.

Similarly, the average latency plot for the scenarios are presented in Figure 6.29, Figure 6.33 and Figure 6.37.

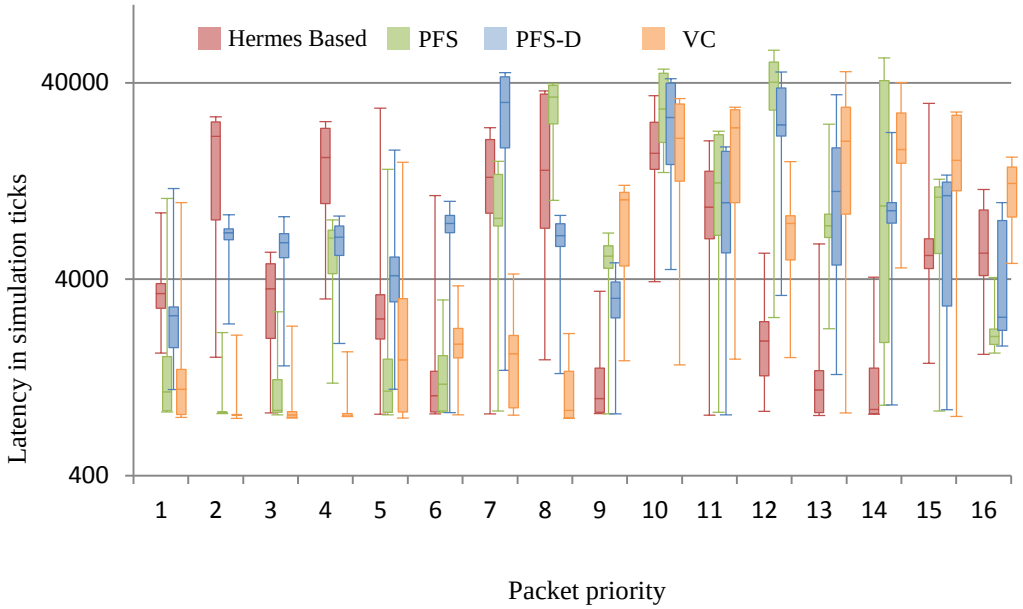


Figure 6.28: Latency box plot of random traffic 4

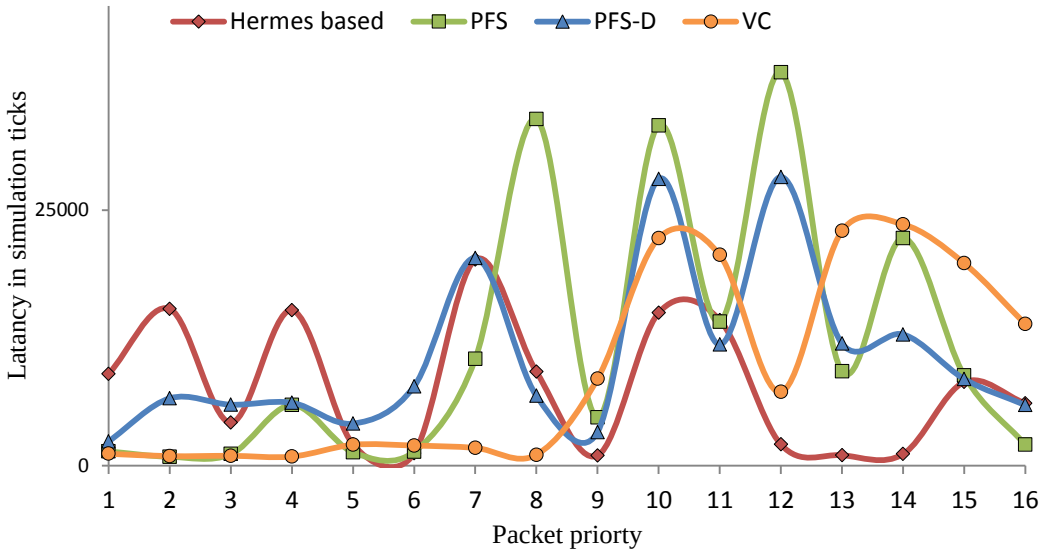


Figure 6.29: Average latency plot of random traffic 4

As seen with previous experiments, PFS is seen to provide lower average latency for high priority packets than the Hermes based NoC (packets 1 to 7) but suffers high peaks in average latency with the lower priority packets (packets 8, 10 and 12). PFS-D is seen to moderate these high peaks by trading the residual slack associated with higher priority packets. The VC based NoC presents the best performance among the four NoCs with very low average latency for high priority packets (packets 1 to 9) and moderate levels of average latency for lower priority packets compared to the other schemes.

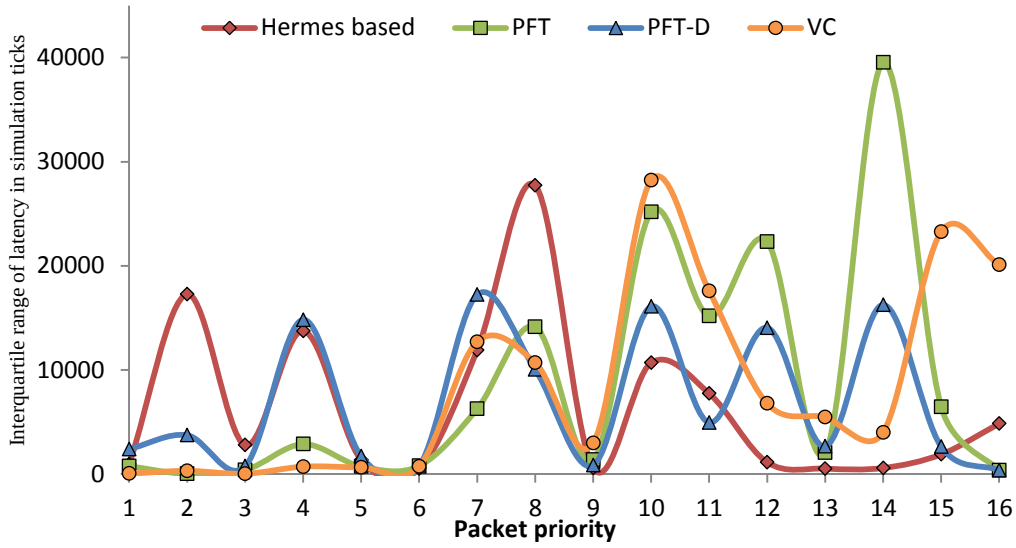


Figure 6.30: Interquartile range of latency of random traffic 4

Figure 6.30 show the plot that depicts the interquartile range (to show latency variability) in packet latency. With the Hermes based NoC, it can be seen that there are peaks in interquartile latency (depicting high latency variation) regardless of the priority range (like packet 2 and 8). Although PFS resolves the latency variation of high priority packets (1 to 9), lower priority packets encounter high variation in latency (packet 10, 12 and 14). The VC based NoC is seen to provide marginally better predictability than PFS especially with the packets 12 and 14. The performance of the NoC over the whole priority range is quantified as S-index and is presented in Figure 6.31. As explained in Section 3.1.1, the S-index is used to quantify the latency variability of the NoC over the whole priority range and a lower value of S-index depict lower variation in latency.

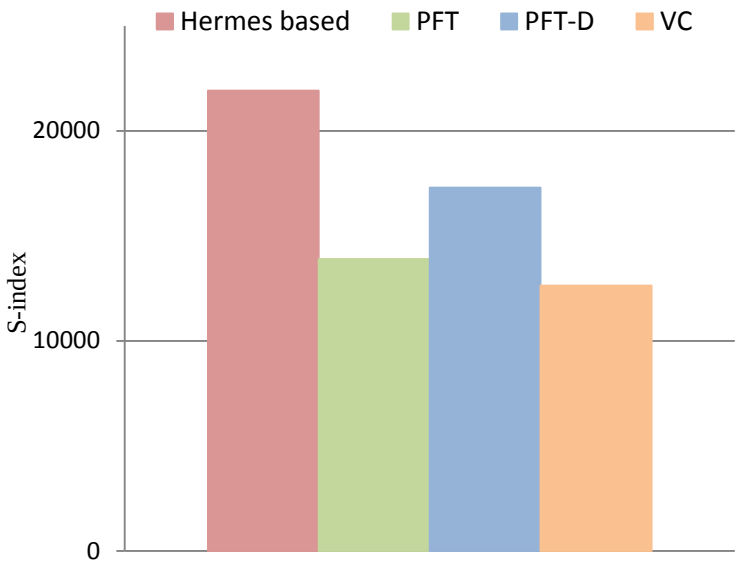


Figure 6.31: S-index plot of random traffic 4

In this particular case, the PFS-D NoC had high slack value assigned equally to all the packets thus resulting in higher S-index than PFS and VC. However the performance of PFS-D can be improved by assigning slack for packets based on necessity rather than equally as in the experiment.

The latency performance of the four NoCs under another random traffic scenario is interpreted as box plot in Figure 6.32. The average latency plot for the system is presented in Figure 6.33.

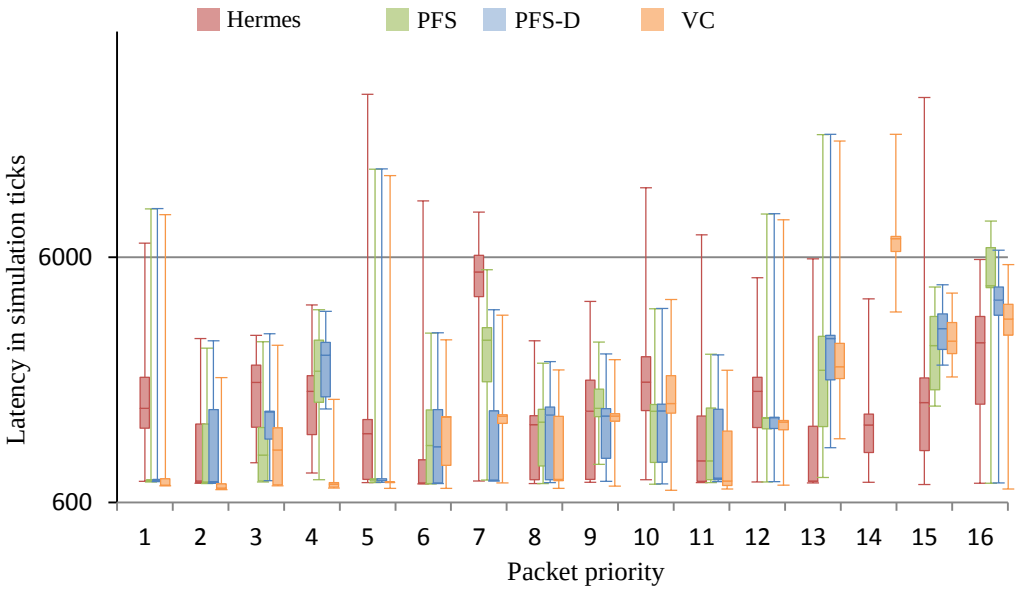


Figure 6.32: Latency box plot of random traffic 5

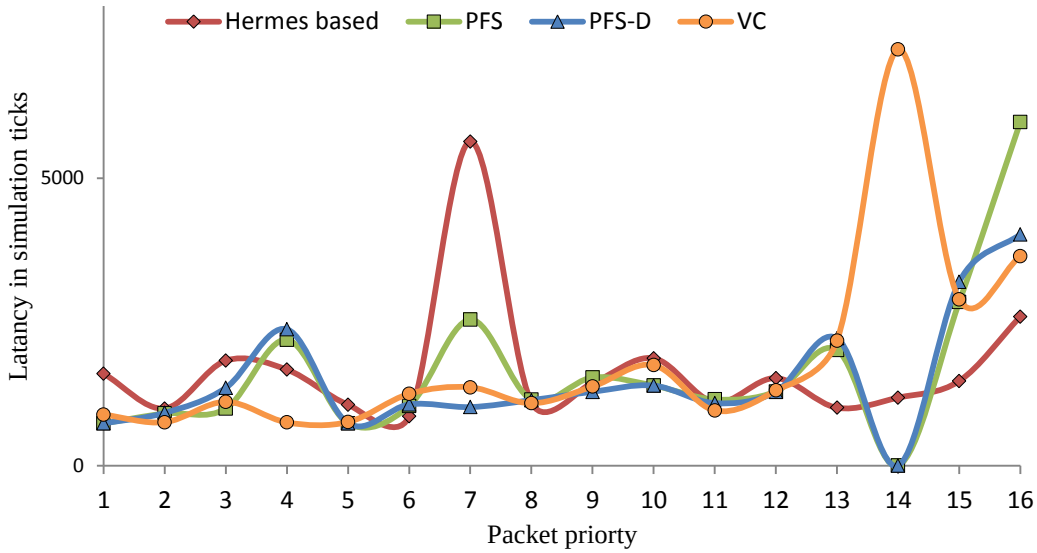


Figure 6.33: Average latency plot of random traffic 5

Similar to the Traffic scenario 4, in traffic scenario 5, PFS-D is seen to provide lower average latency performance throughout the entire priority range compared to the Hermes based NoC (Figure 6.33). The performance of PFS, PFS-D and VC is quite similar in this case however there are differences in latency variation as evident from Figure 6.34 where the interquartile range of latency is plotted.

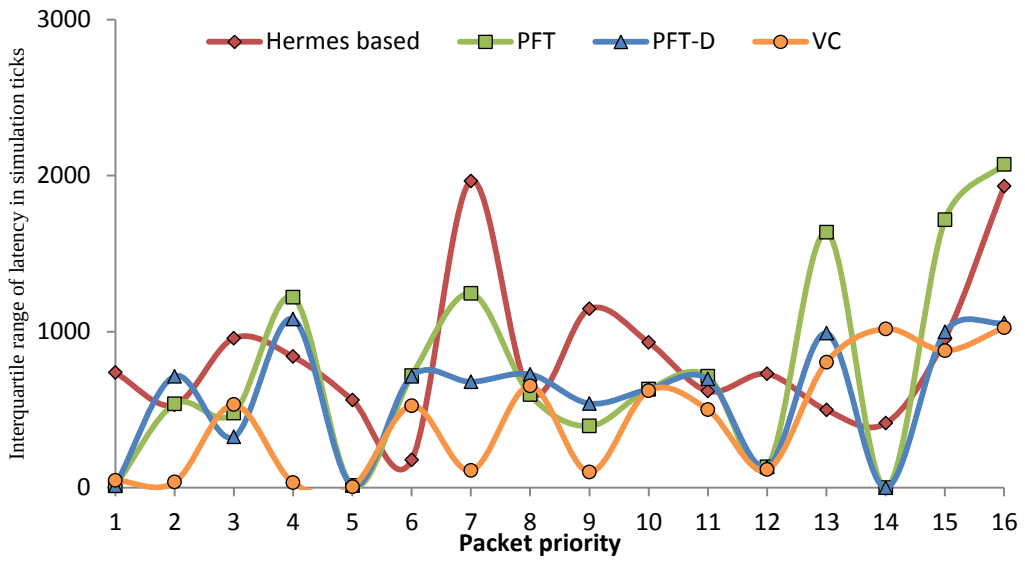


Figure 6.34: Interquartile range of latency of random traffic 5

It can be seen that the PFS based NoC shows lower latency variation with high priority packets than the Hermes based NoC and the PFS-D based NoC show even lower variation in latency. The VC based NoC is seen to show the most predictable behaviour and this can be seen more clearly in Figure 6.35 where the S-index of the four NoCs is presented.

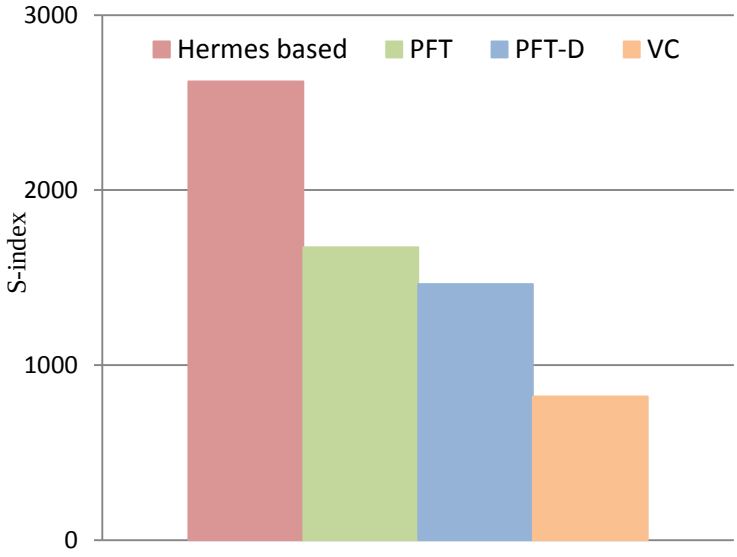


Figure 6.35: S-index plot of random traffic 5

It can be seen the VC based NoC provides the best predictability than the others followed by PFS-D and PFS while the Hermes based NoC show the worst predictability in the group.

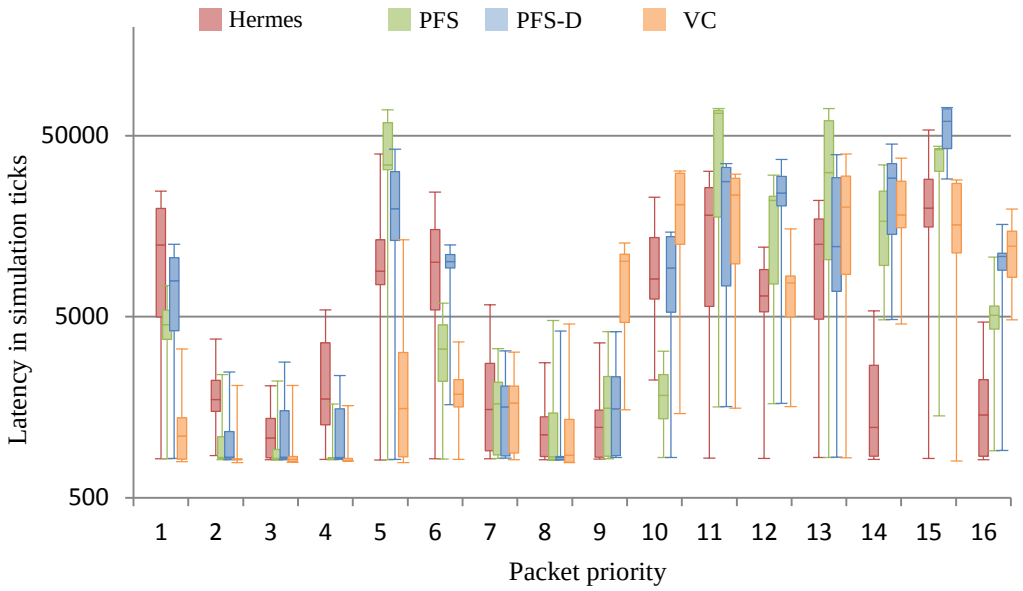


Figure 6.36: Latency box plot of random traffic 6

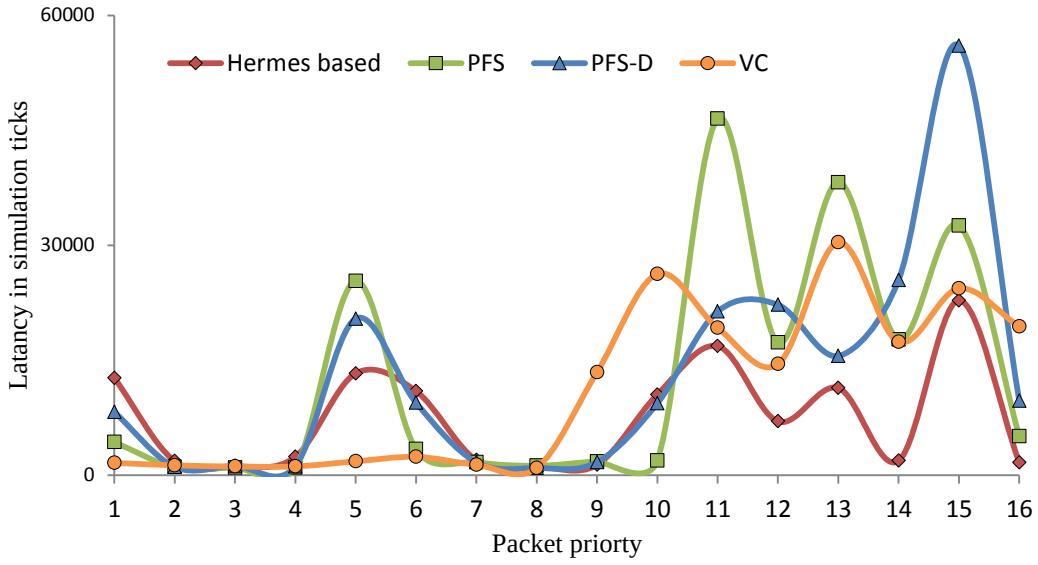


Figure 6.37: Average latency plot of random traffic 6

Figure 6.36, Figure 6.37 and Figure 6.38 show the latency box plot, average latency plot and interquartile range plot of random traffic 6. In this case, the PFS and VC based NoC showed similar performance in magnitude and variation of latency

(better than Hermes based NoC) and the PFS-D based NoC showed the best performance of the four in both aspects.

