THESIS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Dual Data Rate Network-on-Chip Architectures

AHSEN EJAZ

Division of Computer Engineering
Department of Computer Science & Engineering
Chalmers University of Technology
Göteborg, Sweden, 2021

Dual Data Rate Network-on-Chip Architectures

AHSEN EJAZ

Advisor: Ioannis Sourdis, Professor at Chalmers University of Technology
Co-Advisor: Vassilios Papaefstathiou, Researcher at FORTH-ICS, Greece

Examiner: Per Stenström, Professor at Chalmers University of Technology

Thesis Opponent:
Tushar Krishna, Assistant Professor at Georgia Institute of Technology, USA

Grading Committee:
Giorgos Dimitrakopoulos, Associate Professor at Democritus Univ. of Thrace, Greece
Boris Grot, Associate Professor at University of Edinburgh, UK
Jens Sparsø, Professor at Technical University of Denmark, Denmark
Lars Svensson, Senior Lecturer at Chalmers Univ. of Technology (deputy member)

ISBN: 978-91-7905-497-7
Doktorsavhandlingar vid Chalmers tekniska högskola.
Series number: 4964
ISSN: 0346-718X
Technical report number: 197D

Department of Computer Science & Engineering
Division of Computer Engineering
Chalmers University of Technology
Göteborg, Sweden

This thesis has been prepared using LaTeX.
Printed by Chalmers Reproservice,
Göteborg, Sweden 2021.

Abstract

Networks-on-Chip (NoCs) are becoming increasing important for the performance of modern multi-core systems-on-chip. The performance of current NoCs is limited, among others, by two factors: their limited clock frequency and long router pipeline. The clock frequency of a network defines the limits of its saturation throughput. However, for high throughput routers, clock is constrained by the control logic (for virtual channel and switch allocation) whereas the datapath (crossbar switch and links) possesses significant slack. This slack in the datapath wastes network throughput potential. Secondly, routers require flits to go through a large number of pipeline stages increasing packet latency at low traffic loads. These stages include router resource allocation, switch traversal (ST) and link traversal (LT). The allocation stages are used to manage contention among flits attempting to simultaneously access switch and links, and the ST stage is needed to change the routing dimension. In some cases, these stages are not needed and then requiring flits to go through them increases packet latency. The aim of this thesis is to improve NoC performance, in terms of network throughput, by removing the slack in the router datapath, and in terms of average packet latency, by enabling incoming flits to bypass, when possible, allocation and ST stages. More precisely, this thesis introduces Dual Data-Rate (DDR) NoC architectures which exploit the slack present in the NoC datapath to operate it at DDR. This requires a clock with period twice the datapath delay and removes the control logic from the critical path. DDR datapaths enable throughput higher than existing single data-rate (SDR) networks where the clock period is defined by the control logic. Additionally, this thesis supplements DDR NoC architectures with varying levels of pipeline stage bypassing capabilities to reduce low-load packet latency. In order to avoid complex logic required for bypassing from all inputs to all outputs, this thesis implements and evaluates a simplified bypassing approach. DDR NoC routers support bypassing of the allocation stage for flits propagating an in-network straight hop (i.e. East to West, North to South and vice versa) and when entering or exiting the network. Disabling bypassing during XY-turns limits its benefits, but, for most routing algorithms under low traffic loads, flits encounter at most one XY-turn on their way to the destination. Bypassing allocation stage enables incoming flits to directly initiate ST, when required conditions are met, and propagate at one cycle per hop. Furthermore, DDR NoC routers allow flits to bypass the ST stage when propagating a straight hop from the head of a specific input VC. Restricting ST bypassing from a specific VC further simplifies check logic to have clock period defined by datapath delays. Bypassing ST requires dedicated bypass paths from non-local input ports to opposite output ports. It enables flits to propagate half a cycle per hop. This thesis shows that compared to current state-of-the-art SDR NoCs, operating router's datapath at DDR improves throughput by up to 20%. Adding to a DDR NoC an allocation bypassing mechanism for straight hops reduces its packet latency by up to 45%, while maintaining the DDR throughput advantage. Enhancing allocation bypassing to include flits entering and exiting the network further reduces latency by another 24%. Finally, adding ST bypassing further reduces latency by another 32%. Overall, DDR NoCs offer up to 40% lower latency and about 20% higher throughput compared to the SDR networks.

Keywords

Network-on-Chip, On-Chip Interconnect, System-on-Chip, Dual Data-Rate, Multiprocessor System-on-Chip, Chip Multiprocessors

Acknowledgment

First of all, I would like to express my gratitude to my PhD supervisor Ioannis Sourdis for giving me the opportunity to pursue a PhD degree under his supervision. Ioannis has been very patient and supportive during my research and has kept me motivated since I started on this path to a PhD degree.

I am also grateful to my co-supervisor, Vassilis Papaefstathiou, for his valuable comments and feedback based on his deep insight of the field, during our meetings.

I would also like to thank my PhD research examiner, Per Stenström, and PhD follow-up committee members Jan Jonsson, Agneta Nilsson, Wolfgang Ahrendt and Nir Piterman, for their constructive feedback and support over the years.

I am also very grateful to Lars Svensson for his useful critiques of my research and for his support with setting up the physical design tools I needed.

Place-and-Route of the networks I designed had been a daunting and time consuming task, and it would have taken much longer had it not been for the useful impromptu discussions with Kevin Cushon, Christoffer Fougstedt, Erik Ryman and Victor Åberg. Thanks guys.

I would also like to thank the administrative staff of the department, Eva Axelsson, Monica Månhammar, Marianne Pleen-Schreiber, Nadja Johansson, Lars Norén, Michael Morin and Rune Ljungbjörn for their help and support.

I am also thankful to my colleagues in the Department of Computer Science and Engineering, Muhammad Waqar Azhar, Evangelos Vasilakis, Prajith Ramakrishnan Geethakumari, Albin Eldstal-Damlin, Stefano Ribes, Mehrzad Nejat, Petros Voudouris, Bhavishya Goel, Madhavan Manivannan and Pedro Petersen Moura Trancoso for our various discussions and for creating a positive work environment.

I would also like to thank my office mates over the long years, Alirad Malek, Stavros Tzilis and Piyumal Ranawaka for keeping me motivated and entertained.

This work has been partly funded by EUROSERVER project (grant agreement 610456), EuroLab-4-HPC project (grant agreement 371610) and Horizon 2020 Programme ECOSCALE project (grant agreement 671632).

Finally, none of this would have been possible without the unfaltering support, encouragement and prayers of my parents, my siblings and my wife. I am really grateful to have them by my side.