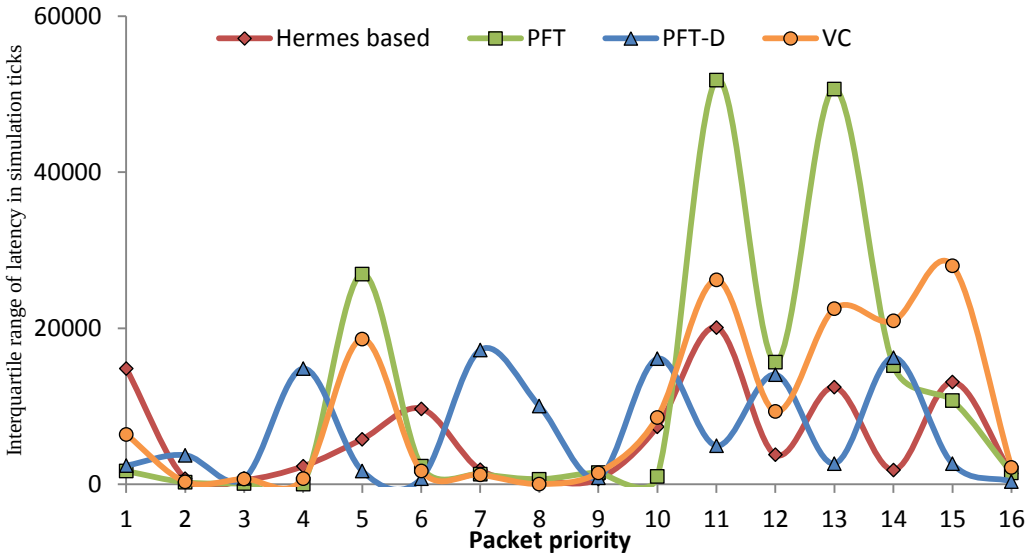


Figure 6.38: Interquartile range of latency of random traffic 6

The predictability comparison of the NoCs can be seen in Figure 6.39 where the S-index in each case is presented.

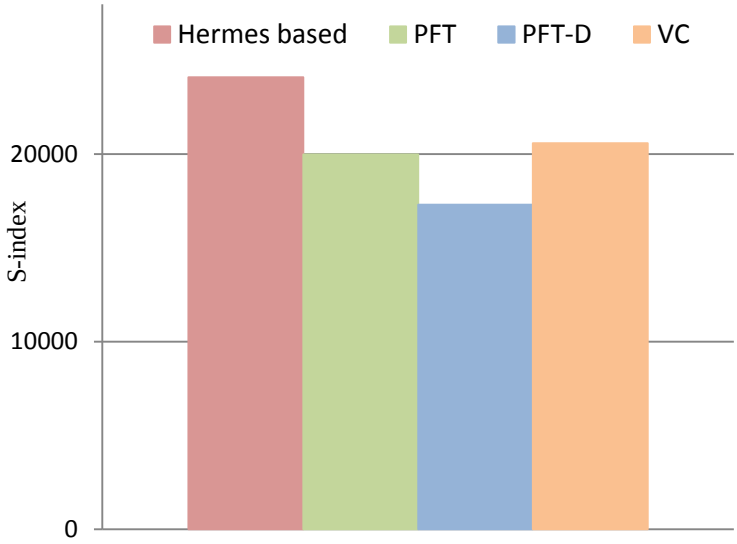


Figure 6.39: S-index plot of random traffic 6

6.5. Hardware Overhead and VC Scalability

The hardware requirements for a PFS based router compared to PFS-D design was evaluated using Xilinx Vivado and was found to be minimalistic with only 16% more lookup tables and 12 % registers on a xc7a350t Artix-7 FPGA.

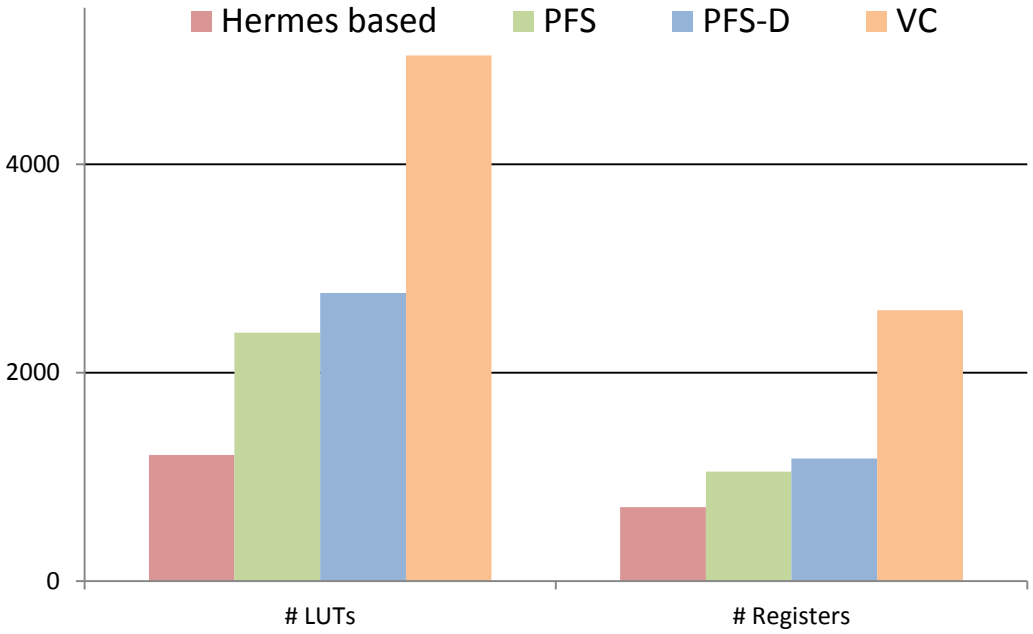


Figure 6.40: Hardware overhead

The hardware overhead of the VC based router (with 4 VCs) can also be seen in the plot and as evident from the plot, it costs almost double in terms of LUTs and registers than the PFS-D NoC. Also, it is almost four times as expensive in LUTs and registers than the Hermes based NoC. Detailed hardware overhead details is added in Appendix 3 in sections R2, R7-F, R7-FD and R8.

Furthermore, with the increase in size of the NoC or the increase in packet priority numbers, the performance of the VC based NoC will deteriorate. To counter this effect, the number of VCs would have to be increased and this would result in further increase in hardware overhead. However with PFS and PFS-D, there is no limitation in the number of packet priorities the system can handle and hence are more suited to scaling than other techniques.

The hardware overhead comparison of a 2 VC design is shown in Figure 6.41.

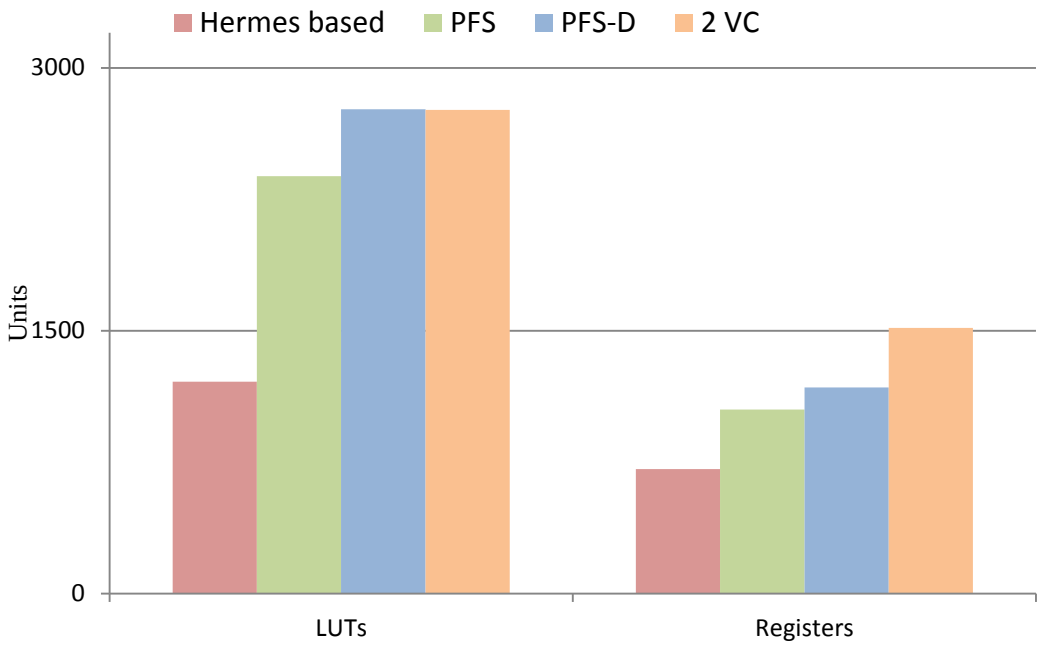


Figure 6.41: Hardware comparison with a 2 VC design

The hardware overhead is seen similar to PFS-D and the latency box plot and average latency plot (for random traffic 5) are presented as Figure 6.42 and Figure 6.43 respectively.

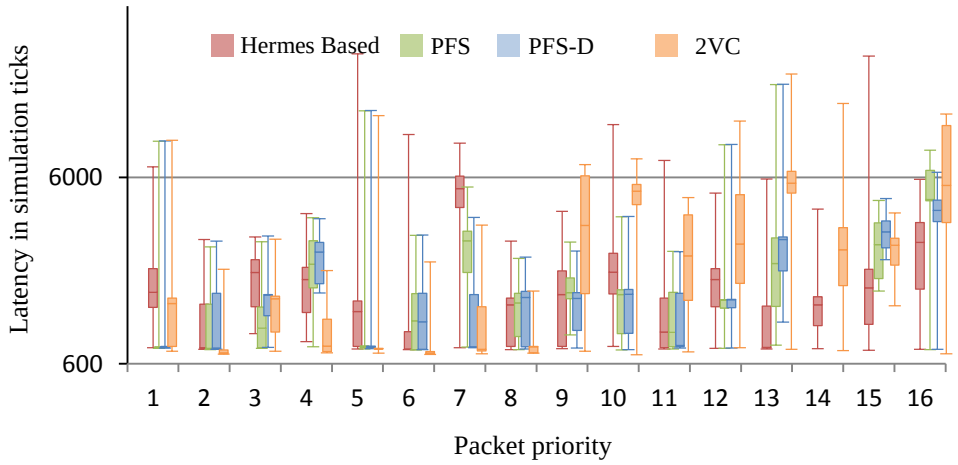


Figure 6.42: Latency comparison with a 2 VC design

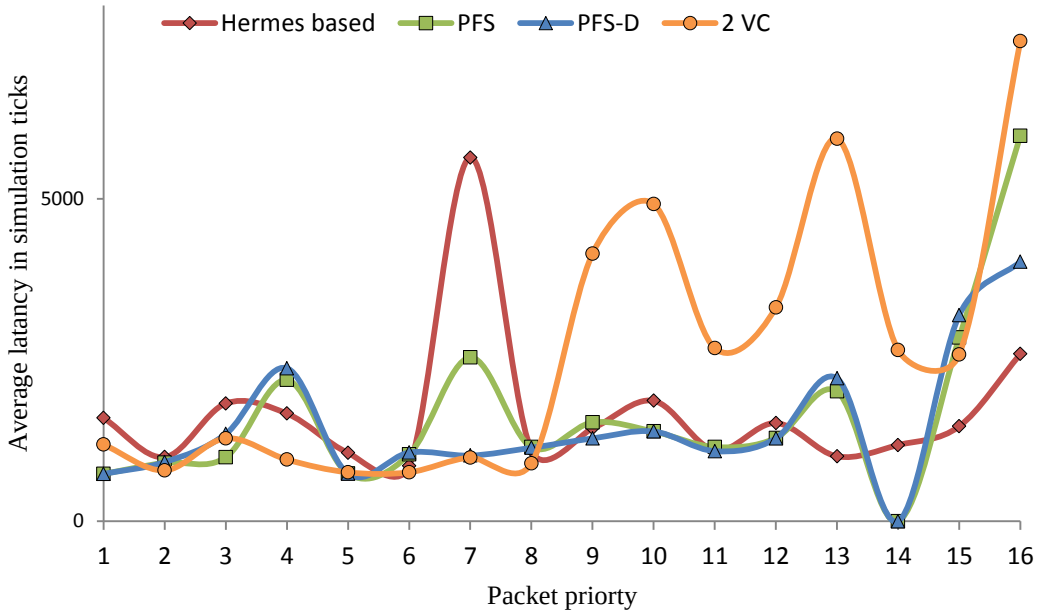


Figure 6.43: Average latency comparison with a 2 VC design

It can be seen that the performance of a 2 VC NoC is similar to PFS and PFS-D with the high priority packets (1 to 8) but the performance of the low priority packets (9 to 16) is seen poorer than the Hermes based NoC, PFS and PFS-D based NoCs.

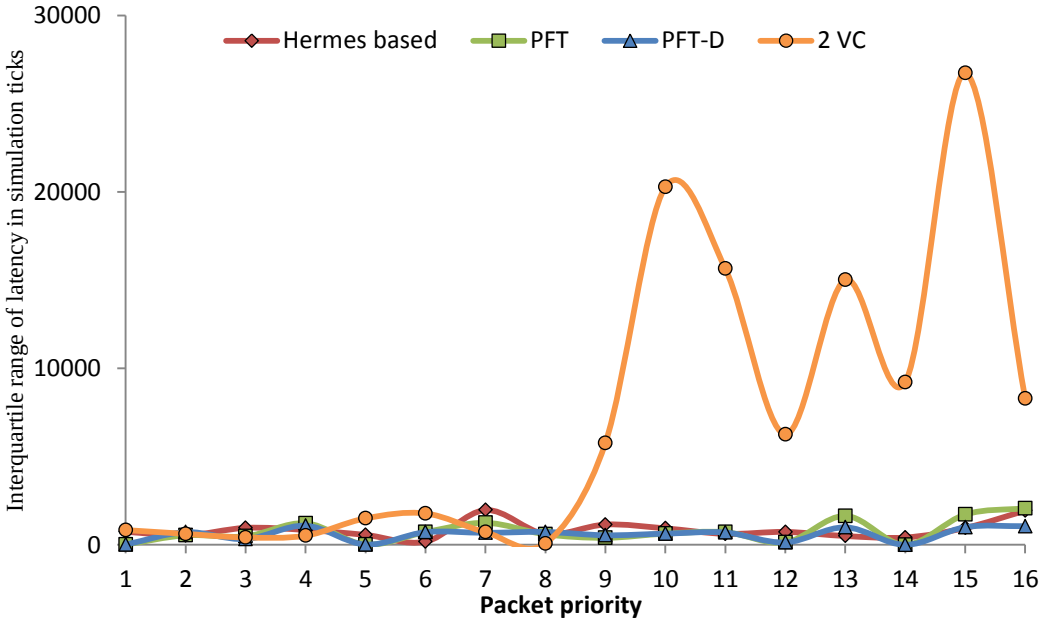


Figure 6.44: Interquartile range of latency with 2 VCs

This is evident in Figure 6.44 where the interquartile range of latency is plotted. It can be seen that the 2 VC NoC suffers has high latency variation with the lower priority spectrum of packets. Figure 6.45 show the S-index comparison between the NoCs and the degradation of performance with the switch from 4 VCs to 2 VCs is quite evident.

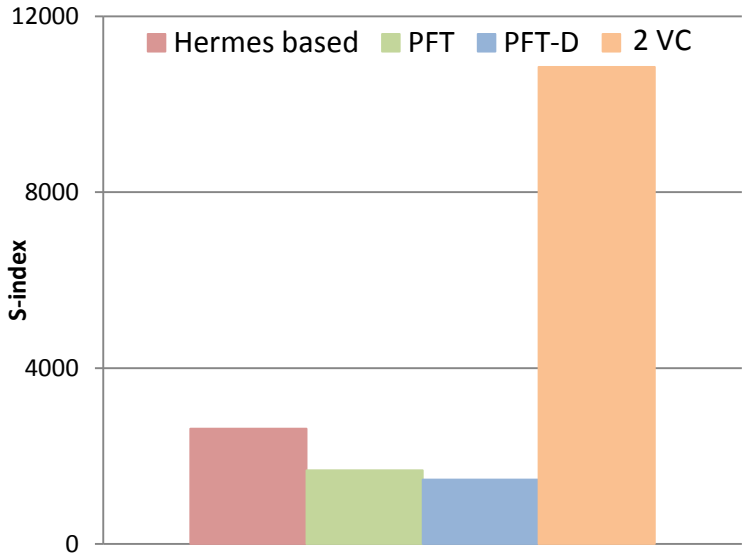


Figure 6.45: S-index plot with the 2 VC design

As evident from the plot, with the switch from 4 VC design to 2 VC design, the performance of the lower spectrum of the priority range (9 to 16) got significantly poor. With the increase in size of the NoC or packet priorities, this range of poorly performing packets would increase thus requiring increase in VC numbers for better latency performance. This will result in a linearly increased hardware overhead as evident from the hardware overhead figures in Figure 6.40 and Figure 6.41.

6.6. Summary

The chapter introduced DHARA a low overhead protocol that enables the notion of timeliness in NoC packets. DHARA aimed at using dynamic residual slack as an additional parameter for arbitration (the other being packet priority) thereby improving QoS of lower priority packets at the cost of the residual slack of higher priority packets. The protocol employed additional logic in routers to decrement

the residual slack of packets while they are forced to wait and thus keeping it up-to-date. This slack value was then used by the routers combined with the application-supplied priority of the packet to compute an instantaneous priority, which is then used as the metric for arbitration or predictability enhancement measures.

As seen in Section 2.2.5, monitoring timeliness is typically achieved in NoC packets by using time-stamping which requires a notion of the global time which can be hardware expensive in routers. There has also been research encompassing a static slack value in the packet header which is then used for arbitration decisions. However the approach does not account for the time the packet spend in the NoC waiting for arbitration and hence is not suitable for large NoCs.

The work by Berejuck et. al. in [65] employed ageing on packets as a method to introduce a timeliness parameter into NoC. In the design, packets had an age field in headers that would get incremented when the packet waits for arbitration and this field is used by arbiters to provide arbitration if packets of the same service level compete for arbitration. The drawback of the system is that under intense load there is a probability of multiple packets reaching the maximum age and hence result in lowering of QoS of packets.

As DHARA uses slack value as the timeliness parameter, the QoS of packets would not be affected regardless of the load on the NoC. Unlike [65], DHARA uses instantaneous priority to arbitrate packets which have both slack and the application supplied priority as components. So in case of high load scenarios, the NoC will perform reliably even when all the packets have zero slack due to the high contention encountered (utilising the application supplied priority component in determining instantaneous priority).

To test DHARA, the PFS based prototype (used in previous chapter) was modified to DHARA specification and was tested with varying load levels and with random as well as realistic traffic. Test result show improvement in magnitude and variation in latency of lower priority packets compared to PFS and Hermes based NoCs.

Average latency tests show more consistent (lower magnitude over the whole priority range) plots compared with the Hermes based and PFS based NoC. With Hermes based NoCs, peaks in average latency were encounter regardless of the priority value and with PFS based NoC, high peaks in average latency were encountered with lower priority packets. DHARA based PFS showed resolution of such peaks by trading the residual slack in packets. DHARA based PFS was also seen to show lower variation in average latency and magnitude of latency with increase in load. Also, DHARA showed lowering of the number of late packets (depending on priority) with synthetic, realistic and hybrid traffic.

The hardware overhead for DHARA was evaluated and was found minimalistic with only 16% more LUTs and 12% more registers. The hardware overhead of a VC based NoC with 4 VC were also evaluated and it was found to be almost double in LUTs and registers compared to PFS-D. The 4 VC NoC showed better latency performance than the PFS-D based NoC in most cases.

The 2 VC based NoC however had almost identical hardware utilisation figures compared to the PFS-D based NoC and showed similar performance with higher priority packets latencies. However with lower priority packets the 2 VC NoC showed high variation and magnitude in latency compared to both the PFS based NoC and the PFS-D based NoC depicting the scalability limitation of VC based NoCs. However, unlike VCs, as PFT, PFS or PFS-D does not depend of service levels, the techniques are completely scalable both in terms of packet priority numbers and NoC sizes.

Chapter 7

Conclusion

7.1. Thesis summary

Network-on-Chip designs are widely seen as the communication infrastructure for large many-core systems where packet predictability can be an important attribute. Though hardware inexpensive, tests show that even with priority based arbitration, non-preemptive NoC packets can have high magnitudes and variation in latency regardless of the priority value due to HOL blocking and tailbacking of packets. The techniques presented in the thesis were tested and verified to be providing improvement in variability and magnitude of latency for packets with respect to its priority value. This was achieved by using scalable techniques that resolved HOL blocking (Chapter 4) and tailbacking (Chapter 5) and by using a timeliness parameter in arbitration decisions (Chapter 6).

This validates the hypothesis tested in the thesis that "**Latency predictability can be enhanced in scalable non-preemptive NoC designs using modifications that dynamically alter arbitration policies or packet structure**".

As the techniques presented in the thesis use logic that require a fixed number of logic elements regardless of the size of the NoC, they are completely scalable. As aimed with the thesis hypothesis, the tests results confirm the predictability enhancement achieved with the techniques while ensuring complete scalability and dynamic behaviour of the routers. By resolving HOL blocking and tailbacking, the techniques presented on average a latency variability reduction of 70% (in S-index

value) and with the DHARA based timeliness a latency variability reduction of 68% compared to the Hermes based baseline.

On the contrary, even though contemporary predictability enhancement techniques like Time Division Multiplexing, Link Division Multiplexing and Virtual Channels resolve predictably degrading issues, they result in excessive hardware requirements or limitation in scalability or dynamic behaviour. While TDM based routers suffer limitation in dynamic behaviour and scalability, LDM based NoCs have high hardware overhead and limitation in scalability.

With Virtual Channels, the communication is classified into separate service levels such that a higher priority service level is allowed to utilise a communication link even if the link is being used by a lower priority service level. This is enabled by employing separate set of buffers for each service level thus resulting in increased hardware requirements. With the increase in packet priority numbers the effect of predictability enhancement would deteriorate and would require additional service levels to maintain the required predictability performance. As a result, Virtual Channel based systems are not scalable without succumbing to linearly increasing hardware overhead with each additional service level. This can be seen clearly in Chapter 6 where the latency performance of the NoC under test deteriorated both in magnitude and predictability with the switch from 4 VCs to 2 VCs.

As the techniques presented in the thesis rely of dynamic techniques that do not require additional hardware with the increase in size of the NoC, the techniques are completely scalable. As a result, the tests reveal predictability enhancement for high priority packets regardless of the NoC size. Tests also reveal that the techniques presented not only improved the magnitude and predictability of the packets compared to the non-preemptive Hermes based baseline, the predictability is seen to be comparable to a 4 VC design with only half the hardware overhead. While PFS-D had an average variability reduction (S-index value) of 68% compared to the Hermes based baseline, the 4 VC design had an average variability reduction of 58% with a hardware overhead of 182% more LUTs and 220% more registers than PFS-D.

A 2 VC version of the NoC had similar hardware overhead (0.1% less LUT and 33% less registers) compared to PFS-D, however the scalability limitation of VCs was quite evident as the predictability of the packets dropped remarkably with the reduction in the number of VCs. On the traffic scenario tested, the switch from 4 VCs to 2 caused the latency variability to increase from 31% to 414% (S-index increased 13 times) while PFS and PFS-D showed 63% and 55% variability than the Hermes based baseline showing the scalability limitation associated with VC based NoCs.

7.2. Novelty contributions

The novelty contributions presented in the thesis are added below in descending order of importance as per the author.

1) **Selective packet splitting:** Chapter 5 presented the SPS technique using which the effect of pre-emptive arbitration can be emulated (to reduce latency variability) without major hardware overheads as seen with the classical preemption technique. The classical preemption approach in NoCs use VCs for its functionality, which is hardware expensive especially with the increase in size of the NoC.

   SPS however employs splitting of packets which requires simpler hardware than VCs, and hence do not require extra hardware with the increase in size of the NoC. As a result, SPS could be a cheaper alternative than VCs in large many-core systems.

2) **Dynamic slack Hard-line Aware Router Architecture:** Chapter 6 presented DHARA, which enabled routers to use timeliness as a parameter while in arbitration decisions. Typical approaches to enable timeliness in packets employ timestamping in packets, which would require long counters in routers thus limiting its practicality. Other approaches employ static fields in packet headers to denote timeliness, however these do not account for the lateness the packet had to encounter in transit.

   With DHARA, a dynamic field is added to the packets that denote the residual slack in latency the packet has. This is in turn decremented by routers when a packet waits for arbitration thus ensuring its correctness at eve-

ry point of time. This slack value is utilised by the routers in arbitration decisions hence allowing packets of lower priority better QoS if higher priority packets have residual slack.

3) **Priority Forwarding and Tunnelling:** Chapter 4 presented the PFS technique that enables routers to resolve HOL blocking scenarios and thus resolve starvation of packets. In simple NoCs which do not support non-preemptive arbitration, HOL blocking can cause unpredictable behaviour and starvation of packets regardless of packet priority.

   With PFS, additional logic was added to the routers which enabled them to modify arbitration request priorities during HOL situations so that the issue can be resolved. Furthermore, with increase in size of the NoC, PFS does not require any additional hardware for its functionality thus ensuring scalability.

7.3. Further Work

7.3.1. Dynamic Time Multiplexed Virtual Channels (DTMVC)

The research in the thesis spawned the concept of Dynamic Time Multiplexed Virtual Channels (DTMVC) [124], in which a VC based NoC would be able to vary the intensity of predictability enhancement dynamically. The standard VC approach can cause high magnitude and significant variation in latency for lower priority VCs. Starvation of packets of lower priority VCs is even possible in scenarios presenting a continuous stream of packets from higher priority VCs (as seen in Section 0). The aim of DTMVC is to avoid this when possible, and hence reduce the magnitude and variation in latency of lower priority VC packets.

With DTMVC, the operational time of the router would be divided into recurring time frames which consists of several time slots as shown in Figure 7.1a. There would be a table in each router that will denote the priority order of the VCs in each time slot. Assume that there are four VCs; VC0, VC1, VC2 and VC3. Under the highest priority setting, VC0 will be treated as the highest priority VC followed by VC1, VC2 and VC3 (shown in Figure 7.1b) in all the time slots and hence the router will work like a classical VC based NoC providing VC0 with the best quality of service.

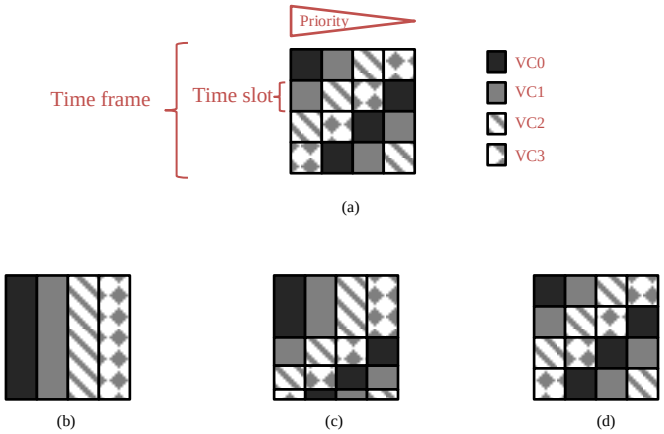


Figure 7.1: DTMVC functionality (a) Time frame (b) Time frame for highest performance setting (c) Time frame for intermediate performance setting (d) Time frame for lowest performance setting

If the latency performance of the highest priority VC is satisfactory (which can be estimated using DHARA based slack awareness or any other approach), the router can be switched to a lower performance setting where some time slots will be allocated to the other VCs (shown in Figure 7.1c), so that they can assume the highest priority momentarily and can transmit flits without getting blocked. This will improve the magnitude and variation in latency of lower priority VC packets. If the high priority VC packets are still performing satisfactorily, the router can be switched into an even lower performance setting where more time slots will be allotted to the lower priority VCs as the highest priority VC. With the lowest performance setting, all the VCs will get equal time slots thus getting even quality of service (shown in Figure 7.1d).

7.3.2. HYper Criticality Enabled NoC Architecture (HYENA)

The use of components with different levels of criticality in embedded systems brought about mixed criticality traffic flows through the communication infrastructure. As per Burns et.al. [125] “a mixed criticality system (MCS) is one that has two or more distinct levels (e.g. safety critical, mission critical and non-critical)”. Although typical mixed criticality NoC systems like [126] and [127] support multiple criticality traffic, the number of criticality levels supported during the experimental evaluation is limited to two, HI and LO. The idea of the

HYper criticality Enabled NoC Architecture (HYENA) is to provide a NoC infrastructure for multi criticality traffic supporting many criticality levels.

With HYENA, the idea is to have an extended range of criticality levels for mixed criticality traffic. With HYENA, the NoC packet headers carry an additional field along with the application-supplied priority that will specify the criticality value of the packet. Therefore, the number of criticality levels supported is unlimited and will depend on the number of bits in the header allocated for the criticality parameter.

Similar to DHARA, HYENA based routers use an equation (equation (7.1) to determine the instantaneous priority of a packet for all arbitration and predictability enhancement efforts.

$$P_i = (P_a >> D_a) + (C >> D_c) \quad \textbf{(7.1)}$$

(P_i – Instantaneous priority, P_a – Application supplied priority, D_a – Divider index a, C- Criticality, D_c – Divider index c)

As seen in the equation, in HYENA based routers, the criticality value (C) in the header is combined with the application supplied priority (P_a) to generate the instantaneous priority (P_i) of the packet. As evident from the equation, the instantaneous priority is generated by the routers using an addition operation between two components. The first component in the equation '$P_a >> D_a$' represents the application supplied priority component and its weightage in computing the instantaneous priority is determined by D_a. For example, setting D_a to 0,1 and 2 realise the function of P_a, $P_a/2$ and $P_a/4$ respectively thus varying its weightage.

Similarly the second component in the equation '$C >> D_c$' represents the criticality of the packet and its weightage in calculating the instantaneous priority can be varied by setting the value of D_c as seen with the first component. As the equation use two shift operations and an addition, implementation of the equation is efficient and simple in hardware terms.

To ensure scalability and to resolve HOL blocking and tailbacking, HYENA based NoCs will use PFS logic. To enable, criticality change in the packet that is mid-way in transmission, the Network Interface will be enabled with logic that

will send a C-flit (criticality change flit) that will be transmitted like an ordinary flit to the destination. Unlike an ordinary flit, C-flits will however change the criticality value of the packet through the routers it pass through. If the packet is not blocked mid-way during its transmission, the C-flit will get to the destination without playing any role in the NoC's functionality.

However if the packet gets blocked, the router will have additional logic (similar to Priority Forwarding) that would forward the criticality value to routers down the line until the blocked header is reached and is updated. This will allow the routers to completely convey the criticality change and this happens only if it is necessary unlike typical systems like [103] where the criticality change is flooded throughout the NoC to all routers.

7.3.3. Power Analysis and moving into ASIC

As per Chen et.al. in [98] almost 64% of the total leakage power of the router is consumed by the buffers. Unlike VC or LDM based approaches, the techniques presented in the thesis do not require wide use of buffers and hence the routers are likely to dissipate lower dynamic and static power. However, as power dissipation is outside the scope of the thesis this is considered future work.

Similarly, further work will involve simulation of the designs in ASIC platforms so that the performance and the related overhead can be compared with FPGA based implementation.

Appendix 1- Traffic Scenarios

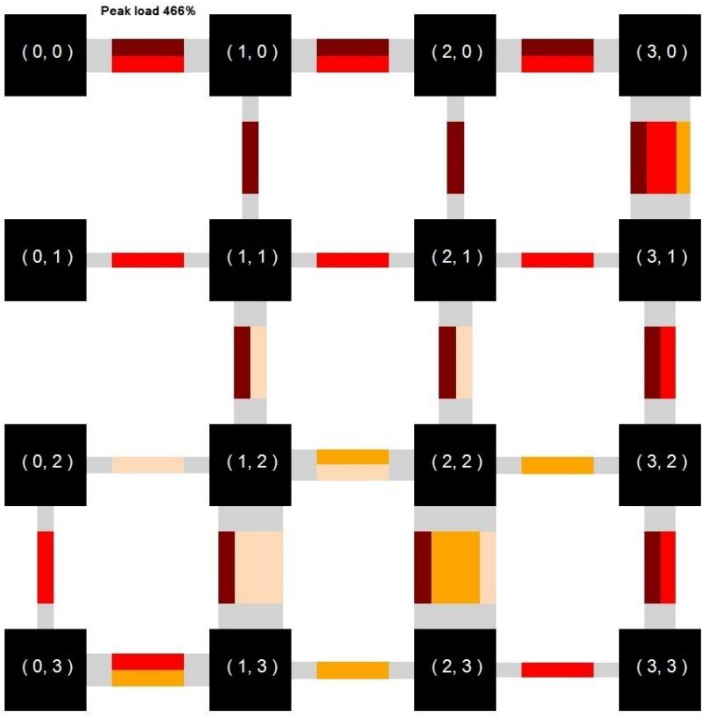
In this section, some of the traffic patterns used in the tests are added. The column titled priority presents the application supplied priority of the packet flow from the source to the destination as mentioned in the respective columns. Start time specifies the time when the packet was injected for the first time. Packet size column present the length of the packet in flits and the period column show the clock cycles the IP would spend idle before starting transmission of a new packet after transmitting one. Some of the tests utilised packet generators that have four fixed destination values. In such generators, the packet generator would switch between each of those destination (when sending packets) in a round robin fashion.

Along with table, the traffic pattern is also visualised as shown below.

The black squares depict routers with their addresses shown in white within them. The grey rectangles connecting the routers show a packet flow path and the width of the grey rectangle shows the utilisation. The link with the maximum average utilisation (peak load) is scaled to the same size as the routers and the other links are scaled accordingly. The grey rectangles are divided inside into section depicting flow priorities. Marron colour show the contribution of the packets with the highest 25 % of priorities (for example priority 1 to 4 if there are 16 packet priorities) towards the total link utilisation followed by red, orange and light orange depicting the utilisation of packets with decreasing priorities (5 to 8, 9 to 12 and 13 to 16 respectively if there are 16 packet priorities).

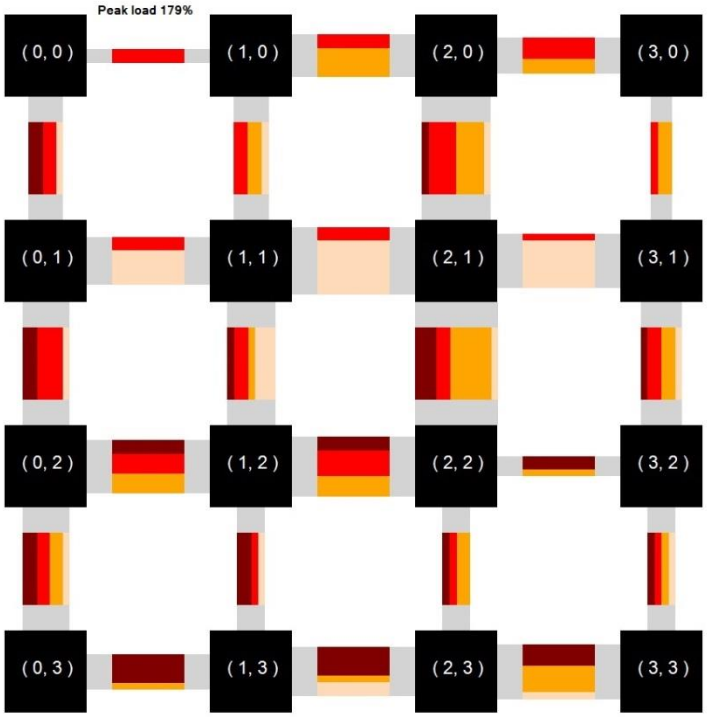
Appendix 1a

Priority	Source	Destination	Start time	Packet size	Period
1	10	13	153	50	53
2	20	23	305	50	50
3	33	30	126	50	54
4	00	30	48	50	51
5	30	00	229	50	54
6	13	02	155	50	54
7	23	30	257	50	59
8	01	30	275	50	59
9	31	30	248	50	59
10	03	22	44	50	52
11	12	23	245	50	59
12	32	23	139	50	50
13	21	23	75	50	55
14	22	13	307	50	54
15	02	13	203	50	53
16	11	13	213	50	54



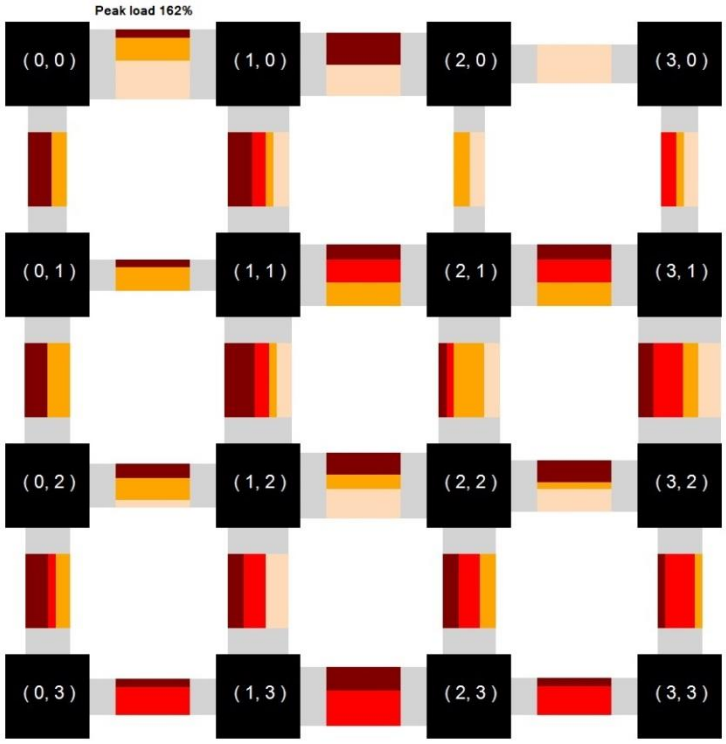
Appendix 1b

Priority	Source	Destination				Start time	Packet size	Period
		1	2	3	4			
1	12	13	21	12	00	37	500	805
2	03	02	11	33	32	158	500	813
3	23	13	00	33	20	206	500	826
4	32	21	02	31	32	202	500	884
5	30	22	32	12	23	148	500	807
6	22	00	03	01	13	190	500	894
7	21	33	20	03	02	224	500	892
8	00	21	01	12	00	240	500	867
9	10	12	22	22	33	11	500	814
10	33	21	20	23	02	229	500	867
11	20	22	31	11	20	62	500	819
12	02	31	03	22	21	26	500	870
13	01	00	31	11	33	283	500	802
14	31	01	03	12	22	110	500	890
15	11	31	21	12	20	183	500	837
16	13	33	10	13	23	298	500	822