Ahsen Ejaz
Göteborg, May 2021

List of Publications

This thesis is based on the following publications:

1. **Ahsen Ejaz**, Vassilios Papaefstathiou and Ioannis Sourdis, "DDRNoC: Dual Data-Rate Network-on-Chip", *ACM Transactions on Architecture and Code Optimization (TACO), vol. 15, no. 2, June 2018.*

2. **Ahsen Ejaz**, Vassilios Papaefstathiou and Ioannis Sourdis, "FreewayNoC: a DDR NoC with Pipeline Bypassing", *IEEE/ACM International Symposium on Networks-on-Chip, 2018.*

3. **Ahsen Ejaz**, Vassilios Papaefstathiou and Ioannis Sourdis, "HighwayNoC: Approaching Ideal NoC Performance With Dual Data Rate Routers", *IEEE/ACM Transactions on Networking, vol. 29, no. 1, pp. 318-331, 2021.*

4. **Ahsen Ejaz** and Ioannis Sourdis, "FastTrackNoC: A NoC with FastTrack Router Datapaths", *Under submission, 2021.*

Contents

Abstract iii

Acknowledgement v

List of Publications vii

1 Introduction 1
- 1.1 Problem Statement . . . 2
  - 1.1.1 Router Control Logic Limits NoC Throughput . . . 2
  - 1.1.2 Deep Router Pipeline Increases Packet Latency . . . 3
- 1.2 Thesis Objectives . . . 3
  - 1.2.1 Improving NoC Throughput . . . 3
    - 1.2.1.1 Related Work . . . 3
    - 1.2.1.2 Objective 1: Increase Network Throughput with DDR Datapath . . . 4
  - 1.2.2 Reducing NoC Packet Latency . . . 5
    - 1.2.2.1 Related Work . . . 6
    - 1.2.2.2 Objective 2: Reduce Packet Latency with Control Forwarding in a DDR Router . . . 6
    - 1.2.2.3 Objective 3: Reduce Packet Latency with Allocation Bypassing in a DDR Router . . . 6
    - 1.2.2.4 Objective 4: Reduce Packet Latency with ST Bypassing . . . 8
- 1.3 Thesis Outline . . . 8

2 Related Work 9
- 2.1 Increasing NoC Throughput . . . 9
- 2.2 Reducing NoC Packet Latency . . . 11
- 2.3 Summary . . . 14

3 The DDRNoC Architecture 15
- 3.1 Router Datapath . . . 15
- 3.2 Timing . . . 17
- 3.3 Zero-Load Latency Analysis . . . 19
- 3.4 DDRNoC Control . . . 20
  - 3.4.1 Virtual Channel Allocation . . . 20
  - 3.4.2 Switch Allocation . . . 20
  - 3.4.3 Flow Control and Minimum Buffer Size . . . 21

3.5 Discussion . . . 22
3.6 Summary . . . 22

4 The FreewayNoC Architecture 23
4.1 Router Datapath . . . 24
4.1.1 DDR Flit Datapath . . . 24
4.1.2 Datapath of Forwarded Control signals . . . 25
4.2 Timing Example . . . 25
4.3 Zero-Load Latency Analysis . . . 26
4.4 Router Control . . . 27
4.4.1 Combined VC and Switch Allocation . . . 27
4.4.2 Allocation Bypassing . . . 29
4.4.3 Next Route Computation . . . 30
4.4.4 Flow Control and Minimum Buffer Size . . . 30
4.4.5 Control Path Analysis . . . 31
4.5 Summary . . . 31

5 The HighwayNoC Architecture 33
5.1 Router Datapath . . . 34
5.1.1 DDR Flit Datapath . . . 34
5.1.2 Datapath of Forwarded Control Bits . . . 35
5.2 Timing . . . 37
5.3 Zero-Load Latency Analysis . . . 38
5.4 Router Control . . . 39
5.4.1 Combined VC and Switch Allocation . . . 39
5.4.2 Allocation Bypassing . . . 41
5.4.3 Next Route Computation . . . 43
5.4.4 Flow Control and Minimum Buffer Size . . . 44
5.4.5 Control Path Analysis . . . 44
5.5 Summary . . . 45

6 The FastTrackNoC Architecture 47
6.1 Router Datapath . . . 48
6.1.1 DDR Flit Datapath . . . 49
6.1.2 Datapath of Forwarded Control Signals . . . 50
6.2 Router Control . . . 52
6.2.1 Combined VC and Switch Allocation . . . 52
6.2.2 Allocation Bypassing Controller . . . 53
6.2.3 FastTrack Controller . . . 53
6.2.4 Next Route Computation . . . 55
6.2.5 Flow Control and Minimum Buffer Size . . . 55
6.3 Zero-Load Latency Analysis . . . 56
6.4 Timing Example . . . 57
6.5 Summary . . . 59

7 Evaluation **61**
7.1 Experimental Setup 61
7.2 Implementation Results 63
7.3 Performance Evaluation 65
7.3.1 Performance with Synthetic Traffic 65
7.3.2 Performance Comparison against Networks with Longer Links 66
7.3.3 Performance Comparison against Networks with DDR Links 74
7.3.4 Energy Efficiency 74
7.3.5 Trading DDRNoC Performance Advantage for Energy Efficiency 77
7.3.6 Evaluation with Application Driven Traffic 79
7.3.7 Measuring the Impact of DDRNoC Throughput Gain on System Performance 80
7.4 Summary 82

8 Conclusions **83**
8.1 Summary 83
8.2 Contributions 84
8.3 Future Work 86

Bibliography **89**

Chapter 1

Introduction

Chip Multiprocessors (CMP) are one of the most promising solutions for supporting the continuous need for single chip performance improvement. Shrinking transistor geometries still allow more cores to be integrated on a die [1–3]. The increasing number of cores as well as the increasingly complex workloads place stringent requirements over the underlying communication fabric. On one hand, applications that exhibit small transfers at low loads are often more sensitive to network latency [4]. On the other hand, systems that run concurrent scale-out applications are more demanding, push the network close to its saturation point, and are more sensitive to network throughput [4, 5]. Moreover, power constraints prevent chips from fully utilizing all these cores at their maximum performance potential [6]. The on-chip interconnection network is a critical component for power efficiency as well [7] since roughly a sixth of the available chip power budget, if not more, goes to its interconnects [8–14]. As a consequence, the design of high-performance and low-power networks-on-chip (NoCs) is essential for many-core scaling.

In the past decades, NoC designs have improved their performance substantially, both in terms of throughput and in terms of latency.

Network throughput has increased with the use of better topologies [15] and multiple sub-networks [16, 17] as well as with improvements in allocation [18]. Higher throughput is also achieved by splitting switch traversal (ST) and link traversal (LT) into two pipeline stages. Currently, NoCs require complex control logic to implement various allocation schemes which allocate available network resources to propagating packets and flits. For high throughput networks, the delay of their control is longer than the datapath, which then leads to define the NoC clock period [18–21]. The datapath in such a case possesses some slack, which essentially wastes bandwidth. Compared to an ideal network which would operate at a clock rate defined only by the datapath delay, current 2D-mesh NoC architectures with virtual channel (VC) flow control still offer lower throughput. This is because an ideal network would deliver more packets per node as a consequence of improved datapath utilization achieved by removing the slack present in the datapath. Higher network throughput obtained from increased datapath utilization would also improve the performance of the entire system.

Furthermore, packet latency has previously been reduced with improvements in the network topology [22], the routing algorithm [23] and the router architecture [24, 25]. In theory, the latency for a single hop, i.e., crossing a tile, can be as short as the (register

to register) delay of a link that traverses the tile dimension. Some existing networks such as single-cycle multi-hop routers [26], routers with express links [22, 27], or routerless rings [28] offer low latency, but sacrifice substantially their throughput. Current high throughput NoC routers have reduced their minimum hop latency to the time spent for ST + LT [13, 29, 30]. This is achieved by a combination of techniques, such as precomputed routing [24], speculative switch allocation (SA) [25], lookahead control signalling [13, 29], and allocation bypassing [13, 30]. Then, considering that ST and LT are pipelined and their delays are split evenly, the minimum hop latency of current NoC routers [30] is roughly twice the above theoretical minimum, i.e. 2×LT.

This introductory Chapter provides a brief overview of the work presented in this thesis. The remainder of this Chapter is organized as follows: Section 1.1 presents the main problem statement of this research. Section 1.2 discusses the key objectives of this thesis.

1.1 Problem Statement

A NoC router has two sets of components: router datapath and control logic. Router datapath is composed of input/output link wires, VC buffers, crossbar and output registers. It is used to either store flits in NoC routers due to contention or unavailability of required resources, or to move them forward through the network towards their destination. The control logic, on the other hand, is responsible for managing the flow of flits propagating through the network by allocating router datapath resources to them. It also resolves contention among flits competing to use shared router resources at the same time. It comprises next route computation (NRC) logic, VC allocators (VA) and switch allocators (SA). In order to increase network clock frequency, the control logic and the datapath of NoC routers are divided into smaller and faster pipeline stages. The delay of the control stages is longer than the datapath stages because of greater complexity of allocators. This leads to an imbalance in router pipeline which limits network's clock frequency and prevents it from achieving throughput close to an ideal network with a balanced router pipeline. Furthermore, a flit must go through many pipeline stages to traverse a router. Traversing pipeline stages when they are not needed delays packets and increases average packet latency unnecessarily. These two inefficiencies are further discussed below.

1.1.1 Router Control Logic Limits NoC Throughput

As shown in literature and confirmed by our experiments, the critical path of existing 2D-mesh routers with multiple VCs, is in its control logic [18, 19, 21, 24]. More specifically, the clock period of a typical 3-stage (NRC/VA/SA, switch traversal (ST), link traversal (LT)) router [24] is dictated by the VA/SA stage of the router pipeline. Even for one of the fastest state-of-the-art NoC routers with VC-flow control, ShortPath [30], a significant fraction (about 20%) of the clock period is reserved for router control. Consequently, the datapath possesses considerable slack and is used only for a fraction of the clock cycle to forward the flit and remains idle otherwise. This reduces network datapath utilization, wastes link bandwidth and hinders the NoC from achieving its maximum performance potential by limiting its saturation throughput and increasing packet latency.

1.1.2 Deep Router Pipeline Increases Packet Latency

Conventional NoC routers have three pipeline stages: (i) VA/SA, (ii) ST and (ii) LT, but going through all these stages is not always necessary. For example, the VA/SA stage, which allocates switch resources to flits, is not needed in the absence of competing flits and can be bypassed. Similarly, ST stage is needed for changing the routing dimension but it could be bypassed when propagating in a single dimension. Many previous NoC architectures allow incoming flits to bypass the SA stage and save a clock cycle per hop [13, 29, 30]. These networks, however, do not allow flits to bypass the ST stage, which could further reduce packet latency.

1.2 Thesis Objectives

In this thesis, we target the slack in the datapath as well as long pipeline of NoC routers. VC based 2D-mesh NoC architectures are targeted aiming to increase NoC datapath utilization and to avoid delays caused by deep router pipeline. We aim at designing a NoC router architecture capable of achieving throughput defined solely by the network datapath delays and minimum hop latency close to the LT delay. The implications of such a network on throughput and latency are presented below along with a brief description of related works.

1.2.1 Improving NoC Throughput

One of the factors limiting the throughput of high performance single data-rate (SDR) networks is the under-utilization of its datapath during a clock cycle. It is caused by an imbalance in the datapath and the control logic path delays of SDR NoC routers, as shown in Figure 1.1. This imbalance exists because the complex allocation schemes employed to control these routers are substantially slower than the datapath. Then the clock period of the resulting NoC router is longer than the datapath delay and leads to datapath under-utilization during the cycle. We analyze the throughput of one of the fastest state-of-the-art SDR NoC architectures, called the ShortPath network [30] and compare it to the throughput of an ideal (ShortPath) network with clock period defined purely by datapath delays. The results for 32×32 2D-mesh networks with uniform random traffic of 1 and 5 flit packets, presented in Figure 1.2, indicate about 20% throughput gap between the ShortPath network operated at its maximum clock frequency versus the clock frequency defined by its datapath delays. This is because clock period for ShortPath routers is shared by both its datapath and control logic making it about 20% longer than the datapath delays.

1.2.1.1 Related Work

Existing NoC architectures attempt to either reduce or exploit the imbalance between the router datapath and control aiming to improve performance. Many architectural techniques try to simplify VA and SA to reduce allocation complexity and thereby the router critical path delay [13, 30]. Others employ low level implementation techniques like time stealing (or retiming), currently supported by CAD tools, to balance the pipeline of a router and have a clock period equal to the average delay of all router stages [20]. Nevertheless, for all these techniques, the control logic either ends up with a delay larger than the datapath delay or it adds to the datapath delay causing a

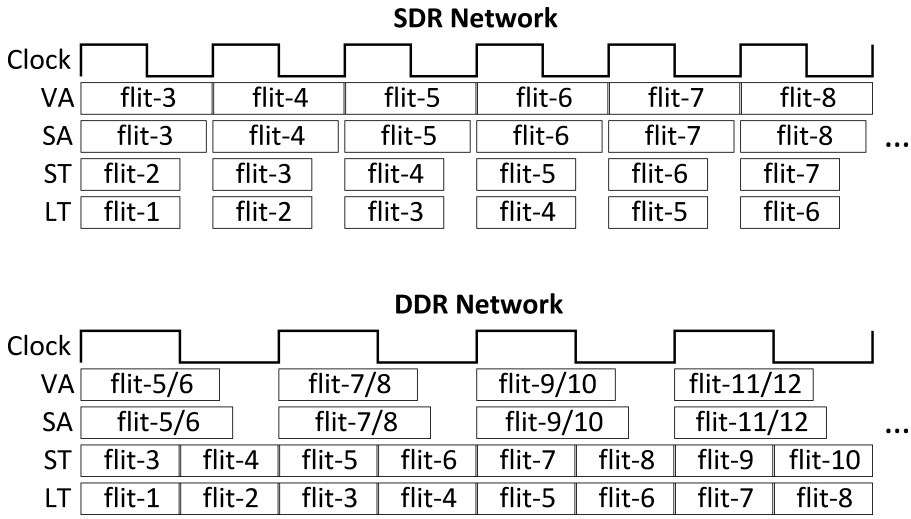


Figure 1.1: For a conventional SDR network, the clock period is completely (or partly) determined by the control logic. The DDR network extends the clock period to twice the datapath delay, implements control logic capable of allocating datapath resources to two flits per cycle per port and also enables the datapath (ST and LT) to propagate two flits per cycle. The clock period is now defined only by the datapath delays. This improves datapath utilization and enhances network throughput.

single clock cycle to be sequentially shared by a part of both the control logic and the datapath. Both these cases result in a router critical path which includes some control logic, require a clock with a period larger than the datapath delay and lead to NoC datapath under-utilization. The router architecture we present in this thesis minimizes the share of the control logic in the network clock period by using DDR datapaths.

In the past, NoC research community has proposed router architectures which employ DDR flit traversal but only in parts of a NoC. Such NoC routers either use double-pumped crossbars [9] or DDR links [17, 31, 32]. These designs employ DDR over a part of the router's datapath to improve either area, power or performance but they do not eliminate the slack in the router datapath, so they still allow its under-utilization. Contrary to these works, our approach increases network throughput by enabling DDR flit propagation over all parts of the router datapath to route flits at a rate determined by the datapath delays rather than by the control logic delays.