Appendix 1c

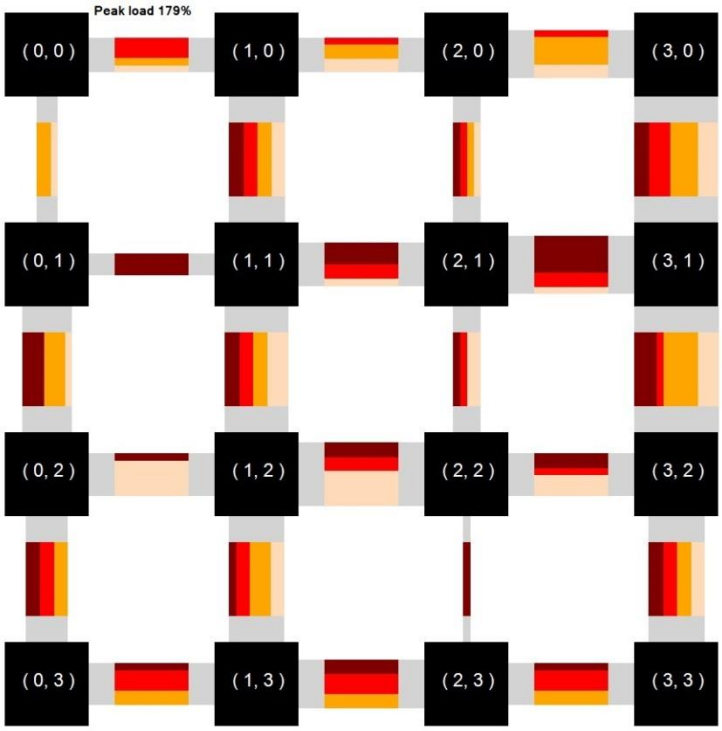
Priority	Source	Destination				Start time	Packet size	Period
		1	2	3	4			
1	32	03	32	11	01	131	500	896
2	20	11	13	03	13	176	500	800
3	21	32	00	12	33	296	500	847
4	13	00	22	21	33	111	500	822
5	33	31	31	12	21	148	500	846
6	03	22	22	02	10	194	500	897
7	11	32	11	31	30	218	500	830
8	23	30	03	10	32	99	500	880
9	10	02	11	00	02	248	500	837
10	01	23	03	20	20	251	500	823
11	31	21	22	33	23	262	500	823
12	02	03	21	30	11	268	500	880
13	12	02	32	30	13	216	500	852
14	30	20	20	00	30	31	500	816
15	22	13	10	20	31	233	500	822
16	00	30	13	22	32	170	500	820



Appendix 1d

Priority	Source	Destination				Start time	Packet size	Period
		1	2	3	4			
1	00	30	03	32	10	131	500	896
2	10	01	12	33	20	176	500	800
3	20	33	01	30	10	296	500	847
4	30	31	33	12	31	111	500	822
5	01	30	30	10	13	148	500	846
6	11	12	12	33	20	194	500	897
7	21	02	02	30	03	218	500	830
8	31	13	30	10	00	99	500	880
9	02	13	32	02	32	248	500	837
10	12	13	10	13	12	251	500	823
11	22	21	20	32	33	262	500	823
12	32	01	33	31	00	268	500	880
13	03	00	31	10	20	216	500	852
14	13	22	12	21	22	31	500	816

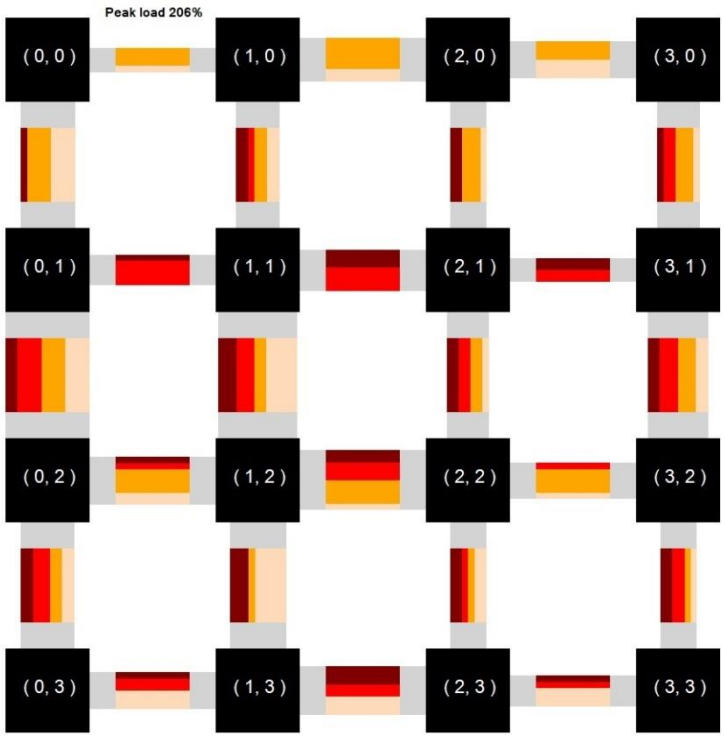
15	23	33	30	13	33	233	500	822
16	33	00	13	31	31	170	500	820



Appendix 1e

Priority	Source	Destination				Start time	Packet size	Period
		1	2	3	4			
1	12	13	23	03	20	86	500	847
2	23	00	10	10	30	197	500	825
3	31	31	23	33	12	137	500	896
4	11	11	02	21	20	99	500	892
5	21	22	31	12	03	295	500	892
6	01	10	02	33	21	264	500	810
7	22	11	11	01	30	172	500	836
8	13	21	01	30	02	197	500	863
9	20	22	03	33	32	272	500	874
10	00	12	02	21	01	139	500	859
11	32	02	01	03	00	167	500	853

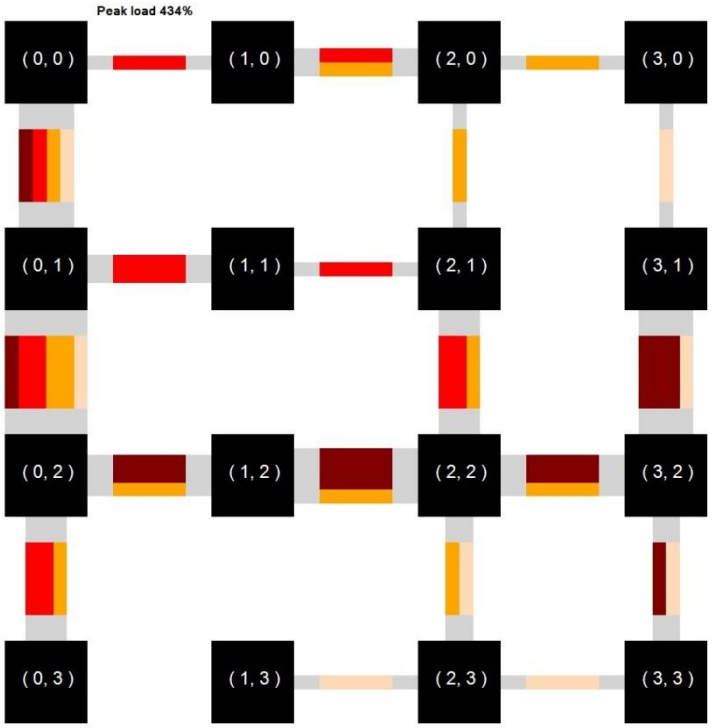
12	10	32	20	23	13	128	500	800
13	33	11	10	33	23	289	500	806
14	02	00	31	11	00	128	500	837
15	30	13	33	23	03	236	500	807
16	03	12	22	00	11	226	500	820



Appendix 1f

Priority	Source	Destination	Start time	Packet size	Period
1	22	00	134	800	1105
2	12	31	281	800	1105
3	02	31	69	800	1107
4	33	31	202	800	1153
5	01	22	229	800	1101
6	11	03	67	800	1104
7	20	03	36	800	1108
8	21	22	201	800	1103
9	30	20	196	800	1104

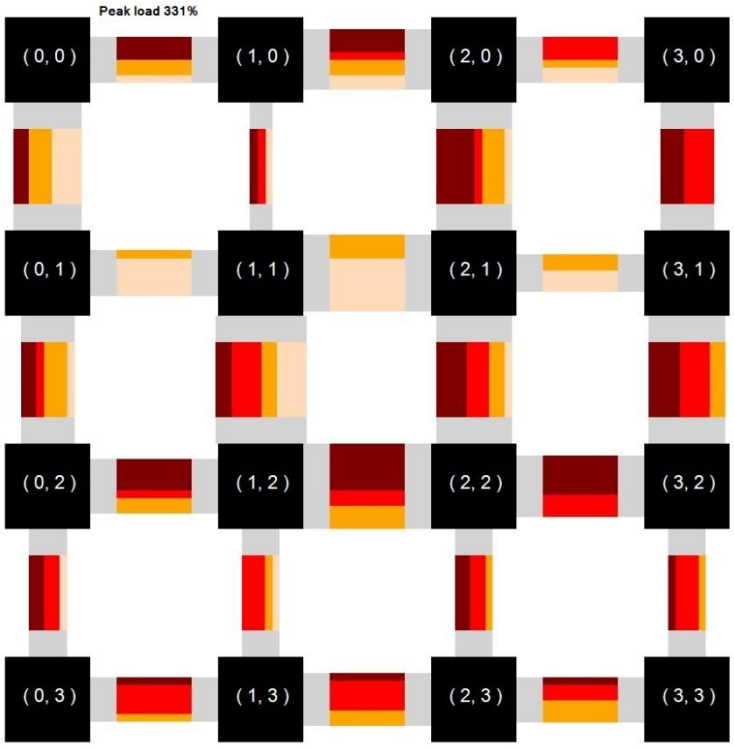
10	32	00	141	800	1106
11	03	01	121	800	1108
12	10	23	24	800	1109
13	00	02	302	800	1104
14	31	31	88	800	1108
15	13	30	159	800	1106
16	23	22	169	800	1103



Appendix 1g

Priority	Source	Destination				Start time	Packet size	Period
		1	2	3	4			
1	00	21	22	22	03	276	807	722
2	10	22	30	02	02	242	801	729
3	20	31	33	32	13	118	801	718
4	30	30	01	11	30	246	802	725
5	01	00	12	22	00	57	803	740
6	11	32	20	31	00	167	803	706
7	21	01	12	03	00	207	806	708

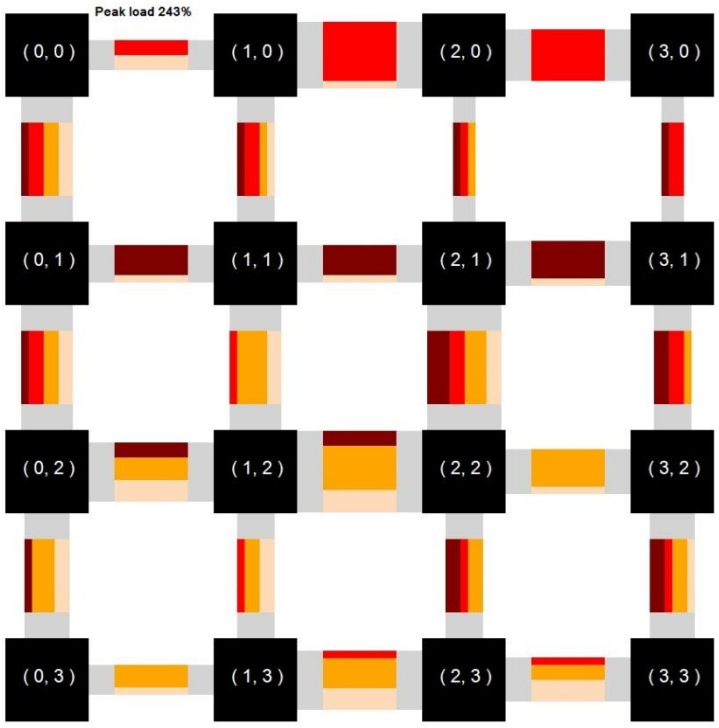
8	31	31	20	13	12	159	801	748
9	02	31	32	32	20	144	801	700
10	12	30	11	10	30	201	806	701
11	22	22	11	01	02	200	806	710
12	32	21	03	32	11	272	800	747
13	03	30	03	11	20	211	800	773
14	13	01	11	31	21	59	804	705
15	23	00	30	22	20	221	800	725
16	33	31	11	03	20	299	805	730



Appendix 1h

Priority	Source	Destination				Start time	Packet size	Period
		1	2	3	4			
1	00	21	22	22	03	276	807	722
2	02	31	32	32	20	144	801	700
3	23	00	30	22	20	221	800	725
4	12	30	11	10	30	201	806	701
5	32	21	03	32	11	272	800	747

6	20	31	33	32	13	118	801	718
7	13	01	11	31	21	59	804	705
8	03	30	03	11	20	211	800	773
9	11	32	20	31	00	167	803	706
10	33	31	11	03	20	299	805	730
11	22	22	11	01	02	200	806	710
12	10	22	30	02	02	242	801	729
13	30	30	01	11	30	246	802	725
14	01	00	12	22	00	57	803	740
15	21	01	12	03	00	207	806	708
16	31	31	20	13	12	159	801	748



Appendix 1i

Priority				Source	Destination				Start time				Packet size				Period			
1	2	3	4		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	7	36	38	01	11	13	02	12	507	454	122	307	256	256	873	873	640	640	16000	32000
2	**12**	**40**	**55**	**30**	**20**	**20**	**11**	**11**	**372**	**513**	**140**	**195**	**256**	**256**	**167**	**145**	**640**	**640**	**271**	**229**
5	31	32	50	03	13	12	02	11	411	423	231	239	256	27	27	5	640	8000	8000	365

6	**34**	**61**	**47**	**02**	**12**	**01**	**11**	**11**	**535**	**128**	**574**	**282**	**256**	**218**	**152**	**176**	**640**	**16000**	**210**	**318**
8	33	27	57	00	10	32	02	11	389	244	250	487	256	12	54	101	640	8000	1600	352
9	**42**	**63**	**54**	**33**	**23**	**11**	**11**	**11**	**144**	**115**	**261**	**495**	**256**	**129**	**123**	**148**	**640**	**250**	**268**	**208**
10	30	35	41	32	22	10	33	11	147	394	184	563	256	27	54	109	640	1600	16000	288
11	**25**	**29**	**56**	**31**	**21**	**12**	**32**	**11**	**365**	**144**	**478**	**300**	**256**	**27**	**54**	**146**	**640**	**1600**	**1600**	**298**
13	3	46	44	13	22	12	11	11	391	140	328	552	12	27	126	178	640	640	275	257
14	**24**	**26**	**37**	**12**	**22**	**30**	**10**	**01**	**423**	**305**	**306**	**456**	**12**	**27**	**54**	**54**	**640**	**1600**	**1600**	**32000**
15	22	58	39	11	22	21	11	11	291	250	574	589	12	51	140	127	640	640	232	232
16	**60**	**64**	**43**	**10**	**22**	**11**	**11**	**11**	**590**	**221**	**366**	**153**	**12**	**199**	**167**	**101**	**640**	**315**	**220**	**220**
17	28	45	59	23	11	32	11	11	476	398	516	109	12	27	121	107	640	1600	221	266
18	**21**	**48**	**51**	**22**	**11**	**21**	**11**	**11**	**337**	**228**	**414**	**371**	**12**	**51**	**115**	**193**	**640**	**640**	**330**	**301**
19	23	49	52	21	11	02	11	11	331	276	302	234	12	218	105	124	640	640	395	212
20	**4**	**62**	**53**	**20**	**11**	**12**	**11**	**11**	**148**	**380**	**447**	**556**	**12**	**27**	**183**	**102**	**640**	**640**	**308**	**383**

Appendix 1j

Priority	Source	Destination				Start time	Packet size	Period
		1	2	3	4			
1	44	05	53	55	35	180	800	1199
2	20	22	50	12	45	300	800	1237
3	05	40	34	11	54	78	800	1250
4	02	25	21	42	05	269	800	1246
5	50	15	52	45	04	233	800	1280
6	30	22	04	54	35	247	800	1124
7	34	15	44	13	53	50	800	1100
8	41	51	01	02	05	241	800	1245
9	52	05	22	54	24	187	800	1253
10	31	24	52	13	10	242	800	1215
11	15	45	03	51	11	109	800	1259
12	03	45	25	24	25	58	800	1142
13	04	51	15	10	52	95	800	1272
14	51	51	51	10	21	188	800	1119
15	42	44	20	42	00	278	800	1154
16	21	52	02	51	33	104	800	1112
17	43	35	53	40	55	221	800	1140
18	14	30	25	10	15	45	800	1191

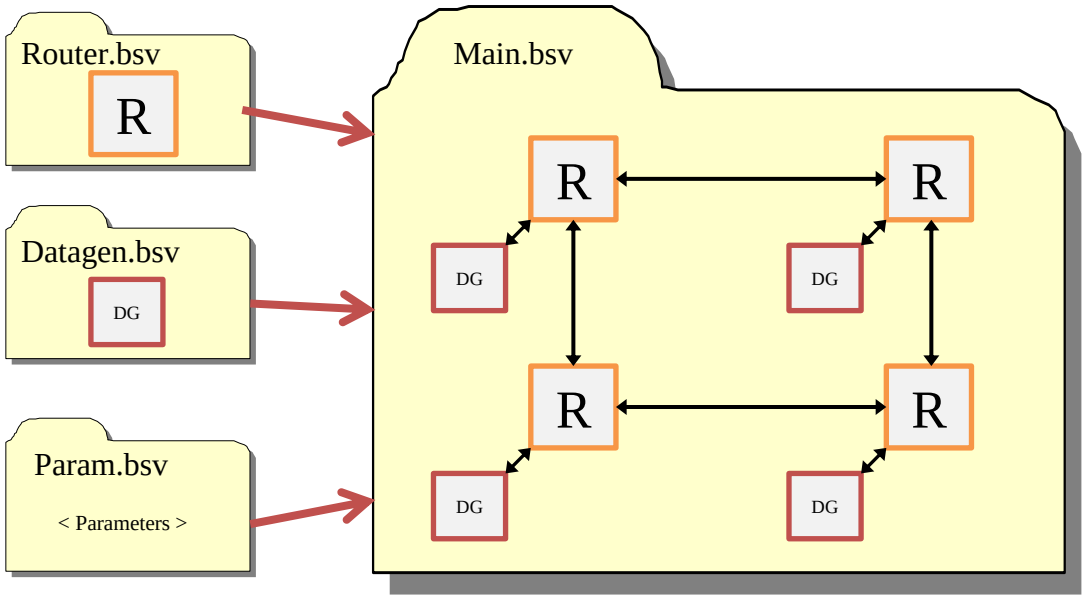
19	24	14	01	23	00	10	800	1152
20	12	53	25	03	03	98	800	1109
21	45	20	33	31	15	98	800	1231
22	25	30	51	23	23	188	800	1216
23	32	45	02	33	22	13	800	1259
24	54	20	12	50	42	68	800	1158
25	23	14	34	45	13	306	800	1169
26	10	23	44	01	43	146	800	1121
27	01	32	24	42	22	272	800	1273
28	22	30	12	42	43	298	800	1123
29	13	25	33	11	52	132	800	1104
30	53	44	02	51	41	34	800	1198
31	11	21	31	35	03	306	800	1124
32	40	10	14	32	35	132	800	1194
33	55	22	31	31	55	87	800	1144
34	33	51	45	21	04	230	800	1205
35	00	00	13	11	12	145	800	1298
36	35	24	10	23	24	165	800	1278

Appendix 2- Prototypes

The following table presents the prototype designation and its functionality specification. The open source Bluespec System Verilog implementation of the key models, R2, R3, R7-F, R7-FD and R8 can be downloaded from the URL provided in the respective specification column.