1.2.1.2 Objective 1: Increase Network Throughput with DDR Datapath

The first objective of this thesis is to improve NoC throughput by removing the slack present in the router's datapath and having its clock frequency defined by datapath delays. Figure 1.2 shows the targeted gap in the throughput of ShortPath [30], one of the fastest SDR NoCs, when operated at its maximum clock frequency versus the clock frequency defined solely by its datapath delays. In order to cover this throughput gap, this thesis proposes the DDRNoC architecture, presented in Chapter 3, which operates the router datapath at DDR allowing two flits to propagate in a single cycle as shown in Figure 1.1. Contrary to current SDR routers which have clock period defined by slow control logic, the clock period of DDRNoC is equal to twice the delay of the

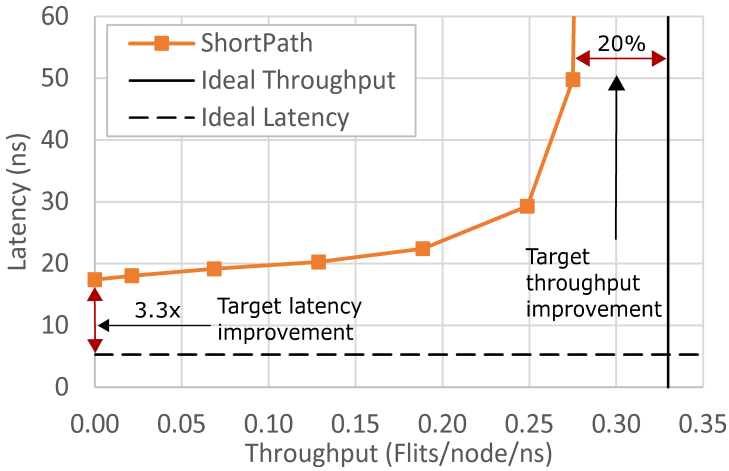


Figure 1.2: Throughput and latency of one of the fastest SDR networks, called the ShortPath network [30], is presented in comparison to the ideal network throughput and latency. Network size of 32×32 is considered with uniform random traffic of 1 and 5 flit packets. Ideal throughput is the saturation throughput of the ShortPath network when operated at a clock frequency defined solely by the datapath (ST and LT) delays. Ideal latency is for a (pipelined) direct connection between the source and the destination nodes, separated by a hop distance equal to the average hop count with uniform random traffic in a 32×32 2D-mesh network. The link for this direct connection is pipelined considering tile dimensions similar to ShortPath and it considers a packet containing 3 flits. All networks are operated at their maximum clock frequency.

slowest datapath stage ($2 \times max(ST, LT)$). DDR transmission of flits in the network ensures that the datapath can potentially be utilized for the complete duration of a cycle, serving a flit during both high and low clock phases. For a balanced router datapath, where the ST and LT delays are almost the same, this removes the slack from the router datapath and shifts it to the control logic as shown in Figure 1.1. DDRNoC uses this slack to modify the SA of the router to allocate input and output ports of the crossbar switch to up to two flits in a clock cycle. It is important to note that whereas the datapath of the DDRNoC (crossbar and link) can be used twice in a single cycle, the control logic (VA and SA) uses a complete cycle to make allocation decisions. The logic delay of VA and SA is less than twice the delay of the longest datapath stage (ST or LT), so the NoC routers can operate at a rate defined by datapath delays and achieve higher throughput.

1.2.2 Reducing NoC Packet Latency

Although the DDR traversal of flits over the NoC datapath significantly improves the throughput of the network compared to SDR networks, the latency of packets in the network suffers because the network operates at a lower clock frequency (although at DDR). In order to improve packet latency this thesis supplements the DDR flit traversal in a router with techniques that reduce average packet latency.

1.2.2.1 Related Work

On the packet latency front, previous NoC designs use control forwarding (also called lookahead signaling) to inform the downstream router of incoming flits one clock cycle earlier. This enables the downstream router to complete VA and SA for incoming flits before their arrival [13, 33, 34], effectively hiding the cycle required for these allocation stages for each hop. Moreover, networks which allow flits to bypass allocation stages in the absence of contention, usually under low network load, have also been proposed [29, 30, 34]. These networks enable incoming flits to bypass the allocation stages when no competing requests exist. We analyze the performance of the ShortPath network [30], one of the fastest SDR networks which enables incoming flits to bypass allocation stages, when required conditions are met, and compare its average packet latency to that of an ideal pipelined connection between source and destination nodes. The results, presented in Figure 1.2, demonstrate that ShortPath has 3.3$\times$ higher packet latency compared to the ideal at low loads. This is because of two reasons: (i) all flits must spend at least 2 cycles per hop to go through ST and LT stages and (ii) clock period of the ShortPath network is about 20% longer than its ST delay which not only increases packet latency but also wastes datapath bandwidth. Our aim is to reduce the low-load packet latency gap by utilizing control forwarding and bypassing both SA and ST stages when possible. We analyze the application of these techniques in routers where both the control and the datapath allow flits to be routed at DDR. Moreover, we implement these techniques while ensuring that the critical path remains with in the datapath of the router.

1.2.2.2 Objective 2: Reduce Packet Latency with Control Forwarding in a DDR Router

The second objective of this thesis is to implement control forwarding in DDR NoC routers, to reduce packet latency. Control forwarding is used to transmit and register control signals at the downstream router one cycle before their corresponding flits are received. This enables the downstream router to initiate VA/SA in advance and to prepare the input VC registers to register the (up to) two incoming flits at the proper clock edge (negative clock edge for the flit propagating link during the high clock phase and vice versa). By the time the (up to) two incoming flits (per input port) are registered in the input VC buffers of the downstream router, their allocation decisions are already made, which, if successful, allow the flits to initiate ST without any allocation overhead. Control forwarding overlaps LT stage of the current router with VA/SA stage of the downstream router and hides the cycle a flit needs at each hop for allocation. For DDR NoC architectures, control forwarding reduces packet zero-load latency by one cycle per hop. It is implemented using dedicated crossbar switches (for control switch traversal), link wires (for control link traversal) and bypass paths which are separate from the flit datapath. Control forwarding is described in more detail in Chapter 3 of this thesis for the DDRNoC architecture.

1.2.2.3 Objective 3: Reduce Packet Latency with Allocation Bypassing in a DDR Router

The third objective of this thesis is to allow incoming flits in a DDR router to bypass the SA pipeline stage, when required network resources available and free from contention, and immediately initiate ST, followed by LT, in order to reduce packet

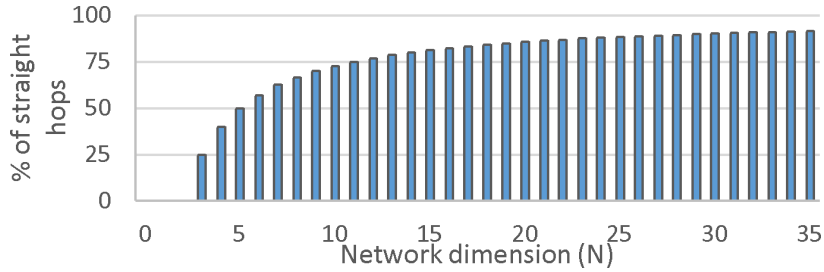


Figure 1.3: Percentage of non-turning hops in 2D-mesh networks of size N×N with dimension order routing for uniform random traffic.

latency. Allocation bypassing targets end-to-end packet latency at low traffic loads when packets encounter reduced contention on their way to the destination. By allowing the incoming flits at a router input port to bypass the allocation stage, in the absence of contention, and immediately undergo ST, followed by LT, if downstream buffer resources are available, the amount of time a flit spends in the network is significantly reduced. To this end, this thesis proposes the FreewayNoC (described in Chapter 4) and the HighwayNoC (described in Chapter 5) router architectures which improve over the DDRNoC, by implementing allocation bypassing at different levels, in order to reduce zero-load packet latency.