Designation	***Specification***
R1	*Hermes based with round robin arbitration*
R2	*Hermes based with priority based arbitration* *(URL: https://drive.google.com/open?id=0B3kfh6Qv5__WaEt4LTFJQnVFOUE)*
R3	***PFT enabled*** *(URL: https://drive.google.com/open?id=0B3kfh6Qv5__WT3R5UXU1ck04Sm8)*
R7-F	***PFS enabled*** *(URL: https://drive.google.com/open?id=0B3kfh6Qv5__WdVc4QURmbUNabkk)*
R7-FD	***PFS-D (PFS+DHARA) enabled*** *(URL: https://drive.google.com/open?id=0B3kfh6Qv5__WbTNVbzlWdGVOMkU)*
*R7-FH**	*PFS-H (PFS+HYENA) enabled*
R8	***Virtual Channel based*** *(URL: https://drive.google.com/open?id=0B3kfh6Qv5__WRkdnbWVveHd0RE0)*
R9#	*'Dynamic Time Multiplexed Virtual Channel' enabled*
*N1**	*'Dynamic Time Multiplexed Virtual Channels with DHARA' enabled*

Models R7-FH*, R9# and N1* are experimental designs# and conceptual designs* as part of further work.



As shown in the above figure, the implementation infrastructure consists of four files. ‘Router.bsv’ is the router design and it utilises some values from inside the parameters file, ‘Param.bsv’ for setting its internal configuration (like buffer size and packet width).

The data generator\receptor module is contained in ‘Datagen.bsv’ and these three files are used by the master file ‘Main.bsv’. ‘Main.bsv’ consists of code that replicates and interconnects the routers and data generator\receptors as per the parameters in ‘Param.bsv’. With more advanced designs in the thesis having extra connection lines, ‘Main.bsv’ would also deal with the configuration and connection of those.

Appendix 3- Hardware Overhead

R2

Adders	
2 Input 10 Bit	10
Registers	
26 Bit	20
10 Bit	15
7 Bit	6
3 Bit	10
2 Bit	5
1 Bit	20
Muxes	
6 Input 26 Bit	5
2 Input 10 Bit	10
2 Input 3 Bit	35
2 Input 2 Bit	5
2 Input 1 Bit	30

BlackBox cell count	
BUFG	1
LUT1	3
LUT2	156
LUT3	103
LUT4	188
LUT5	319
LUT6	949
FD	280
FDE	260
FDRE	170
IBUF	150
OBUF	140
Primitives	
LUT6	949
FDCE	540
LUT5	319
LUT4	188
FDRE	170
LUT2	156
IBUF	150
OBUF	140
LUT3	103
LUT1	3
BUFG	1
LUT6	949
FDCE	540
LUT5	319
LUT4	188

R3

Adders	
2 Input 10 Bit	10
2 Input 4 Bit	1
2 Input 1 Bit	5
Registers	
26 Bit	20
16 Bit	26
10 Bit	15
7 Bit	10
4 Bit	11
3 Bit	16
2 Bit	11
1 Bit	56
Muxes	
6 Input 26 Bit	5
10 Input 16 Bit	1
2 Input 16 Bit	13
2 Input 10 Bit	10
4 Input 7 Bit	5
2 Input 4 Bit	2
2 Input 3 Bit	60
6 Input 3 Bit	5
2 Input 2 Bit	5
6 Input 2 Bit	1
2 Input 1 Bit	80

BlackBox cell count	
BUFG	1
LUT1	7
LUT2	320
LUT3	111
LUT4	196
LUT5	328
LUT6	1200
MUXF7	30
MUXF8	15
FD	488
FDE	388
FDR	5
FDRE	395
IBUF	222
OBUF	212
Primitives	
LUT6	1200
FDCE	836
FDRE	400
LUT5	328
LUT2	320
IBUF	222
OBUF	212
LUT4	196
LUT3	111
MUXF7	30
MUXF8	15
LUT1	7
BUFG	1

R7-F

Adders	
2 Input 10 Bit	10
2 Input 4 Bit	1
Registers	
27 Bit	20
10 Bit	10
7 Bit	41
4 Bit	11
3 Bit	21
2 Bit	11
1 Bit	40
Muxes	
6 Input 27 Bit	5
2 Input 27 Bit	10
2 Input 10 Bit	10
4 Input 7 Bit	5
10 Input 7 Bit	1
2 Input 4 Bit	2
6 Input 3 Bit	5
4 Input 3 Bit	5
2 Input 3 Bit	60
2 Input 2 Bit	20
6 Input 2 Bit	1
2 Input 1 Bit	86

BlackBox cell count	
BUFG	1
LUT1	1
LUT2	195
LUT3	185
LUT4	288
LUT5	234
LUT6	784
FD	368
FDE	326
FDRE	103
IBUF	192
OBUF	190
Primitives	
LUT6	784
FDCE	694
LUT4	288
LUT5	234
LUT2	195
IBUF	192
OBUF	190
LUT3	185
FDRE	103
BUFG	1
LUT1	1

R7-FD

Adders	
2 Input 10 Bit	5
2 Input 7 Bit	9
2 Input 4 Bit	1
2 Input 3 Bit	1
Registers	
34 Bit	36
10 Bit	11
7 Bit	56
4 Bit	11
3 Bit	25
1 Bit	68
Muxes	
3 Input 34 Bit	5
2 Input 34 Bit	5
6 Input 34 Bit	5
7 Input 10 Bit	5
2 Input 10 Bit	5
6 Input 10 Bit	1
8 Input 7 Bit	5
6 Input 7 Bit	9
2 Input 7 Bit	5
10 Input 7 Bit	1
2 Input 4 Bit	2
8 Input 3 Bit	5
5 Input 3 Bit	1
3 Input 3 Bit	5
2 Input 3 Bit	16
6 Input 3 Bit	13
4 Input 3 Bit	5
2 Input 2 Bit	5
7 Input 1 Bit	5
2 Input 1 Bit	87

BlackBox cell count	
BUFG	1
CARRY4	2
LUT1	9
LUT2	216
LUT3	149
LUT4	197
LUT5	565
LUT6	1447
MUXF7	74
MUXF8	7
XORCY	1
FD	624
FDE	566
FDR	7
FDRE	438
FDS	1
IBUF	227
OBUF	225
Primitives	
LUT6	1447
FDCE	1190
LUT5	565
FDRE	445
IBUF	227
OBUF	225
LUT2	216
LUT4	197
LUT3	149
MUXF7	74
LUT1	9
MUXF8	7
CARRY4	3
BUFG	1
FDSE	1

R8

Adders	
2 Input 10 Bit	5
2 Input 2 Bit	120
Registers	
84 Bit	20
28 Bit	60
10 Bit	20
8 Bit	20
3 Bit	60
2 Bit	22
1 Bit	256
RAMs	
84 Bit	20
3 Bit	20
Muxes	
2 Input 28 Bit	120
6 Input 28 Bit	5
4 Input 28 Bit	5
4 Input 10 Bit	5
7 Input 10 Bit	20
4 Input 3 Bit	5
2 Input 3 Bit	140
4 Input 2 Bit	20
6 Input 2 Bit	16
2 Input 2 Bit	130
2 Input 1 Bit	590
4 Input 1 Bit	14

BlackBox cell count	
BUFG	1
LUT1	1
LUT2	725
LUT3	447
LUT4	560
LUT5	998
LUT6	2965
MUXF7	1
RAM16X1D	20
RAM32M	100
FD	436
FDE	840
FDRE	1226
FDSE	96
IBUF	220
OBUF	210
Primitives	
LUT6	2965
FDCE	1276
FDRE	1226
LUT5	998
LUT2	725
RAMD32	640
LUT4	560
LUT3	447
IBUF	220
OBUF	210
RAMS32	200
FDSE	96
BUFG	1
LUT1	1
MUXF7	1

Appendix 4- S-index test

As mentioned in 3.1.1, the S-index allows latency variability comparison between two NoCs under the same traffic scenario over the whole priority range. To understand the effect of each priority level towards the S-index, consider the table below.

In the table, eight traffic scenarios (T1 to T8) are shown. Under each traffic scenario, the interquartile range of packet latencies of the eight packet priorities can be seen in the respective columns. The last row shows the S-index value of each traffic scenario.

Priority	T1	T2	T3	T4	T5	T6	T7	T8
1	10	30	40	60	10	10	10	10
2	20	30	50	30	20	20	20	20
3	30	30	60	30	30	30	30	30
4	40	40	40	40	40	40	40	40
5	50	50	50	50	80	60	120	140
6	60	60	60	60	100	100	150	180
7	70	70	70	70	120	150	200	220
8	80	80	80	80	140	200	250	270
S-index	**80**	**105**	**135**	**135**	**107**	**115**	**148**	**163**

It can be seen that with T1, the interquartile range of latencies increase with the decrease in packet priority linearly thus amounting to an S-index of 80. S-index equation is formulated in such a way that an increase in latency variation of the higher priority packets bring about major increase in S-index. This can be seen in T2 where the variation in latency of packet 1 and 2 are higher than T1 thus resulting in the increase of S-index from 80 to 105. The same effect can be seen in T3 where the latency variation of packets 1, 2 and 3 are increased further thus resulting in an increased S-index of 135. In T4, only packet 1 has a high latency variation and all the other packets have latency variation similar to T1. As packet 1 is the highest priority packet, its high latency variation is critical and hence it is seen to reflect in its S-index (of 135).

The lower priority packets however is supposed to show a lower impact on S-index than the higher priority packets. For example, in T5, high priority packets (1 to 4) have latency variation similar to T1 but the lower priority packets (5 to 8) has higher variability than before. The effect of this can be seen in S-index (increased from 80 to 107) however the effect is minor compared to how it behaved with the higher priority packets. With

T6, T7 and T8, it can be seen that the effect of the lower priority packets significantly impacts the S-index value only when the latency variation with them are extremely high as it should be. The thesis assumes that the weightage of packets in determining the S-index decreases linearly with the decrease in packet priority value. As mentioned in section 3.1.1, this is done by setting the weightage relation in the S-index equation to 1.

Appendix 5- Simulator functionality validation

The following section depicts the functionality of the prototypes during simulation. Extra code was added to the designs to export functionality details during simulation into a log file.

R2

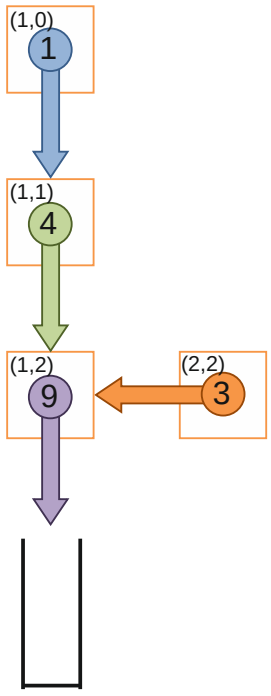
The following is the log file of the functionality of the routers when router (0,0) sends a 20 flit wide packet to router (2,0).

11<= Sim tick **Packet injected at 0,0 to 2,0 with priority 1 with packet size 20**
12<= Sim tick
13<= Sim tick
14<= Sim tick **Router 0, 0 port local arbitration request to east**
15<= Sim tick **Normal arbitration at router 0, 0 to port local**
16<= Sim tick Header send at router 0, 0 port local to port east
17<= Sim tick Payload send at router 0, 0 port local to port east with 19 flits left
18<= Sim tick Payload send at router 0, 0 port local to port east with 18 flits left
19<= Sim tick Payload send at router 0, 0 port local to port east with 17 flits left
Router 1, 0 port west arbitration request to east
20<= Sim tick Payload send at router 0, 0 port local to port east with 16 flits left
21<= Sim tick Payload send at router 0, 0 port local to port east with 15 flits left
22<= Sim tick
23<= Sim tick
24<= Sim tick **Normal arbitration at router 1, 0 to port local**
25<= Sim tick Header send at router 1, 0 port west to port east
26<= Sim tick Payload send at router 1, 0 port west to port east with 19 flits left
27<= Sim tick Payload send at router 1, 0 port west to port east with 18 flits left
28<= Sim tick Payload send at router 0, 0 port local to port east with 14 flits left
Payload send at router 1, 0 port west to port east with 17 flits left
Router 2, 0 port west arbitration request to local
29<= Sim tick Payload send at router 0, 0 port local to port east with 13 flits left
Payload send at router 1, 0 port west to port east with 16 flits left
Normal arbitration at router 2, 0 to port local
30<= Sim tick Payload send at router 0, 0 port local to port east with 12 flits left
Payload send at router 1, 0 port west to port east with 15 flits left
Header send at router 2, 0 port west to port local
31<= Sim tick Payload send at router 0, 0 port local to port east with 11 flits left
Payload send at router 2, 0 port west to port local with 19 flits left
32<= Sim tick Payload send at router 0, 0 port local to port east with 10 flits left
Payload send at router 2, 0 port west to port local with 18 flits left
33<= Sim tick Payload send at router 0, 0 port local to port east with 9 flits left
Payload send at router 1, 0 port west to port east with 14 flits left
Payload send at router 2, 0 port west to port local with 17 flits left
34<= Sim tick Payload send at router 1, 0 port west to port east with 13 flits left
Payload send at router 2, 0 port west to port local with 16 flits left
35<= Sim tick Payload send at router 1, 0 port west to port east with 12 flits left
Payload send at router 2, 0 port west to port local with 15 flits left
36<= Sim tick Payload send at router 0, 0 port local to port east with 8 flits left
Payload send at router 1, 0 port west to port east with 11 flits left
Payload send at router 2, 0 port west to port local with 14 flits left
37<= Sim tick Payload send at router 0, 0 port local to port east with 7 flits left
Payload send at router 1, 0 port west to port east with 10 flits left
Payload send at router 2, 0 port west to port local with 13 flits left
38<= Sim tick Payload send at router 0, 0 port local to port east with 6 flits left
Payload send at router 1, 0 port west to port east with 9 flits left

```
                Payload send at router  2, 0 port west to port local with   12 flits left
39<= Sim tick   Payload send at router  0, 0 port local to port east with    5 flits left
                Payload send at router  1, 0 port west to port east with    8 flits left
                Payload send at router  2, 0 port west to port local with   11 flits left
40<= Sim tick   Payload send at router  0, 0 port local to port east with    4 flits left
                Payload send at router  1, 0 port west to port east with    7 flits left
                Payload send at router  2, 0 port west to port local with   10 flits left
41<= Sim tick   Payload send at router  0, 0 port local to port east with    3 flits left
                Payload send at router  1, 0 port west to port east with    6 flits left
                Payload send at router  2, 0 port west to port local with    9 flits left
42<= Sim tick   Payload send at router  0, 0 port local to port east with    2 flits left
                Payload send at router  1, 0 port west to port east with    5 flits left
                Payload send at router  2, 0 port west to port local with    8 flits left
43<= Sim tick   Connection closed at router  0, 0 port local to port east
                Payload send at router  1, 0 port west to port east with    4 flits left
                Payload send at router  2, 0 port west to port local with    7 flits left
44<= Sim tick   Payload send at router  1, 0 port west to port east with    3 flits left
                Payload send at router  2, 0 port west to port local with    6 flits left
45<= Sim tick   Payload send at router  1, 0 port west to port east with    2 flits left
                Payload send at router  2, 0 port west to port local with    5 flits left
46<= Sim tick   Connection closed at router  1, 0 port west to port east
                Payload send at router  2, 0 port west to port local with    4 flits left
47<= Sim tick   Payload send at router  2, 0 port west to port local with    3 flits left
48<= Sim tick   Payload send at router  2, 0 port west to port local with    2 flits left
49<= Sim tick   Connection closed at router  2, 0 port west to port local
50<= Sim tick
51<= Sim tick   Packet received at 2,0 from 0,0 with priority 1 with packet size 20
```

R3

The following log files show the functionality of the routers under a HOL blocking scenario when packets with priorities 1, 4, 9, and 3 are send from routers (1,0), (1,1), (1,2) and (2,2) respectively to router (1,3) as in the following figure (which is a simplified version of the traffic scenario seen earlier in Figure 2.4).



With PFT, it can be seen that the arbitration request priority of packet 4 gets updated to 1 from 4 at router (1,2) enabling packet 4 to get arbitration ahead of packet 3. It can also be seen that as priority tunnelling is done to the south port of router (1,2), packet 3 will be withheld from getting arbitrated until packet 1 would get transmitted completely.

```
11<= Sim tick   Packet injected at 1,0 to 1,3 with priority 1 with packet size 10
                Packet injected at 1,1 to 1,3 with priority 4 with packet size 10
                Packet injected at 1,2 to 1,3 with priority 9 with packet size 20
                Packet injected at 2,2 to 1,3 with priority 3 with packet size 10
12<= Sim tick
13<= Sim tick   Router  1, 0   port local arbitration request to south
                Router  1, 1   port local arbitration request to south
                Router  1, 2   port local arbitration request to south
                Router  2, 2   port local arbitration request to west
14<= Sim tick   Arbitration provided at router  1, 0 from local port to south port
                Arbitration provided at router  1, 1 from local port to south port
                Arbitration provided at router  1, 2 from local port to south port
                Arbitration provided at router  2, 2 from local port to west port
15<= Sim tick   Flit send from router  1, 0's local port to south port with   9 flits left
                Flit send from router  1, 1's local port to south port with   9 flits left
                Flit send from router  1, 2's local port to south port with  19 flits left
                Flit send from router  2, 2's local port to west port with   9 flits left
16<= Sim tick   Flit send from router  1, 0's local port to south port with   8 flits left
                Flit send from router  1, 1's local port to south port with   8 flits left
                Flit send from router  1, 2's local port to south port with  18 flits left
                Flit send from router  2, 2's local port to west port with   8 flits left
17<= Sim tick   Flit send from router  1, 0's local port to south port with   7 flits left
                Router  1, 1   port north arbitration request to south
                Flit send from router  1, 1's local port to south port with   7 flits left
                Router  1, 2   port north arbitration request to south
                Flit send from router  1, 2's local port to south port with  17 flits left
                Router  1, 3   port north arbitration request to local
                Flit send from router  2, 2's local port to west port with   7 flits left
18<= Sim tick   Flit send from router  1, 0's local port to south port with   6 flits left
                Flit send from router  1, 1's local port to south port with   6 flits left
                Router  1, 2   port east arbitration request to south
                Flit send from router  1, 2's local port to south port with  16 flits left
                Arbitration provided at router  1, 3 from north port to local port
                Flit send from router  2, 2's local port to west port with   6 flits left
19<= Sim tick   Flit send from router  1, 3's north port to local port with  19 flits left
Alpha registers loaded with local blocking information
20<= Sim tick   > Router  1, 1 => For packet   1 Alpha register North loaded
                > Router  1, 2 => For packet   4 Alpha register North loaded
                > Router  1, 2 => For packet   3 Alpha register East loaded
                Flit send from router  1, 3's north port to local port with  18 flits left
21<= Sim tick   Flit send from router  1, 2's local port to south port with  15 flits left
                Flit send from router  1, 3's north port to local port with  17 flits left
22<= Sim tick   Flit send from router  1, 2's local port to south port with  14 flits left
                Flit send from router  1, 3's north port to local port with  16 flits left
23<= Sim tick   Flit send from router  1, 2's local port to south port with  13 flits left
                Flit send from router  1, 3's north port to local port with  15 flits left
24<= Sim tick   Flit send from router  1, 2's local port to south port with  12 flits left
PFT flits send from Router 1,2 gets received at router 1,3 and is loaded into its North Beta register
                > Router  1, 3 => Beta register North loaded
                Flit send from router  1, 3's north port to local port with  14 flits left
PFT flits send from Router 1,1 gets received at router 1,2 and is loaded into its North Beta register
25<= Sim tick   > Router  1, 2 => Beta register North loaded
                Flit send from router  1, 2's local port to south port with  11 flits left
                Flit send from router  1, 3's north port to local port with  13 flits left
26<= Sim tick   Flit send from router  1, 2's local port to south port with  10 flits left
                Flit send from router  1, 3's north port to local port with  12 flits left
Priority forwarding initiated at router 1,2 as the priority of arbitration request at the north port is updated to 1 from 4
27<= Sim tick   > Priority forwarded from   4 to   1 at router  1, 2
Priority tunnelling initiated at the south port of router 1,2
                > Priority tunnelling for router  1, 2 at south port with priority  1
                > PFT information forwarded towards south from router  1, 2
                Flit send from router  1, 2's local port to south port with   9 flits left
                > Router  1, 3 => Beta register North loaded
                Flit send from router  1, 3's north port to local port with  11 flits left
28<= Sim tick   Flit send from router  1, 2's local port to south port with   8 flits left
                Flit send from router  1, 3's north port to local port with  10 flits left
```