The FreewayNoC architecture, which implements the simplest form of allocation bypassing, allows incoming flits to bypass the SA stage when propagating straight through a router (i.e. N to S, E to W and vice versa). Here, SA bypassing is supported only for straight flits because offering complete bypassing, from all inputs to all outputs, would require complex checks to ensure conflict-free ST and LT. The delay of these checks is prohibitively large and it would reduce the clock frequency of the network. So, FreewayNoC reduces the complexity of bypassing check logic by only enabling flits propagating a straight hop to bypass SA stage. This reduces the benefit of allocation bypassing, however, for most routing algorithms for 2D mesh networks, the number of turns a flit takes during its journey under low network loads are only two or three (to/from local port and an XY turn). Moreover, as shown in Figure 1.3, within the network, non-turning hops are more common than turning hops for uniform random traffic, specially for larger 2D-mesh network sizes. Further details of the FreewayNoC architecture are presented in Chapter 4 of this thesis.

The HighwayNoC improves over the FreewayNoC by enhancing the allocation bypassing support. In addition to allowing flits to bypass the allocation stage when propagating a straight hop, HighwayNoC further allows flits entering or exiting the network to bypass allocation as well. Flits entering through the local port and attempting to bypass to a non-local output can face competition from flits already in the network which are also trying to bypass to the same non-local output. HighwayNoC resolves this contention by always giving priority to in-network bypassing flits and requiring competing flits entering through the local input to request SA. Similarly, a router can simultaneously receive flits at multiple non-local inputs attempting to bypass to the local output port. In order to resolve contention in this case, HighwayNoC uses a simple fixed priority arbiter which selects the non-local input port which will be allowed to bypass allocation and directly switch incoming flits to the local output port. Other competing flits have to request allocation. HighwayNoC tolerates bypassing

check logic delays at the local port by utilizing the fact that there is a shorter link (relative to inter-router links) connecting the local port of a router to the network interface. Further details of the HighwayNoC architecture are presented in Chapter 5 of this thesis.

1.2.2.4 Objective 4: Reduce Packet Latency with ST Bypassing

The fourth objective of this thesis is to enable incoming flits in a DDR NoC router to bypass the ST stage, in addition to the SA stage, when required network resources are available and free from contention, and immediately initiate LT, in order to reduce packet latency. To this end, this thesis proposes the FastTrackNoC architecture which improves over the HighwayNoC and allows incoming flits propagating an in-network straight hop to bypass the ST stage as well as the SA stage and directly initiate LT, in half a clock cycle. It utilizes dedicated intra-router bypass paths which allow both the data and the forwarded control information to bypass the crossbar switch in order to initiate LT. To reduce complexity of the bypass paths and satisfy tight timing constraints, ST bypassing requires pre-computation of the required checks and is restricted to flits at the head of a specific input VC propagating an in-network straight hop. As a result, FastTrackNoC maintains the clock period of previously described DDR NoC routers which do not offer ST bypassing and is able to reduce the low-load packet latency gap to ideal, shown in Figure 1.2. Further details on the FastTrackNoC architecture are presented in Chapter 6 of this thesis.

The contributions of this thesis towards the above mentioned objectives are presented at the end of this thesis, in Section 8.2.

1.3 Thesis Outline

The remainder of this thesis is organized as follows: Chapter 2 discusses the related work. Chapter 3 describes the design details of the DDRNoC network architecture which implements DDR datapaths to improve network throughput. Chapter 4 describes the design details of the FreewayNoC network architecture which enables incoming flits propagating an in-network straight hop to bypass allocation stage of the DDR router to reduce packet latency. Chapter 5 describes the design details of the HighwayNoC network architecture which extends allocation bypassing support offered in the FreewayNoC to also include flits entering and exiting through the local port, in order to further reduce packet latency. Chapter 6 describes the design details of the FastTrackNoC network architecture which enhances the HighwayNoC architecture by enabling incoming flits to bypass the ST stage as well as the allocation stage, in order to further improve packet latency. Chapter 7 presents the implementation, performance, power and energy efficiency results of the four DDR networks and compares them to relevant related work. Chapter 8 offers concluding remarks and possible directions for future work.

Chapter 2

Related Work

This Chapter presents an overview of techniques in the router architecture for increasing network throughput and reducing packet latency. The switching rate and minimum hop latency, define the limits for network throughput and packet latency, respectively. At the end of this Chapter, we report these two parameters for various state-of-the-art NoC routers measured in FO4 delays as reported in the published articles and confirmed by our post place and router (P&R) implementations.

2.1 Increasing NoC Throughput

There are three main approaches of improving throughput of single data rate (SDR) networks. They are: (i) use path diversity [35] and flow control to avoid congestion [36–38], (ii) improve matching quality of VC and switch allocators [18, 39, 40] and (iii) increase network switching rate by increasing network clock frequency. On one hand, the first two approaches improve the utilization per cycle of the network datapath resources to increase throughput and are orthogonal to the proposed DDR NoCs. On the other hand, the third approach enables the NoC datapath to propagate flits at a faster rate and increase network throughput. Previous NoC architectures have improved network clock frequency, and thereby the switching rate of their datapath, by increasing router pipeline depth, balancing the delay of different pipeline stages and simplifying or pipelining control (allocation) stages of a router.

A first step towards increasing clock frequency is to *pipeline router's datapath*. Typically, the datapath of a NoC router is divided into two stages, namely, the switch and the link traversal stages (ST and LT). Deeper pipelining of the datapath can further improve clock frequency at the cost of packet latency. One such example is Intel's full custom input buffered wormhole-switched NoC router which splits ST into two stages [9]. It achieves higher clock frequency by simplifying the datapath and the control stages and using a deeper pipeline with 6 stages per hop. Its simple datapath is only 32 bits wide and it has only two lanes (VCs) per input port to reduce the delay and complexity of the crossbar. It uses a crossbar of half the datapath width to save area, but operates it at DDR to maintain full throughput. Furthermore, it statically assigns VCs to packets and determines the route a packet will take in the network at the source node in order to avoid VC allocation and route computation, respectively, in the routers. This enables it to achieve a clock period of 15 FO4 delays and deliver

high throughput. However, this comes at a high cost of minimum hop latency of 90 FO4 delays per hop, as presented in Table 2.1.

In order to avoid the high hop latency overhead of many pipeline stages, a router's datapath is typically split into two stages, ST and LT. Although this offers reasonably high clock frequency, it places router's control in the critical path and leads to wasteful slack in the datapath stages [18, 20]. This creates an imbalance in the router's pipeline stages because of which the control of the router (and not its datapath) defines the network clock frequency and the rate at which flits are switched. This reduces the throughput potential of the network because of the slack introduced in the datapath stages. This slack can be reduced, and the network throughput increased, by either retiming the router's pipeline stages or pipelining and simplifying its control logic.