29<= Sim tick Flit send from router 1, 2's local port to south port with 7 flits left
Flit send from router 1, 3's north port to local port with 9 flits left
30<= Sim tick Flit send from router 1, 2's local port to south port with 6 flits left
Flit send from router 1, 3's north port to local port with 8 flits left
31<= Sim tick Flit send from router 1, 2's local port to south port with 5 flits left
Flit send from router 1, 3's north port to local port with 7 flits left
32<= Sim tick Flit send from router 1, 2's local port to south port with 4 flits left
Flit send from router 1, 3's north port to local port with 6 flits left
33<= Sim tick Flit send from router 1, 2's local port to south port with 3 flits left
Flit send from router 1, 3's north port to local port with 5 flits left
34<= Sim tick Flit send from router 1, 2's local port to south port with 2 flits left
Flit send from router 1, 3's north port to local port with 4 flits left
35<= Sim tick Flit send from router 1, 2's local port to south port with 1 flits left
Flit send from router 1, 3's north port to local port with 3 flits left
36<= Sim tick Flit send from router 1, 2's local port to south port with 0 flits left
Flit send from router 1, 3's north port to local port with 2 flits left

Packet 4 gets arbitration ahead of packet 3 due to priority forwarding

37<= Sim tick **Arbitration provided at router 1, 2 from north port to south port**
Flit send from router 1, 3's north port to local port with 1 flits left
38<= Sim tick Flit send from router 1, 2's north port to south port with 9 flits left
Flit send from router 1, 3's north port to local port with 0 flits left
39<= Sim tick Flit send from router 1, 2's north port to south port with 8 flits left
40<= Sim tick **Packet received at 1,3 from 1,2 with priority 9 with packet size 20**
Flit send from router 1, 1's local port to south port with 5 flits left
Flit send from router 1, 2's north port to south port with 7 flits left
Router 1, 3 port north arbitration request to local
41<= Sim tick Flit send from router 1, 1's local port to south port with 4 flits left
Flit send from router 1, 2's north port to south port with 6 flits left
Arbitration provided at router 1, 3 from north port to local port
42<= Sim tick Flit send from router 1, 1's local port to south port with 3 flits left
Flit send from router 1, 3's north port to local port with 9 flits left
43<= Sim tick Flit send from router 1, 1's local port to south port with 2 flits left
Flit send from router 1, 3's north port to local port with 8 flits left
44<= Sim tick Flit send from router 1, 2's north port to south port with 5 flits left
Flit send from router 1, 3's north port to local port with 7 flits left
45<= Sim tick Flit send from router 1, 2's north port to south port with 4 flits left
Flit send from router 1, 3's north port to local port with 6 flits left
46<= Sim tick Flit send from router 1, 1's local port to south port with 1 flits left
Flit send from router 1, 2's north port to south port with 3 flits left
Flit send from router 1, 3's north port to local port with 5 flits left
47<= Sim tick Flit send from router 1, 1's local port to south port with 0 flits left
Flit send from router 1, 2's north port to south port with 2 flits left
Flit send from router 1, 3's north port to local port with 4 flits left
48<= Sim tick **Arbitration provided at router 1, 1 from north port to south port**
Flit send from router 1, 2's north port to south port with 1 flits left
Flit send from router 1, 3's north port to local port with 3 flits left
49<= Sim tick Flit send from router 1, 1's north port to south port with 9 flits left
Flit send from router 1, 2's north port to south port with 0 flits left
Flit send from router 1, 3's north port to local port with 2 flits left
50<= Sim tick Flit send from router 1, 1's north port to south port with 8 flits left
Flit send from router 1, 3's north port to local port with 1 flits left
51<= Sim tick Flit send from router 1, 0's local port to south port with 5 flits left
Flit send from router 1, 1's north port to south port with 7 flits left

Packet 1 gets arbitration ahead of packet 3 despite being a few simulation ticks delayed as south port was tunnelled preventing packet 3 from getting arbitration

Router 1, 2 port north arbitration request to south
Flit send from router 1, 3's north port to local port with 0 flits left
52<= Sim tick Flit send from router 1, 0's local port to south port with 4 flits left
Flit send from router 1, 1's north port to south port with 6 flits left
Arbitration provided at router 1, 2 from router 2 to south port

Packet 4 received at router 1,3 ahead of packet 3 due to priority forwarding

53<= Sim tick **Packet received at 1,3 from 1,1 with priority 4 with packet size 10**
Flit send from router 1, 0's local port to south port with 3 flits left
Flit send from router 1, 2's north port to south port with 9 flits left
54<= Sim tick Flit send from router 1, 0's local port to south port with 2 flits left
Flit send from router 1, 2's north port to south port with 8 flits left
55<= Sim tick Flit send from router 1, 1's north port to south port with 5 flits left
Flit send from router 1, 2's north port to south port with 7 flits left
Router 1, 3 port north arbitration request to local
56<= Sim tick Flit send from router 1, 1's north port to south port with 4 flits left
Flit send from router 1, 2's north port to south port with 6 flits left
Arbitration provided at router 1, 3 from north port to local port
57<= Sim tick Flit send from router 1, 0's local port to south port with 1 flits left
Flit send from router 1, 1's north port to south port with 3 flits left
Flit send from router 1, 3's north port to local port with 9 flits left
58<= Sim tick Flit send from router 1, 0's local port to south port with 0 flits left

```
                Flit send from router  1, 1's north port to south port with    2 flits left
                Flit send from router  1, 3's north port to local port with    8 flits left
59<= Sim tick   Flit send from router  1, 2's north port to south port with    5 flits left
                Flit send from router  1, 3's north port to local port with    7 flits left
60<= Sim tick   Flit send from router  1, 2's north port to south port with    4 flits left
                Flit send from router  1, 3's north port to local port with    6 flits left
61<= Sim tick   Flit send from router  1, 1's north port to south port with    1 flits left
                Flit send from router  1, 2's north port to south port with    3 flits left
                Flit send from router  1, 3's north port to local port with    5 flits left
62<= Sim tick   Flit send from router  1, 1's north port to south port with    0 flits left
                Flit send from router  1, 2's north port to south port with    2 flits left
                Flit send from router  1, 3's north port to local port with    4 flits left
63<= Sim tick   Flit send from router  1, 2's north port to south port with    1 flits left
                Flit send from router  1, 3's north port to local port with    3 flits left
64<= Sim tick   Flit send from router  1, 2's north port to south port with    0 flits left
                Flit send from router  1, 3's north port to local port with    2 flits left
65<= Sim tick   Arbitration provided at router  1, 2 from east port to south port
                Flit send from router  1, 3's north port to local port with    1 flits left
66<= Sim tick   Flit send from router  1, 2's east port to south port with    9 flits left
                Flit send from router  1, 3's north port to local port with    0 flits left
67<= Sim tick   Flit send from router  1, 2's east port to south port with    8 flits left
Packet 1 received at router 1,3 ahead of packet 3 as a result of priority forwarding and priority tunnelling.
68<= Sim tick   Packet received at 1,3 from 1,0 with priority 1 with packet size 10
                Flit send from router  1, 2's east port to south port with    7 flits left
                Router  1, 3   port north arbitration request to local
                Flit send from router  2, 2's local port to west port with    5 flits left
69<= Sim tick   Flit send from router  1, 2's east port to south port with    6 flits left
                Arbitration provided at router  1, 3 from router 2 to local port
                Flit send from router  2, 2's local port to west port with    4 flits left
70<= Sim tick   Flit send from router  1, 3's north port to local port with    9 flits left
                Flit send from router  2, 2's local port to west port with    3 flits left
71<= Sim tick   Flit send from router  1, 3's north port to local port with    8 flits left
                Flit send from router  2, 2's local port to west port with    2 flits left
72<= Sim tick   Flit send from router  1, 2's east port to south port with    5 flits left
                Flit send from router  1, 3's north port to local port with    7 flits left
73<= Sim tick   Flit send from router  1, 2's east port to south port with    4 flits left
                Flit send from router  1, 3's north port to local port with    6 flits left
74<= Sim tick   Flit send from router  1, 2's east port to south port with    3 flits left
                Flit send from router  1, 3's north port to local port with    5 flits left
                Flit send from router  2, 2's local port to west port with    1 flits left
75<= Sim tick   Flit send from router  1, 2's east port to south port with    2 flits left
                Flit send from router  1, 3's north port to local port with    4 flits left
                Flit send from router  2, 2's local port to west port with    0 flits left
76<= Sim tick   Flit send from router  1, 2's east port to south port with    1 flits left
                Flit send from router  1, 3's north port to local port with    3 flits left
77<= Sim tick   Flit send from router  1, 2's east port to south port with    0 flits left
                Flit send from router  1, 3's north port to local port with    2 flits left
78<= Sim tick   Flit send from router  1, 3's north port to local port with    1 flits left
79<= Sim tick   Flit send from router  1, 3's north port to local port with    0 flits left
80<= Sim tick
81<= Sim tick   Packet received at 1,3 from 2,2 with priority 3 with packet size 10
```

R7-F/R7-FD

The following is the log file of the functionality of the routers when routers (0,0) and (1,0) sends a 40 flit wide packet to router (0,1). The packet from router (0,0) is transmitted first with priority 3followed by the packet from router (1,0) five clock cycles later with priority 1 and slack value 6. As a result, it can be seen that as packet 1 waits for arbitration, its slack value gets decremented several times until it gets to a point where it gets an instantaneous priority greater than packet 3 hence initiating packet splitting. As a result, packet 1 can be seen getting complete transmission before packet 3.

```
11<= Sim tick   Packet injected at 1,0 to 0,1 with priority 3 with packet size 40
12<= Sim tick
13<= Sim tick
14<= Sim tick   Router  1, 0   port local arbitration request to west
15<= Sim tick   Normal arbitration at router  1, 0 to port east
16<= Sim tick   Packet injected at 0,0 to 0,1 with priority 1 with packet size 40
                Header send at router  1, 0 port local to port west
17<= Sim tick   Payload send at router  1, 0 port local to port west with   39 flits left
18<= Sim tick   Payload send at router  1, 0 port local to port west with   38 flits left
```

19<= Sim tick Router 0, 0 port local arbitration request to south
Router 0, 0 port east arbitration request to south
Payload send at router 1, 0 port local to port west with 37 flits left

As the packet from router (0,0) waits for arbitration, its slack can be seen getting reduced

20<= Sim tick **> Dynamic slack update- Router 0 0 port local slack: 5 inst_prio: 6 packet priority 1**
Payload send at router 1, 0 port local to port west with 36 flits left
21<= Sim tick Payload send at router 1, 0 port local to port west with 35 flits left
22<= Sim tick Normal arbitration at router 0, 0 to port east
23<= Sim tick Header send at router 0, 0 port east to port south
24<= Sim tick Payload send at router 0, 0 port east to port south with 39 flits left

Slack being reduced further

25<= Sim tick **> Dynamic slack update- Router 0 0 port local slack: 4 inst_prio: 5 packet priority 1**
Payload send at router 0, 0 port east to port south with 38 flits left
26<= Sim tick Payload send at router 0, 0 port east to port south with 37 flits left
Router 0, 1 port north arbitration request to local
Payload send at router 1, 0 port local to port west with 34 flits left
27<= Sim tick Payload send at router 0, 0 port east to port south with 36 flits left
Payload send at router 1, 0 port local to port west with 33 flits left
28<= Sim tick Payload send at router 0, 0 port east to port south with 35 flits left
Payload send at router 1, 0 port local to port west with 32 flits left
29<= Sim tick Payload send at router 1, 0 port local to port west with 31 flits left
30<= Sim tick **> Dynamic slack update- Router 0 0 port local slack: 3 inst_prio: 4 packet priority 1**
Payload send at router 1, 0 port local to port west with 30 flits left
31<= Sim tick Normal arbitration at router 0, 1 to port east
Payload send at router 1, 0 port local to port west with 29 flits left
32<= Sim tick Header send at router 0, 1 port north to port local
33<= Sim tick Payload send at router 0, 1 port north to port local with 39 flits left
34<= Sim tick Payload send at router 0, 1 port north to port local with 38 flits left
35<= Sim tick **> Dynamic slack update- Router 0 0 port local slack: 2 inst_prio: 3 packet priority 1**
Payload send at router 0, 0 port east to port south with 34 flits left
Payload send at router 0, 1 port north to port local with 37 flits left
36<= Sim tick Payload send at router 0, 0 port east to port south with 33 flits left
Payload send at router 0, 1 port north to port local with 36 flits left
37<= Sim tick Payload send at router 0, 0 port east to port south with 32 flits left
Payload send at router 0, 1 port north to port local with 35 flits left
38<= Sim tick Payload send at router 0, 0 port east to port south with 31 flits left
Payload send at router 0, 1 port north to port local with 34 flits left
Payload send at router 1, 0 port local to port west with 28 flits left
39<= Sim tick Payload send at router 0, 0 port east to port south with 30 flits left
Payload send at router 0, 1 port north to port local with 33 flits left
Payload send at router 1, 0 port local to port west with 27 flits left
40<= Sim tick **> Dynamic slack update- Router 0 0 port local slack: 1 inst_prio: 2 packet priority 1**
Payload send at router 0, 0 port east to port south with 29 flits left
Payload send at router 0, 1 port north to port local with 32 flits left
Payload send at router 1, 0 port local to port west with 26 flits left
41<= Sim tick Payload send at router 0, 0 port east to port south with 28 flits left
Payload send at router 0, 1 port north to port local with 31 flits left
Payload send at router 1, 0 port local to port west with 25 flits left
42<= Sim tick Payload send at router 0, 0 port east to port south with 27 flits left
Payload send at router 0, 1 port north to port local with 30 flits left
Payload send at router 1, 0 port local to port west with 24 flits left
43<= Sim tick Payload send at router 0, 0 port east to port south with 26 flits left
Payload send at router 0, 1 port north to port local with 29 flits left
Payload send at router 1, 0 port local to port west with 23 flits left
44<= Sim tick Payload send at router 0, 0 port east to port south with 25 flits left
Payload send at router 0, 1 port north to port local with 28 flits left
Payload send at router 1, 0 port local to port west with 22 flits left

Packet splitting request initiated as the packet from router (0,0) achieves a higher dynamic priority than the packet from router(1,0)

45<= Sim tick **Packet split request at router 0, 0 from port local to split port east**
Payload send at router 0, 0 port east to port south with 24 flits left
Payload send at router 0, 1 port north to port local with 27 flits left
Payload send at router 1, 0 port local to port west with 21 flits left

Packet splitting initiated

46<= Sim tick **Payload split at router 0, 0 port east to port south with 23 flits left**
Payload send at router 0, 1 port north to port local with 26 flits left
Payload send at router 1, 0 port local to port west with 20 flits left
47<= Sim tick Payload send at router 0, 1 port north to port local with 25 flits left
Payload send at router 1, 0 port local to port west with 19 flits left
48<= Sim tick Payload send at router 0, 1 port north to port local with 24 flits left
Payload send at router 1, 0 port local to port west with 18 flits left

Connection closed for the split packet

49<= Sim tick **Connection closed at router 0, 1 port north to port local**
Payload send at router 1, 0 port local to port west with 17 flits left
50<= Sim tick Normal arbitration at router 0, 0 to port north

Transmission of the packet from router (0,0) initiated post splitting of the other packet

51<= Sim tick **Header send at router 0, 0 port local to port south**
52<= Sim tick Payload send at router 0, 0 port local to port south with 39 flits left
53<= Sim tick Payload send at router 0, 0 port local to port south with 38 flits left
54<= Sim tick Payload send at router 0, 0 port local to port south with 37 flits left
Router 0, 1 port north arbitration request to local55<= Sim tick
Payload send at router 0, 0 port local to port south with 36 flits left
56<= Sim tick Payload send at router 0, 0 port local to port south with 35 flits left
Normal arbitration at router 0, 1 to port local
57<= Sim tick **Header send at router 0, 1 port north to port local**
58<= Sim tick Payload send at router 0, 1 port north to port local with 39 flits left
59<= Sim tick Payload send at router 0, 1 port north to port local with 38 flits left
60<= Sim tick Payload send at router 0, 0 port local to port south with 34 flits left
Payload send at router 0, 1 port north to port local with 37 flits left
61<= Sim tick Payload send at router 0, 0 port local to port south with 33 flits left
Payload send at router 0, 1 port north to port local with 36 flits left
62<= Sim tick Payload send at router 0, 0 port local to port south with 32 flits left
Payload send at router 0, 1 port north to port local with 35 flits left
63<= Sim tick Payload send at router 0, 0 port local to port south with 31 flits left
Payload send at router 0, 1 port north to port local with 34 flits left
64<= Sim tick Payload send at router 0, 0 port local to port south with 30 flits left
Payload send at router 0, 1 port north to port local with 33 flits left
65<= Sim tick Payload send at router 0, 0 port local to port south with 29 flits left
Payload send at router 0, 1 port north to port local with 32 flits left
66<= Sim tick Payload send at router 0, 0 port local to port south with 28 flits left
Payload send at router 0, 1 port north to port local with 31 flits left
67<= Sim tick Payload send at router 0, 0 port local to port south with 27 flits left
Payload send at router 0, 1 port north to port local with 30 flits left
68<= Sim tick Payload send at router 0, 0 port local to port south with 26 flits left
Payload send at router 0, 1 port north to port local with 29 flits left
69<= Sim tick Payload send at router 0, 0 port local to port south with 25 flits left
Payload send at router 0, 1 port north to port local with 28 flits left
70<= Sim tick Payload send at router 0, 0 port local to port south with 24 flits left
Payload send at router 0, 1 port north to port local with 27 flits left
71<= Sim tick Payload send at router 0, 0 port local to port south with 23 flits left
Payload send at router 0, 1 port north to port local with 26 flits left
72<= Sim tick Payload send at router 0, 0 port local to port south with 22 flits left
Payload send at router 0, 1 port north to port local with 25 flits left
73<= Sim tick Payload send at router 0, 0 port local to port south with 21 flits left
Payload send at router 0, 1 port north to port local with 24 flits left
74<= Sim tick Payload send at router 0, 0 port local to port south with 20 flits left
Payload send at router 0, 1 port north to port local with 23 flits left
75<= Sim tick Payload send at router 0, 0 port local to port south with 19 flits left
Payload send at router 0, 1 port north to port local with 22 flits left
76<= Sim tick Payload send at router 0, 0 port local to port south with 18 flits left
Payload send at router 0, 1 port north to port local with 21 flits left
77<= Sim tick Payload send at router 0, 0 port local to port south with 17 flits left
Payload send at router 0, 1 port north to port local with 20 flits left
78<= Sim tick Payload send at router 0, 0 port local to port south with 16 flits left
Payload send at router 0, 1 port north to port local with 19 flits left
79<= Sim tick Payload send at router 1, 0 port local to port west with 14 flits left
105<= Sim tick Payload send at router 1, 0 port local to port west with 13 flits left
106<= Sim tick Normal arbitration at router 0, 1 to port east
Payload send at router 1, 0 port local to port west with 12 flits left
107<= Sim tick **Header send at router 0, 1 port north to port local**
108<= Sim tick Payload send at router 0, 1 port north to port local with 23 flits left
109<= Sim tick Payload send at router 0, 1 port north to port local with 22 flits left
110<= Sim tick Payload send at router 0, 0 port east to port south with 18 flits left
Payload send at router 0, 1 port north to port local with 21 flits left
111<= Sim tick Payload send at router 0, 0 port east to port south with 17 flits left
Payload send at router 0, 1 port north to port local with 20 flits left
112<= Sim tick Payload send at router 0, 0 port east to port south with 16 flits left
Payload send at router 0, 1 port north to port local with 19 flits left
113<= Sim tick Payload send at router 0, 0 port east to port south with 15 flits left
Payload send at router 0, 1 port north to port local with 18 flits left
Payload send at router 1, 0 port local to port west with 11 flits left
114<= Sim tick Payload send at router 0, 0 port east to port south with 14 flits left
Payload send at router 0, 1 port north to port local with 17 flits left
Payload send at router 1, 0 port local to port west with 10 flits left
115<= Sim tick Payload send at router 0, 0 port east to port south with 13 flits left
Payload send at router 0, 1 port north to port local with 16 flits left
Payload send at router 1, 0 port local to port west with 9 flits left
116<= Sim tick Payload send at router 0, 0 port east to port south with 12 flits left
Payload send at router 0, 1 port north to port local with 15 flits left
Payload send at router 1, 0 port local to port west with 8 flits left
117<= Sim tick Payload send at router 0, 0 port east to port south with 11 flits left
Payload send at router 0, 1 port north to port local with 14 flits left

Payload send at router 1, 0 port local to port west with 7 flits left
118<= Sim tick Payload send at router 0, 0 port east to port south with 10 flits left
Payload send at router 0, 1 port north to port local with 13 flits left
Payload send at router 1, 0 port local to port west with 6 flits left
119<= Sim tick Payload send at router 0, 0 port east to port south with 9 flits left
Payload send at router 0, 1 port north to port local with 12 flits left
Payload send at router 1, 0 port local to port west with 5 flits left
120<= Sim tick Payload send at router 0, 0 port east to port south with 8 flits left
Payload send at router 0, 1 port north to port local with 11 flits left
Payload send at router 1, 0 port local to port west with 4 flits left
121<= Sim tick Payload send at router 0, 0 port east to port south with 7 flits left
Payload send at router 0, 1 port north to port local with 10 flits left
Payload send at router 1, 0 port local to port west with 3 flits left
122<= Sim tick Payload send at router 0, 0 port east to port south with 6 flits left
Payload send at router 0, 1 port north to port local with 9 flits left
Payload send at router 1, 0 port local to port west with 2 flits left
123<= Sim tick Payload send at router 0, 0 port east to port south with 5 flits left
Payload send at router 0, 1 port north to port local with 8 flits left
Connection closed at router 1, 0 port local to port west
124<= Sim tick Payload send at router 0, 0 port east to port south with 4 flits left
Payload send at router 0, 1 port north to port local with 7 flits left
125<= Sim tick Payload send at router 0, 0 port east to port south with 3 flits left
Payload send at router 0, 0 port local to port south with 15 flits left
Payload send at router 0, 1 port north to port local with 18 flits left
80<= Sim tick Payload send at router 0, 0 port local to port south with 14 flits left
Payload send at router 0, 1 port north to port local with 17 flits left
81<= Sim tick Payload send at router 0, 0 port local to port south with 13 flits left
Payload send at router 0, 1 port north to port local with 16 flits left
82<= Sim tick Payload send at router 0, 0 port local to port south with 12 flits left
Payload send at router 0, 1 port north to port local with 15 flits left
83<= Sim tick Payload send at router 0, 0 port local to port south with 11 flits left
Payload send at router 0, 1 port north to port local with 14 flits left
84<= Sim tick Payload send at router 0, 0 port local to port south with 10 flits left
Payload send at router 0, 1 port north to port local with 13 flits left
85<= Sim tick Payload send at router 0, 0 port local to port south with 9 flits left
Payload send at router 0, 1 port north to port local with 12 flits left
86<= Sim tick Payload send at router 0, 0 port local to port south with 8 flits left
Payload send at router 0, 1 port north to port local with 11 flits left
87<= Sim tick Payload send at router 0, 0 port local to port south with 7 flits left
Payload send at router 0, 1 port north to port local with 10 flits left
88<= Sim tick Payload send at router 0, 0 port local to port south with 6 flits left
Payload send at router 0, 1 port north to port local with 9 flits left
89<= Sim tick Payload send at router 0, 0 port local to port south with 5 flits left
Payload send at router 0, 1 port north to port local with 8 flits left
90<= Sim tick Payload send at router 0, 0 port local to port south with 4 flits left
Payload send at router 0, 1 port north to port local with 7 flits left
91<= Sim tick Payload send at router 0, 0 port local to port south with 3 flits left
Payload send at router 0, 1 port north to port local with 6 flits left
92<= Sim tick Payload send at router 0, 0 port local to port south with 2 flits left
Payload send at router 0, 1 port north to port local with 5 flits left
93<= Sim tick **Connection closed at router 0, 0 port local to port south**
Payload send at router 0, 1 port north to port local with 4 flits left
94<= Sim tick Payload send at router 0, 1 port north to port local with 3 flits left
95<= Sim tick Payload send at router 0, 1 port north to port local with 2 flits left
96<= Sim tick **Connection closed at router 0, 1 port north to port local**
Arbitrating the transmission of the split packet
97<= Sim tick **Resumed arbitration at router 0, 0 to port east**
98<= Sim tick Packet received at 0,1 from 0,0 with priority 1 with packet size 40
New header formulated and send (for retransmission of the split packet)
(Resumed)Header send at router 0, 0 port east to port south
99<= Sim tick Payload send at router 0, 0 port east to port south with 23 flits left
100<= Sim tick Payload send at router 0, 0 port east to port south with 22 flits left
101<= Sim tick Payload send at router 0, 0 port east to port south with 21 flits left
Router 0, 1 port north arbitration request to local
102<= Sim tick Payload send at router 0, 0 port east to port south with 20 flits left
Payload send at router 1, 0 port local to port west with 16 flits left
103<= Sim tick Payload send at router 0, 0 port east to port south with 19 flits left
Payload send at router 1, 0 port local to port west with 15 flits left
104<= Sim tick Payload send at router 0, 1 port north to port local with 6 flits left
126<= Sim tick **Connection closed at router 0, 0 port east to port south**
Payload send at router 0, 1 port north to port local with 5 flits left
127<= Sim tick Payload send at router 0, 1 port north to port local with 4 flits left
128<= Sim tick Payload send at router 0, 1 port north to port local with 3 flits left
129<= Sim tick **Connection closed at router 0, 1 port north to port local**
130<= Sim tick
131<= Sim tick **Packet received at 0,1 from 1,0 with priority 3 with packet size 40**

Glossary of Terms

Application supplied priority	Priority of the packet added by the application that initiated the communication
ASIC	Application Specific Integrated Circuit
BE	Best Effort Service
Blocking	A communication is said to be blocked by another when the communication path needed for the former is being utilised by the later communication thus preventing its transmission
Contention	Contention is defined as the situation when two or more communication flows require transmission through the same connection link
Cumulative count of packet reception	(For each packet priority) Number of packets that were received successfully at that priority level or higher
Deadline	The desired bound on packet latency in simulation ticks
DHARA	Dynamic Slack Hard-line Aware Router Architecture
DI	Divider Index used to specify the weightage of residual

	slack in computing instantaneous priority
DMA	Dynamic Memory Access
DTMVC	Dynamic Time Multiplexed Virtual Channel
Dynamic behaviour	The ability of the router to respond in run time to incoming packets (regardless of its destination) without reconfiguration to the routing logic
Dynamic traffic	Traffic that has no bounded time interval be-tween successive packets and no upper or lower bounds on packet length
ECU	Electronic Control Unit
EPSRC	Engineering and Physical Sciences Research Council
EU FP7	European Union Framework Program Seven
EVC	Express Virtual Channel
FIFO	First In First Out (buffer)
Flit	Flow Control Digit; the basic unit of communication through NoC links

Flow control	The process of managing data transmission between two nodes
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
GS	Guaranteed Service
HDL	Hardware Description Language
Hard deadline	The latency deadline of a packet, missing which can result in a catastrophic failure of the design target
Hermes based NoC	In the thesis, the non-preemptive NoC model (with XY-routing and wormhole switching) based on Hermes (explained in section 2.3.1) is referred to as the Hermes based NoC
HOL blocking	A packet is said to be Head-of-line blocked when it is blocked by a lower priority packet which is already blocked
HYENA	HYper Criticality Enabled NoC Architecture
Interquartile range	The difference between the 3rd and 1st quartile of latency

IP	Intellectual Property
Latency	The time interval between time instant when the net-work interface of the source core is supposed to inject the header flit of the packet to the instant when the whole of the packet is received by the net-work inter-face of the destination core in clock cycles
LDM	Link Division Multiplexing
LUT	Lookup table
Many-core proces-sor	A processor with high number of cores (tens or hun-dreds)
MCS	Mixed Criticality System (system with two or more dis-tinct criticality levels)
MSB	Most Significant Bit
Network utilisation	Network utilisation is defined as the percentage of total number of connection links being used for communica-tion at any point of time
NoC	Network-on-Chip
Non-preemptive	Without pre-emptive arbitration

Packet latency	The time interval between time instant when the net-work interface of the source core is supposed to inject the header flit of the packet to the instant when the whole of the packet is received by the net-work inter-face of the destination core in clock cycles
Packet period	Time in clock cycles between successive packet injec-tion (on a specific IP)
PFS	Priority Forwarded Packet Splitting
PFS-D	Priority Forwarded Packet Splitting with DHARA
PFT	Priority Forwarding and Tunnelling
Predictability	Packet predictability enhancement is defined as the re-duction in variation in latency of the packet. So a pack-et with lower variation in latency is considered more predictable than one with higher variation
QNoC	Quality of Service NoC
QoS	Quality of Service
Residual slack	The time in clock cycles a packet can be delayed with-out missing its soft deadline
RTL	Register Transfer Level

SAF	Store and Forward (switching technique)
Scalability	The ability of the NoC router to handle packets with a wider range of priority values thus enabling the use of the router in bigger NoC topologies than it was initially designed for
Soft deadline	The latency deadline of a packet, missing which may result in performance degradation of the design target and would not cause a catastrophic failure of the design target
SPS	Selective Packet Splitting
Starvation	Blocking of packets indefinitely, resulting in packet delivery failure
Tailbacking	A packet is said to be tailbacked when the link required for its transmission is being utilised by a lower priority packet
TDM	Time Division Multiplexing
TG	Traffic Generator
TLM	Transaction Level Modelling
TR	Traffic Receptor

Traffic pattern	The pattern of traffic flow through the routers over the whole NoC over the entire simulation run (Traffic pattern consists of all the packet flows through the NoC, each specifying parameters like source-destination information, packet priority, injection time and packet size)
URL	Universal Resource Locator
VB	Visual Basic (Microsoft developed programming language)
VC	Virtual Channel
VC based NoC	In the thesis, the NoC with preemptive arbitration enabled by Virtual Channels is referred to as the VC based NoC
VCT	Virtual Cut Through (switching technique)

References

[1] "International Technology Roadmap for Semiconductors- Interconnects," *IRTS*. [Online]. Available: http://www.itrs.net/Links/2005itrs/Interconnect2005.pdf [Accessed: 01/06/2015].

[2] S. Kumar, A. Jantsch, J.-P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, and A. Hemani, "A network on chip architecture and design methodology," in *2002 IEEE Computer Society Annual Symposium on VLSI*, pp. 105–112.

[3] L. Benini and G. De Micheli, "Powering Networks on Chips: Energy-efficient and Reliable Interconnect Design for SoCs," in *2001 14th International Symposium on Systems Synthesis*, New York, USA, pp. 33–38.

[4] P. Guerrier and A. Greiner, "A generic architecture for on-chip packet-switched interconnections," in *2000 Design, Automation and Test in Europe Conference and Exhibition (DATE 2000)*, pp. 250–256.

[5] I. Cidon, "Zooming in on Network-on-Chip Architectures," in *Structural Information and Communication Complexity*, S. Kutten and J. Žerovnik, Eds. Springer Berlin Heidelberg, 2010, pp. 1–1.

[6] A. Agarwal, C. Iskander, and R. Shankar, "Survey of Network on Chip (NoC) Architectures & Contributions," *Journal of Engineering, Computing and Architecture*, vol. 3, 2009.

[7] I. Cidon and I. Keidar, "Zooming in on Network-on-Chip Architectures ∗," presented at the Technion Department of Electrical Engineering, Tech. Rep, 2005.

[8] R. Havemann and J. A. Hutchby, "High-Performance Interconnects: An Integration Overview," *Proceedings of the IEEE, vol. 89, no. 5, may 2001*.

[9] "A comparison of Network-on-Chip and Busses," *Arteris, The Network-On-Chip Company*. [Online]. Available: http://www.design-reuse.com/articles/10496/a-comparison-of-network-on-chip-and-busses.html. [Accessed: 28-May-2012].

[10] N. Magen, A. Kolodny, U. Weiser, and N. Shamir, "Interconnect-power Dissipation in a Microprocessor," in *2004 International Workshop on System Level Interconnect Prediction*, New York, USA, pp. 7–13.

[11] M. Ali, M. Welzl, and M. Zwicknagl, "Networks on Chips: Scalable interconnects for future systems on chips," in *2008 4th European Conference on Circuits and Systems for Communications (ECCSC 2008)*, pp. 240 –245.

[12] E. Rijpkema, K. Goossens, and P. Wielage, “A Router Architecture for Networks on Silicon,” in *2001 2nd workshop on embedded systems*, pp. 181–188.

[13] C. Kim, D. Burger, and S. W. Keckler, “An Adaptive, Non-uniform Cache Structure for Wire-delay Dominated On-chip Caches,” in *2002 10th International Conference on Architectural Support for Programming Languages and Operating Systems*, New York, USA, pp. 211–222.

[14] K. Mai, T. Paaske, N. Jayasena, R. Ho, W. J. Dally, and M. Horowitz, “Smart Memories: a modular reconfigurable architecture,” in *2000 27th International Symposium on Computer Architecture*, pp. 161–171.

[15] F. Moraes, N. Calazans, A. Mello, L. Möller, and L. Ost, “HERMES: an infrastructure for low area overhead packet-switching networks on chip,” *Integration, the VLSI Journal - Elsevier*, vol. 38, no. 1, pp. 69–93, Oct. 2004.

[16] Z. Shi and A. Burns, “Schedulability Analysis and Task Mapping for Real-time On-chip Communication,” *Real-Time Systems-ACM Digital Library*, vol. 46, no. 3, pp. 360–385, Dec. 2010.

[17] K. Goossens, J. van Meerbergen, A. Peeters, and P. Wielage, “Networks on silicon: combining best-effort and guaranteed services,” in *2002 Design, Automation and Test in Europe Conference and Exhibition (DATE 2002)*, pp. 423–425.

[18] A. Ziviani, B. E Wolfinger, J. De Rezende, and S. Fdida, “Joint Adoption of QoS Schemes for MPEG Streams,” *Multimedia Tools and Applications 2005 Springer Science + Business Media, Inc.*

[19] A. Ziviani, B. E. Wolfinger, J. F. de Rezende, O. C. M. B. Duarte, and S. Fdida, *On the Combined Adoption of QoS Schemes to Improve the Delivery Quality of MPEG Video Streams*. International Symposium on Performance Evaluation of Computer and Telecommunications Systems (SPECTS 2002).

[20] R. Stefan, A. Molnos, A. Ambrose, and K. Goossens, “A TDM NoC supporting QoS, multicast, and fast connection set-up,” in *2012 Design, Automation Test in Europe Conference Exhibition (DATE 2012)*, pp. 1283 –1288.

[21] A. Morgenshtein, A. Kolodny, and R. Ginosar, “Link Division Multiplexing (LDM) for Network-on-Chip Links,” in *2006 IEEE 24th Convention of Electrical and Electronics Engineers in Israel*, pp. 245 –249.

[22] T.-C. Huang, U. Y. Ogras, and R. Marculescu, “Virtual Channels Planning for Networks-on-Chip,” in *2007 8th International Symposium on Quality Electronic Design (ISQED 2007)*, pp. 879–884.

[23] L. Thiele and R. Wilhelm, “Design for Timing Predictability,” *Real-Time Systems*, vol. 28, no. 2–3, pp. 157–177, Nov. 2004.

[24] A. Mello, L. Tedesco, N. Calazans, and F. Moraes, “Virtual channels in networks on chip: implementation and evaluation on hermes NoC,” in *Proceedings of the*

18th annual symposium on Integrated circuits and system design, New York,USA, 2005, pp. 178–183.

[25] B. Sudev and L. S. Indrusiak, "PFT- A low overhead predictability enhancement technique for non-preemptive NoCs," in *2013 IFIP/IEEE 21st International Conference on Very Large Scale Integration (VLSI-SoC 2013)*, pp. 314–317.

[26] B. Sudev and L. S. Indrusiak, "Low overhead predictability enhancement in non-preemptive network-on-chip routers using Priority Forwarded Packet Splitting," in *2014 9th International Symposium on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC 2014)*, pp. 1–8.

[27] B. Sudev and L. S. Indrusiak, "Predictability Enhancement in Non-preemptive NoCs using Selective Packet Splitting." 2014 12th IEEE International Conference on Industrial Informatics (INDIN 2014).

[28] B. Sudev, L. Indrusiak, and J. Harbin, "Network-on-Chip Packet Prioritisation based on Instantaneous Slack Awareness," in *13th IEEE International Conference on Industrial Informatics (INDIN 2015)*, Cambridge-UK.

[29] J. Lee, C. Nicopoulos, S. J. Park, M. Swaminathan, and J. Kim, "Do we need wide flits in Networks-on-Chip?," in *2013 IEEE Computer Society Annual Symposium on VLSI (ISVLSI 2013)*, pp. 2–7.

[30] Y. A. Sadawarte, M. A. Gaikwad, and R. M. Patrikar, "Comparative study of switching techniques for network-on-chip architecture," in *2011 International Conference on Communication, Computing & Security*, New York, USA, pp. 243–246.

[31] C. Hilton and B. Nelson, "PNoC: a flexible circuit-switched NoC for FPGA-based systems," *2006 IEEE Computers and Digital Techniques*, vol. 153, no. 3, pp. 181–188.

[32] J. Liu, L.-R. Zheng, and H. Tenhunen, "A circuit-switched network architecture for network-on-chip," in *2004 IEEE International System-on-Chip Conference*, pp. 55–58.

[33] I. Nousias and A. Tughrul, "Wormhole Routing with Virtual Channels using Adaptive Rate Control for Network-on-Chip (NoC)." 2006 1st NASA/ESA Conference on Adaptive Hardware and Systems (AHS 2006).

[34] K. Goossens, J. Dielissen, and A. Radulescu, "AEthereal network on chip: concepts, architectures, and implementations," *Design Test of Computers, IEEE*, vol. 22, no. 5, pp. 414 – 421, Oct. 2005.

[35] S. Pasricha and N. Dutt, *On-Chip Communication Architectures: System on Chip Interconnect [Book]*. Morgan Kaufmann, 2008.

[36] D. Tutsch and M. Malek, "Comparison of network-on-chip topologies for multi-core systems considering multicast and local traffic," in *2009 2nd International*

Conference on Simulation Tools and Techniques, ICST, Brussels, Belgium, Belgium, pp. 23:1–23:9.

[37] M. Millberg, E. Nilsson, R. Thid, S. Kumar, and A. Jantsch, "The Nostrum backbone-a communication protocol stack for Networks on Chip," in *2004 17th International Conference on VLSI Design*, pp. 693 – 696.

[38] M. Millberg, E. Nilsson, R. Thid, and A. Jantsch, "Guaranteed bandwidth using looped containers in temporally disjoint networks within the nostrum network on chip," in *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*, 2004, vol. 2, pp. 890–895 Vol.2.

[39] A. Vieira de Mello, L. C. Ost, F. G. Moraes, and N. Calazans, "Evaluation of Routing Algorithms on Mesh Based NoCs," Faculdade de Informatica - PUCRS, Porto Alegre, Technical Report 40, May 2004.

[40] R. Mutha, "Packet Switched Networks for Communications within Large Multi-Core Systems on Chip," *International Journal of Modelling and Optimization, Vol. 2, No. 4, August 2012*.

[41] S. Liu, A. Jantsch, and Z. Lu, "Analysis and evaluation of circuit switched NoC and packet switched NoC," *2013 Euromicro Conference on Digital System Design (DSD 2013)*.

[42] C.-H. Lu, K.-C. Chiang, and P.-A. Hsiung, "Round-based priority arbitration for predictable and reconfigurable Network-on-Chip," in *2009 International Conference on Field-Programmable Technology (FPT 2009)*, pp. 403–406.

[43] S. N. Routing Abdelkader Saadaoui, "NoC: Qos metrics modelling and analysis based on dynamic routing," *International Journal of Distributed and Parallel Systems*, vol. 3, 2012.