Time stealing (or *automatic retiming*), currently supported by CAD tools, can be used to balance the pipeline of a router. Mishra et al. used time stealing to boost routers performance [20]. In doing so, a router would have a clock period equal to the average of all router stages. Although time stealing improves baseline operating frequency and throughput by up to 25%, by reducing the slack present in the datapath, there is room for improvement because the clock period is still longer than than the datapath delays.

Next step towards increasing network clock frequency is to either pipeline or simplify the control (allocation) stages of a router. An SDR NoC which *pipelines switch allocation* (SA) into two stages and improves clock frequency is SCORPIO [13]. It also utilizes several other architectural techniques, which include *combined VC and switch allocation* [41], *control forwarding* (also called *lookahead signaling*) [42] and *allocation bypassing*, to improve network performance. Implemented on a 36-core chip, SCORPIO demonstrates low latency and short cycle time of 28 FO4 delays.

More recently, ShortPath [30] was proposed, the fastest previous SDR NoC architecture. ShortPath not only splits SA into two pipeline stages, but also simplifies it by using *allocation request queues* to reduce the number of simultaneous allocation request from each input port. It also pipelines its datapath into two stages (ST and LT). ShortPath employs a dynamic allocation bypassing mechanism, but as opposed to SCORPIO, without relying on speculation or control forwarding and it operates at a higher clock frequency compared to SCORPIO. Considering 2D mesh topology and 4 VCs, the critical path of ShortPath is shared by part of its control logic and ST and is estimated to be 25 FO4 delays, which is still about 20% slower than its ST delay. As a consequence, ShortPath offers about 20% lower throughput than what could be achieved if the critical path was on the ST stage.

Trying to gain more throughput by further simplifying allocation to improve network clock frequency does not offer significant benefit because deteriorating matching quality of simple allocators reduces datapath utilization at high traffic [24]. One way to move beyond this limitation is to operate some parts of the router's datapath, which have sufficient slack, at DDR to improve their throughput. In the past, *DDR datapaths* have been used in isolated parts of NoC routers (crossbars or links), primarily to reduce area and power cost of the datapath. For example, Intel's full custom NoC router [9], described above, used a single *DDR crossbar* to support two network lanes. Xu et al. built DDR Wave-Pipelined (DWP) links interconnecting asynchronously NoC router ports to reduce the number of link-wires [31]. A router converts the data to be transmitted through a DWP link from SDR to DDR format by sending odd bits on the positive level of the clock and even bits on the negative level of the clock. NMOS transmission gates are used for this purpose. A separate wire is used to transmit a

reference clock signal to the downstream router, along with the data. This clock signal is used by the semi-static double-edge-triggered-flip-flops (SDETFFs) located at the input of the downstream router to latch the incoming data at both rising and falling edges of the clock. Since this data is being transmitted in DDR mode, the transmitted clock signal needs to toggle at the rate of data transmission.

More recently, RapidLink was proposed which used *DDR links* in a NoC architecture [32], based on the consideration that links can operate at double the frequency of a router. RapidLink improves throughput and latency because of two underlying mechanisms. Firstly, it splits a 4-VC NoC in to two *parallel physical sub-networks*, of 2 VCs each, which share common links. In effect, RapidLink achieves the throughput of two sub-networks that have separate links at half the link cost [16, 43]. Secondly, consecutive sub-routers operate at different clock edges allowing them to perform LT in half a cycle and therefore reduce per hop latency. In Rapidlink, link contention between the two sub-networks is avoided, however, contention within a sub-network router is not addressed. On the contrary, our DDR NoC routers allow all parts of a NoC datapath (input VC buffers, input and output port multiplexers and links) to operate at DDR. This enables DDR NoCs to route at DDR flits of the same or of different packets (stored in VCs of the same or different input ports) rather than only flits of two different sub-networks. RapidLink requires links to have limited length to be twice as fast as the routers.

This last limitation of RapidLink was addressed by the authors in their recent design (henceforth called Rapidlink2) where they proposed to pipeline the network links using *Dual Stream (DS) elastic buffers* with support for VCs and consider them as part of VC buffers [17]. Thereby, links are no longer in the critical path. However, according to our experiments the energy cost for transmitting a bit (for 128 bits wide flit) over a link that is pipelined using one or two sets of DS elastic buffers with support for 4 VCs increases versus a non-pipelined one by 1.8× and 2.5×, respectively. This is because a register is needed per VC, along with a 4-to-1 mux and an arbiter for each pipeline stage. In any case, DDR NoC routers with similar datapath modifications –router split in two sub-networks and replicated, rather than pipelined, links– can achieve better throughput than RapidLink2, as presented in the Section 7.3.3.

The DDR NoC architectures we propose in this thesis utilize the slack present in the router's datapath to improve network throughput, by operating the complete datapath at DDR. Then, the control has an entire, longer than ShortPath, cycle available to make allocation decisions. The clock period of a DDR NoC is 42 FO4 delays, twice its ST delay, and the switching rate of its datapath is half of that, i.e., 21 FO4 delays. As a consequence, the control logic is not in the critical path of a DDR NoC router and the ST delay defines the switching rate, which is about 20% higher than the fastest SDR NoC router, ShortPath.

2.2 Reducing NoC Packet Latency

There is a plethora of designs that aim at minimizing packet latency of SDR NoCs. Other than the strategies presented in the previous Section to increase network clock frequency (which will also reduce packet latency), there are four main approaches of reducing packet latency. They are: (i) deploying NoCs with low diameter topologies (i.e. with low maximum distance to other nodes), (ii) transferring flits across multiple hops or tiles in a single cycle, (iii) using routers with simpler datapaths to increase

clock frequency and (iv) reducing the number of pipeline stages a flit must traverse per hop. These approaches are further discussed below.

In the past, various networks have exploited a lower, relative to the router, link delay to transfer flits across multiple hops. This has been supported by *richer topologies* with longer links to non-neighboring routers, e.g. 2D-torus, flattened butterfly topology [15, 44], multidrop express channels (MECS) [22], and express links [4, 27, 45]. Such low diameter topologies require high radix routers and long inter-router links spanning multiple hops. But, routers with high radix require complex allocators and larger crossbar switches which reduces their clock frequency and throughput. Long inter-router links, if not pipelined, increase LT delay and reduce network clock frequency. Moreover, such topological changes can be considered orthogonal to the techniques incorporated by the proposed DDR NoC router architectures to reduce packet latency.

Another way to reduce packet latency is to transfer flits across multiple hops in a single cycle, as implemented in *SMART NoC* [26]. SMART NoC exploits low link delay to transfer flits across multiple hops in a single cycle using intra-router bypass paths across input VC buffers. Although such *multi-hop links* enable SMART networks to offer minimum latency per hop close to $ST + LT$, fitting multiple hops in a single cycle slows down network clock frequency and reduces offered throughput. It also requires complex allocators with inputs from neighboring routers for multi-hop setup requests.

One way to increase NoC clock frequency while also reducing its area is to simplify its datapath. More precisely, *bufferless NoCs* remove input VCs and FIFO buffers from routers to simplify datapath and control logic and increase clock frequency [46, 47]. In the best case (if control logic is not in the critical path), bufferless network can offer minimum hop latency of $ST + LT$, with ST delay equal to a $(P-1):1$ multiplexer compared to $((P-1) \times V \times B):1$ multiplexer for a network with VC based flow control with P ports per router, V VCs per port and B flit registers per VC buffer. However, bufferless networks offer lower throughput compared to buffered networks because of deflection routing [47–49] or dropping flits at high load, making them unsuitable for applications with high throughput requirements.

More recently proposed *Routerless NoC* removes the router from NoC and uses on-chip wiring resources to create loops (rings) to connect its nodes. It can operate at a very high clock frequency and offer minimum hop latency equal to the delay of the inter-router link plus a 2:1 multiplexer. Compared to the proposed DDR NoC architecture which offers bypassing of SA and ST stages (called the FastTrackNoC), the Routerless NoC minimum hop latency is one 2:1 multiplexer delay less. However, the Routerless NoC has several significant disadvantages. Using fixed loops limits the routing options. The Routerless NoC alleviates this problem by using n overlapping loops in an $n \times n$ network. However, this requires partitioning of the wire resources to n paths that are n times narrower substantially limiting throughput compared to conventional 2D-mesh networks with the equal wire resources. Moreover, Routerless NoC requires large buffers at their network interface to fit an entire packet in order to resolve conflicts. In turn, Routerless NoCs can offer slightly lower latency than the FastTrackNoC described in Chapter 6, however, their fixed loops either limit routing options and thus increase latency, or call for multiple overlapping loops which limit throughput considering equal wire resources.

Packet latency can also be reduced by reducing the number of pipeline stages a flit needs to go through to traverse a hop. Typically, NoC routers require flits to go through five stages per hop, which are: route computation (RC), VC allocation (VA), switch

allocation (SA), switch traversal (ST) and link traversal (LT). *Pre-configured routing* speculatively forwards incoming flits in preferred output directions without waiting for the result of allocation stage, saving one cycle per hop [50]. However, it has to tolerate misrouted, eagerly forwarded, dead flits. Similarly, *XOR-based crossbars* also allow switching to be performed without waiting for the arbitration result but require router datapath to be modified to encode and decode flits [51]. Less wasteful approaches have been the backbone of current state-of-the-art NoC routers. *Lookahead routing* allows to perform VC and switch allocation immediately after a header flit arrives as the route of the flit has already been computed by the upstream router [24]. *Combined allocation* [41] allows VA and SA to be performed in parallel and saves a cycle in the router pipeline [25]. *Control forwarding* enables allocation to start before the incoming flit is received [13, 29]. It is inspired by off-chip network designs [42] and uses a separate, narrow link to forward the control of a flit a cycle ahead of the data enabling the downstream router to process it one cycle earlier [13, 29]. Finally, *pipeline stage bypassing* allows incoming flits which do not encounter any contention in a router to bypass some or all control stages of that router and proceed faster to the switch traversal stage [13, 30, 52].

ShortPath [30], described in the previous Section, utilizes, among others, lookahead routing and allocation bypassing (of VA and SA stages) to offer minimum hop latency of $ST + LT$ stages i.e., 50 FO4 delays, as presented in Table 2.1. Since the clock period of ShortPath (25 FO4 delays) is defined by the router's control delays and is longer than its ST delays, there is potential to further reduce minimum latency by eliminating the slack present in the datapath stages (ST and LT) and operating it at a faster rate.

On the contrary, the DDR NoC architectures described in this thesis have a clock period defined by the datapath (ST) delays and they also implement lookahead routing, control forwarding, combined allocation and pipeline bypassing of the allocation stage to offer minimum hop latency of $ST + LT$ delays, i.e., 42 FO4 delays. It is important to note that, although, minimum hop latency for ShortPath and proposed DDR NoCs (which do not offer ST bypassing) is $ST + LT$, DDR NoCs offer lower latency because of their faster switching rate, as presented in Table 2.1. Furthermore, our best proposed DDR NoC architecture, called the FastTrackNoC, enhances the bypassing capabilities of previous networks and allows incoming flits to bypass ST as well as SA stages, when required conditions are met, and directly initiate LT in half a clock cycle. Consequently, it can offer minimum hop latency of only 21 FO4 delays.

The challenge of supporting pipeline bypassing in a DDR router is that it adds complexity to the datapath of the router and this in turn affects throughput by reducing network clock frequency. More precisely, an incoming flit candidate for bypassing would need to compete with other concurrently arriving candidate flits, so checking whether the path is free adds logic to the ST and LT stages. On the other hand, the main performance advantage of DDR router relies on the fact that its critical path, which determines the rate at which flits are routed, is defined by the ST (or LT) delay without any additional control overhead. So, our proposed DDR NoC architectures need to preserve this throughput advantage by not offering allocation bypassing options to in-network turning flits (i.e., xy and yx turns), while, only offering ST bypassing to flits propagating an in-network straight hop (i.e. north to south, east to west and vice versa) from the head of a particular input VC. This significantly reduces contention among flits competing to bypass SA or ST stages at the same time as well as the overhead of bypass paths needed to support bypassing. The FreewayNoC and the HighwayNoC architectures, presented in Chapter 4 and 5, respectively, implement different levels of

Table 2.1: State-of-the-art NoC router architectures.

Network	Data rate	Pipeline bypass	No. of pl. stages	Switching rate	Min. hop latency
Intel [9]	(DDR)		5+LT	1/15 FO4	90 FO4
SCORPIO [13]	SDR	(SA)	3+LT	1/28 FO4	56 FO4
ShortPath [30]	SDR	VA, SA	3+LT	1/25 FO4	50 FO4
DDRNoC	DDR		2+LT	1/21 FO4	84 FO4
FreewayNoC	DDR	(SA)	2+LT	1/21 FO4	42 FO4
HighwayNoC	DDR	SA	2+LT	1/21 FO4	42 FO4
FastTrackNoC	DDR	SA, ST	2+LT	1/21 FO4	21 FO4

allocation bypassing to reduce packet latency. Finally, the FastTrackNoC architecture, presented in Chapter 6, offers both SA and ST bypassing to incoming flits in order to reduce packet latency.

2.3 Summary

Table 2.1 summarizes the main characteristics of competing designs including performance estimations measured in FO4 delays either taken from the respective papers [9] or based on post place and route results on 28nm technology, as explained in Chapter 7. The aforementioned Intel design offers a very fast router with oversimplified control and datapath which achieves the fastest rate of routing flits (1/15 FO4), however without pipeline bypassing it suffers long hop latency of 90 FO4 delays [9]. SCORPIO is a 3-stage SDR router that offers speculative allocation, control forwarding, and some pipeline bypassing support, and exhibits a cycle time of 28 FO4 delays [13]. It offers low hop latency of 56 FO4 delays achieved by bypassing router pipeline stages. ShortPath is a faster 3-stage SDR router with better pipeline bypassing and shorter critical path (25 FO4 delays) than SCORPIO [30]. It offers a minimum hop latency of 50 FO4 delays. DDRNoC, presented in this thesis, is the first fully DDR router that increases the rate of routing flits to one flit per 21 FO4 delays (every half a cycle) offering higher throughput. It offers pre-computed routing, speculative SA, and control forwarding, however it has no pipeline bypassing support so it suffers high packet latency. FreewayNoC improves on the DDRNoC offering limited in-network pipeline bypassing to non-turning flits, improving packet latency and HighwayNoC adds bypassing support for flits that enter or exit the network. Finally, FastTrackNoC is applied to a DDR router so it maintains the fast switching rates and reduces the minimum router latency to half (21 FO4 delays) employing ST bypassing.