[44] E. de F. Corrêa, L. A. de P. e Silva, F. R. Wagner, and L. Carro, "Fitting the Router Characteristics in NoCs to Meet QoS Requirements," in *Proceedings of the 20th Annual Conference on Integrated Circuits and Systems Design*, New York, USA, pp. 105–110.

[45] S. Murali, D. Atienza, P. Meloni, S. Carta, L. Benini, G. De Micheli, and L. Raffo, "Synthesis of Predictable Networks-on-Chip-Based Interconnect Architectures for Chip Multiprocessors," *2007 IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 8, pp. 869–880.

[46] R. Mraz, "Reducing the Variance of Point to Point Transfers in the IBM 9076 Parallel Computer," in *Proceedings of the 1994 ACM/IEEE Conference on Supercomputing*, Los Alamitos, CA, USA, 1994, pp. 620–629.

[47] J. V. Escamilla, J. Flich, and P. J. Garcia, "Head-of-Line Blocking Avoidance in Networks-on-Chip," in *IEEE 27th International Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW 2013)*, 2013, pp. 796–805.

[48] M. J. Karol, M. G. Hluchyj, and S. P. Morgan, “Input Versus Output Queueing on a Space-Division Packet Switch,” *IEEE Transactions on Communications*, vol. 35, no. 12, pp. 1347–1356, 1987.

[49] R. Elsevier and S. 2010, “A TDM Slot Allocation Flow Based on Multipath Routing in NoCs,” *TechRepublic*. [Online]. Available: http://www.techrepublic.com/resource-library/whitepapers/a-tdm-slot-allocation-flow-based-on-multipath-routing-in-nocs/. [Accessed: 13-Mar-2015].

[50] D. Wiklund and D. Liu, “SoCBUS: switched network on chip for hard real time embedded systems,” in *2003 International Parallel and Distributed Processing Symposium*, p. 8 pp.

[51] T. Marescaux, B. Bricke, P. Debacker, V. Nollet, and H. Corporaal, “Dynamic time-slot allocation for QoS enabled networks on chip,” in *2005 3rd Workshop on Embedded Systems for Real-Time Multimedia*, pp. 47–52.

[52] K. Cheshmi, J. Trajkovic, M. Soltaniyeh, and S. Mohammadi, “Quota setting router architecture for quality of service in GALS NoC,” in *2013 International Symposium on Rapid System Prototyping (RSP 2013)*, pp. 44–50.

[53] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks [Book]*. Morgan Kaufmann, 2004.

[54] W. J. Dally, “Virtual-channel flow control,” in *Proceedings of the 17th annual international symposium on Computer Architecture*, New York, NY, USA, 1990, pp. 60–68.

[55] F. Ge, N. Wu, and Y. Wan, “A network monitor based dynamic routing scheme for Network on Chip,” in *2009 PrimeAsia Asia Pacific Conference on Postgraduate Research in Microelectronics Electronics*, pp. 133 –136.

[56] D. Fan and P. Shi, “Improvement of Dijkstra’s algorithm and its application in route planning,” in *2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2010)*, vol. 4, pp. 1901–1904.

[57] M. Dehyadgari, M. Nickray, A. Afzali-kusha, and Z. Navabi, “Evaluation of pseudo adaptive XY routing using an object oriented model for NOC,” in *2005 17th International Conference on Microelectronics (ICM 2005)*, p. 5 pp.

[58] R. Manevich, I. Cidon, A. Kolodny, I. Walter, and S. Wimer, “A Cost Effective Centralized Adaptive Routing for Networks-on-Chip,” in *2011 14th Euromicro Conference on Digital System Design (DSD 2011)*, pp. 39 –46.

[59] V. Rantala, T. Lehtonen, P. Liljeberg, and J. Plosila, “Distributed Traffic Monitoring Methods for Adaptive Network-on-Chip,” in *2008 Microelectronics conferrence of the Nordic countries (NORCHIP 2008)*, pp. 233 –236.

[60] C. Ciordas, T. Basten, A. Radulescu, K. Goossens, and J. Meerbergen, “An event-based network-on-chip monitoring service,” in *2004 9th IEEE International High-Level Design Validation and Test Workshop*, pp. 149 – 154.

[61] R. B. Mouhoub and O. Hammami, “NOC Monitoring Feedback for Parallel Programmers,” in *2006 IEEE North-East Workshop on Circuits and Systems*, pp. 141 –144.

[62] R. Das, O. Mutlu, T. Moscibroda, and C. Das, *Aérgia: Exploiting Packet Latency Slack in On-Chip Networks*. 2010 International Sympossium on Computer Architecture (ISCA 2010).

[63] D. Andreasson and S. Kumar, “Slack-time aware routing in NoC systems,” in *2005 IEEE International Symposium on Circuits and Systems (ISCAS 2005)*, pp. 2353–2356 Vol. 3.

[64] J. Diemer and R. Ernst, “Back Suction: Service Guarantees for Latency-Sensitive On-chip Networks,” in *2010 Fourth ACM/IEEE International Symposium on Networks-on-Chip (NOCS 2010)*, pp. 155–162.

[65] C. A. Z. Marcelo Daniel Berejuck, “Adding mechanisms for QoS to a network-on-chip.,” *Symposium on Integrated Circuits and Systems Design (SBCCI 2009)*.

[66] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, “QNoC: QoS architecture and design process for network on chip,” *Journal of Systems Architecture*, vol. 50, no. 2–3, pp. 105–128, Feb. 2004.

[67] J. J. H. Pontes, M. T. Moreira, F. G. Moraes, and N. L. V. Calazans, “Hermes-AA: A 65nm asynchronous NoC router with adaptive routing,” in *2010 IEEE International SOC Conference (SOCC 2010)*, pp. 493–498.

[68] M. Mirza-Aghatabar, S. Koohi, S. Hessabi, and M. Pedram, “An Empirical Investigation of Mesh and Torus NoC Topologies Under Different Routing Algorithms and Traffic Models,” in *10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD 2007)*, pp. 19–26.

[69] M. Chatti, S. Yehia, C. Timsit, and S. Zertal, “A hypercube-based NoC routing algorithm for efficient all-to-all communications in embedded image and signal processing applications,” in *2010 International Conference on High Performance Computing and Simulation (HPCS 2010)*, 2010, pp. 623–630.

[70] R. (Reuven) Dobkin, R. Ginosar, and A. Kolodny, “QNoC asynchronous router,” *Integration, the VLSI Journal*, vol. 42, no. 2, pp. 103–115, Feb. 2009.

[71] K. Goossens and A. Hansson, “The aethereal network on chip after ten years: Goals, evolution, lessons, and future,” in *Design Automation Conference (DAC), 2010 47th ACM/IEEE*, 2010, pp. 306 –311.

[72] T. Bjerregaard and J. Sparso, “A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip,” in *Design, Automation and Test in Europe (DATE 2005)*, pp. 1226 – 1231 Vol. 2.

[73] T. Bjerregaard and J. Sparso, “Implementation of guaranteed services in the MANGO clockless network-on-chip,” *Computers and Digital Techniques, IEEE Proceedings* -, vol. 153, no. 4, pp. 217 – 229, Jul. 2006.

[74] S. Penolazzi and A. Jantsch, “A High Level Power Model for the Nostrum NoC,” in *2006 9th EUROMICRO Conference on Digital System Design: Architectures, Methods and Tools (DSD 2006)*, pp. 673–676.

[75] A. Kumar, L.-S. Peh, P. Kundu, and N. K. Jha, “Express Virtual Channels: Towards the Ideal Interconnection Fabric,” in *Proceedings of the 34th Annual International Symposium on Computer Architecture*, New York, NY, USA, 2007, pp. 150–161.

[76] T. Krishna, A. Kumar, P. Chiang, M. Erez, and L.-S. Peh, “NoC with Near-Ideal Express Virtual Channels Using Global-Line Communication,” in *2008 16th IEEE Symposium on High Performance Interconnects*, 2008, pp. 11–20.

[77] B. Grot, J. Hestness, S. W. Keckler, and O. Mutlu, “Kilo-NOC: A heterogeneous network-on-chip architecture for scalability and service guarantees,” in *2011 38th Annual International Symposium on Computer Architecture (ISCA 2011)*, pp. 401–412.

[78] F. Karim, A. Nguyen, and S. Dey, “An interconnect architecture for networking systems on chips,” *2002 IEEE Micro*, vol. 22, no. 5, pp. 36–45.

[79] “NetSpeed Systems | Redefining SoC Design.” [Online]. Available: http://www.netspeedsystems.com/. [Accessed: 29-Nov-2015].

[80] “Arteris - The on-chip SoC communications company.” [Online]. Available: http://www.arteris.com/. [Accessed: 29-Nov-2015].

[81] “Sonics Inc. - The Trusted Leader in On-Chip Networks.” [Online]. Available: http://sonicsinc.com/. [Accessed: 29-Nov-2015].

[82] “Aims Technology Inc.” [Online]. Available: http://aimstechnologyinc.com/company.html. [Accessed: 29-Nov-2015].

[83] A. Laffely, J. Liang, P. Jain, W. Burleson, and R. Tessier, “Adaptive systems on a chip (aSoC) for low-power signal processing,” in *2001 Thirty-Fifth Asilomar Conference on Signals, Systems and Computers*, 2001, vol. 2, pp. 1217–1221 vol.2.

[84] R. Das, S. Narayanasamy, S. K. Satpathy, and R. G. Dreslinski, “Catnap: Energy Proportional Multiple Network-on-Chip,” *2013 ACM/IEEE International Symposium on Computer Architecture (ISCA 2013)*.

[85] I. Miro-Panades, F. Clermidy, P. Vivet, and A. Greiner, “Physical Implementation of the DSPIN Network-on-Chip in the FAUST Architecture,” in *Proceedings of the Second ACM/IEEE International Symposium on Networks-on-Chip*, 2008, pp. 139–148.

[86] M. Forsell, “A scalable high-performance computing solution for networks on chips,” *2002 IEEE Micro*, vol. 22, no. 5, pp. 46–55.

[87] D. Siguenza-Tortosa and J. Nurmi, “VHDL-based simulation environment for Proteo NoC,” in *2002 Seventh IEEE International High-Level Design Validation and Test Workshop*, pp. 1–6.

[88] C. A. Zeferino and A. A. Susin, “SoCIN: a parametric and scalable network-on-chip,” in *2003 16th Symposium on Integrated Circuits and Systems Design (SBCCI 2003)*, pp. 169–174.

[89] R. Salamat and H. R. Zarandi, “Fault-tolerance assessment and enhancement in SoCWire interface: A system-on-chip wire,” in *2011 IEEE 17th International On-Line Testing Symposium (IOLTS 2011)*, pp. 196–197.

[90] S. Stergiou, F. Angiolini, S. Carta, L. Raffo, D. Bertozzi, and G. De Micheli, “Xpipes Lite: a synthesis oriented design library for networks on chips,” in *2005 Design, Automation and Test in Europe*, pp. 1188–1193 Vol. 2.

[91] N. Genko, D. Atienza, G. De Micheli, J. M. Mendias, R. Hermida, and F. Catthoor, “A complete network-on-chip emulation framework,” in *2005 Design, Automation and Test in Europe (DATE 2005)*, 2005, pp. 246 – 251 Vol. 1.

[92] N. Genko, D. Atienza, G.De Micheli, L. Benini, J.M. Mendias, R. Hermida, F. Catthoor, “A novel approach to network on chip emulation,” *IEEE International Symposium on Circuits and Systems, (ISCAS 2005)*.

[93] M. K. Papamichael, “Fast scalable FPGA-based Network-on-Chip simulation models,” in *2011 9th IEEE/ACM International Conference on Formal Methods and Models for Codesign (MEMOCODE 2011)*, 2011, pp. 77 –82.

[94] Z. Tan, A. Waterman, R. Avizienis, Y. Lee, H. Cook, D. Patterson, and K. Asanovic, “RAMP gold: An FPGA-based architecture simulator for multiprocessors,” in *2010 47th ACM/IEEE Design Automation Conference (DAC 2010)*, 2010, pp. 463 –468.

[95] R. Surepeddi, “System Verilog for Quality of Results (QoR),” in *2008 9th International Symposium on Quality Electronic Design (ISQED 2008)*, 2008, pp. 460–464.

[96] Z. Tan, K. Asanovic, and D. Patterson, *An FPGA Host-Multithreaded Functional Model for SPARC v8*. International Symposium on Computer Architecture (ISCA 2008).

[97] M. Lis, K. S. Shim, M. H. Cho, P. Ren, O. Khan, and S. Devadas, “Darsim: A Parallel Cycle-Level NoC Simulator,” *MIT web domain available at http://hdl.handle.net/1721.1/59832 [Accessed: 2015-06-03]*, 2010.

[98] N. Jiang, D. U. Becker, G. Michelogiannakis, J. Balfour, B. Towles, D. E. Shaw, J. Kim, and W. J. Dally, “A detailed and flexible cycle-accurate Network-on-Chip simulator,” in *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS 2013)*, 2013, pp. 86–96.

[99] K.-L. Lin, C.-K. Lo, and R.-S. Tsay, “Source-level timing annotation for fast and accurate TLM computation model generation,” in *2010 15th Asia and South Pacific Design Automation Conference (ASP-DAC 2010)*, 2010, pp. 235–240.

[100] J. Aynsley, “TLM-2.0 Language Reference Manual.” OSCI, Tech. Rep., 2009 available at http://accellera.org/downloads/standards/systemc [Accessed 2015-06-15].

[101] G. Schirner and R. Dömer, “Quantitative Analysis of the Speed/Accuracy Trade-off in Transaction Level Modeling,” *ACM Transaction of Embedded Computing Systems*, vol. 8, no. 1, pp. 4:1–4:29, Jan. 2009.

[102] K. Goossens, J. Dielissen, and A. Radulescu, “AEthereal network on chip: concepts, architectures, and implementations,” *Design & Test of Computers, IEEE*, vol. 22, no. 5, pp. 414–421, 2005.

[103] K. Yan, H. Yang, and H. Wang, “A Low Latency Variance NoC Router,” in *Embedded and Multimedia Computing Technology and Service*, J. J. (Jong H. Park, Y.-S. Jeong, S. O. Park, and H.-C. Chen, Eds. Springer Netherlands, 2012, pp. 89–97.

[104] S. Tobuschat, P. Axer, R. Ernst, and J. Diemer, “IDAMC: A NoC for mixed criticality systems,” in *2013 IEEE 19th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2013)*, 2013, pp. 149–156.

[105] J. Kim, W. J. Dally, J. Dally, and D. Abts, “Adaptive Routing in High-Radix Clos Network,” in *Proceedings of the ACM/IEEE SC 2006 Conference*, 2006, pp. 7–7.

[106] J. Zareei and A. H. Kakaee, “Study and the effects of ignition timing on gasoline engine performance and emissions,” presented at the European Transport Research Review. (2013) 5:109 – 116.

[107] I. Scollar, B. Weidner, and T. S. Huang, “Image Enhancement Using the Median and the Interquartile Distance,” *1984 Computer vison, graphics and image processing*.

[108] F. Ajil Jassim, “Image Denoising Using Interquartile Range Filter with Local Averaging,” *International Journal of Soft Computing and Engineering (IJSCE 2013)*.

[109] K. D. Buch, “Decision based non-linear filtering using interquartile range estimator for Gaussian signals,” in *2014 Annual IEEE India Conference (INDICON 2014)*, pp. 1–5.

[110] M. Daniel Berejuck, “Network-on-Chip with load balancing based on interleave of flits technique,” *Computing Research Repository (CoRR 2015)*.

[111] G. Nychis, T. Moscibroda, O. Mutlu, and S. Seshan, “On-Chip Networks from a Networking Perspective: Congestion and Scalability in Many-Core Intercon-

nects," *ACM Special Interest Group on Data Communication (SIGCOMM 2012)*, no. ACM 978–1–4503–1419–0/12/08.

[112] "Artix-7 FPGA Family," *Artix-7 FPGA Family*. [Online]. Available: http://www.xilinx.com/products/silicon-devices/fpga/artix-7.html. [Accessed: 08-Aug-2015].

[113] D. O'Loughlin, A. Coffey, F. Callaly, D. Lyons, and F. Morgan, "Xilinx Vivado High Level Synthesis: Case studies," in *25th China-Ireland International Conference on Information and Communications Technologies/ 2014 IET Irish Signals Systems Conference (ISSC 2014/CIICT 2014)*, pp. 352–356.

[114] A. Fpgas, K. Gaj, E. Homsirikamol, M. Rogawski, R. Shahid, and M. U. Sharif, *Comprehensive Evaluation of High-Speed and Medium-Speed Implementations of Five SHA-3 Finalists Using*. Springer 264-278,vol 6225, Heidelberg 2009.

[115] Z. Guo, W. Najjar, F. Vahid, and K. Vissers, "A Quantitative Analysis of the Speedup Factors of FPGAs over Processors," in *Proceedings of the 2004 ACM/SIGDA 12th International Symposium on Field Programmable Gate Arrays*, New York, USA, 2004, pp. 162–170.

[116] N. Genko, D. Atienza, G. De Micheli, and L. Benini, "Feature - NoC emulation: a tool and design flow for MPSoC," *IEEE Circuits and Systems Magazine (2007)*, vol. 7, no. 4, pp. 42–51.

[117] "High-Level Synthesis Tools." [Online]. Available: http://www.bluespec.com/high-level-synthesis-tools.html. [Accessed: 05-Jun-2015].

[118] R. Nikhil, "Bluespec System Verilog: efficient, correct RTL from high level specifications," in *Proceedings of Second ACM and IEEE International Conference on Formal Methods and Models for Co-Design (MEMOCODE 2004)*, 2004, pp. 69–70.

[119] M. Ghasempour, L. Mikel, and J. Garside, *SoC Simulator on FPGA using Bluespec System Verilog*. UK Electronics Forum (UKEF 2012).

[120] F. Gruian and M. Westmijze, "VHDL vs. Bluespec System Verilog: A Case Study on a Java Embedded Architecture," in *2008 ACM Symposium on Applied Computing*, New York, NY, USA, pp. 1492–1497.

[121] B. Kim, J. Kim, S. Hong, and S. Lee, "A real-time communication method for wormhole switching networks," in *Proceedings of 1998 International Conference on Parallel Processing*, 1998, pp. 527–534.

[122] B. Sprunt, "Pentium 4 performance-monitoring features," *IEEE Micro*, vol. 22, no. 4, pp. 72 – 82, Aug. 2002.

[123] V. Todorov, A. Ghiribaldi, H. Reinig, D. Bertozzi, and U. Schlichtmann, "Non-intrusive Trace & Debug Noc Architecture with Accurate Timestamping for GALS SoCs," in *Proceedings of the Eighth IEEE/ACM/IFIP International Con-*

ference on Hardware/Software Codesign and System Synthesis, New York, NY, USA, 2012, pp. 181–186.

[124] B. Sudev and L. S. Indrusiak, "Dynamic Time Multiplexed Virtual Channels, a Performance Scalable Approach in Network-On-Chip Routers to Reduce Packet Starvation," in *York Doctoral Symposium on Computer Science & Electronics (YDS)*, 2014, pp. 21–30.

[125] A. Burns and R. I. Davis, "Mixed Criticality on Controller Area Network," in *2013 25th Euromicro Conference on Real-Time Systems (ECRTS 2013)*, 2013, pp. 125–134.

[126] A. Burns, J. Harbin, and L. S. Indrusiak, "A Wormhole NoC Protocol for Mixed Criticality Systems," in *2014 IEEE Real-Time Systems Symposium (RTSS 2014)*, 2014, pp. 184–195.

[127] L. S. Indrusiak, A. Burns, and J. Harbin, "Average and worst-case latency improvements in mixed-criticality wormhole networks-on-chip.," presented at the Euromicro Technical Committee on Real-Time Systems (ECRTS 2015), Lund, Sweden, 2015.

[128] X. Chen and L.-S. Peh, "Leakage power modeling and optimization in interconnection networks," in *Proceedings of the 2003 International Symposium on Low Power Electronics and Design (ISLPED 2003)*, 2003, pp. 90–95.