Chapter 3

The DDRNoC Architecture

This Chapter describes the first DDR NoC architecture, called the DDRNoC. The DDRNoC is based on the observation that NoC clock frequency is defined by its control logic, while its datapath contains slack which reduces maximum achievable network throughput. The DDRNoC is an on-chip interconnect composed of routers that have a double-pumped datapath. As opposed to a conventional SDR NoC router, the critical path of a DDRNoC router is on its switch and link traversal rather than on the control. This allows packets to be routed at a faster rate increasing network throughput. Without loss of generality, our DDRNoC design considers a 2D-mesh network, with look-ahead XY-routing, composed of routers with virtual channels and credit-based flow control.

The top-level view of the DDRNoC router is shown in Figure 3.1. The datapath is composed of two stages: Switch Traversal (ST) and Link Traversal (LT). Each stage is able to handle two flits per cycle, one at the high phase and one at the low phase of the clock. There are three main control blocks in the router: Virtual Channel Allocation (VA), Speculative Switch Allocation (SA) and Next-Next-Route-Computation (N^2RC) which are explained in detail below.

The rest of this Chapter is organized as follows. In the next Section, we describe the DDR datapath of the DDRNoC. In Section 3.2, we describe the timing details of flits and forwarded control information propagating through DDRNoC routers, illustrated using a timing example. In Section 3.3 we analyze the zero-load latency of DDRNoC. In Section 3.4, we describe the control modules of of a DDRNoC router. In Section 3.5, we discuss some of our design decisions. Finally, in Section 3.6, we summarize the main aspects of the DDRNoC architecture.

3.1 Router Datapath

The input port of a DDRNoC router is able to receive two flits per cycle, one at each phase of the clock. As shown in Figure 3.2, the VC buffers are composed of registers that are selectively triggered either at the rising or at the falling edge of the clock to store a flit arriving at the low or the high phase, respectively. This enables two flits per cycle to be enqueued in a single VC buffer because they are stored in different registers of the buffer. Dequeuing two flits per cycle from a single VC buffer is enabled by the *DDR_VC_en* signal, which allows two different VC registers to be selected during two consecutive clock phases. Similarly, dequeuing two flits from different VCs within a

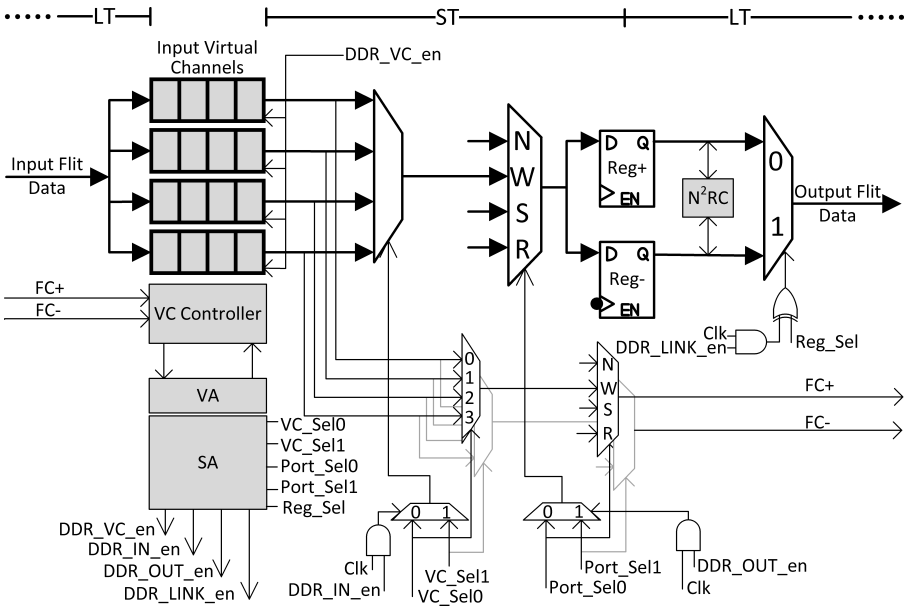


Figure 3.1: The DDRNoC router architecture.

cycle is enabled by the *DDR_IN_en* signal, which is used by the input port multiplexer to implement two input arbitration decisions (VC_Sel0 and VC_Sel1). Besides the flit data, the input port receives once every cycle early forwarded control information for the (up to two) received flits ($FC+$, $FC-$). As explained later, this information refers to flits that arrive a cycle later, thus it is called $forwarded\ control$.

Besides the input port multiplexer, the ST stage includes the output port multiplexer, which applies up-to two output arbitration decisions per cycle ($Port\_Sel0$, $Port\_Sel1$) using the *DDR_OUT_en* signal. This allows two different input ports to send a flit to the same output port during the high or the low phase of a clock cycle.

A positive edge triggered output register ($Reg+$) and a negative edge triggered one ($Reg-$) are used to store the flits switched in the low and the high phase of a cycle, respectively. Subsequently, a multiplexer selects one of the two registers to send a flit through the link. Using the *DDR_LINK_en* signal, this multiplexer allows LT of two flits in a cycle. For packet header flits, the N^2RC module computes routing information for two-hops ahead in half a cycle and the result (2-bits) is embedded in the header flit data before LT.

In general, DDR mode is selected in two cases. Firstly, when flits of multiple packets in different VCs compete for the same datapath part (input port multiplexer, output port multiplexer, or link). Secondly, when multiple flits of a single packet are available in a VC buffer and their requested datapath is not allocated to any other packet during the same cycle. Figure 3.3, illustrates more precisely all alternative uses of the DDRNoC datapath. In Figure 3.3a, two flits in different VCs of the same input port are sent to the same output port in DDR (*DDR_IN_en* and *DDR_LINK_en* enabled). Figure 3.3b shows two flits from different input ports sent to the same output port in DDR (*DDR_OUT_en* and *DDR_LINK_en* enabled). Two flits from different VCs of the same input in DDR going to different outputs are illustrated in Figure 3.3c (*DDR_IN_en* is enabled). Figure 3.3d shows two flits of the same packet (same input

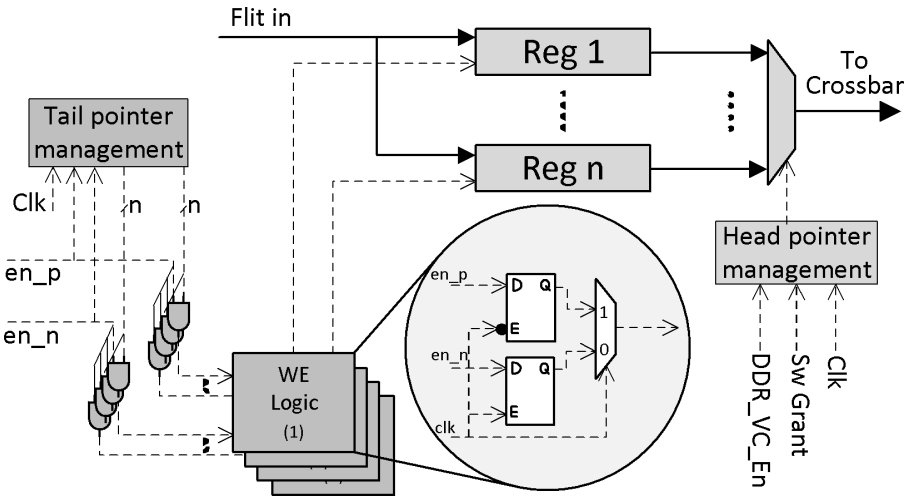


Figure 3.2: DDRNoC VC buffers. Signals en_n and en_p signals indicate whether there is a valid flit in the high and low clock phase. Tail and header pointers as well as a write enable signal (per register) are updated based on these two signals.

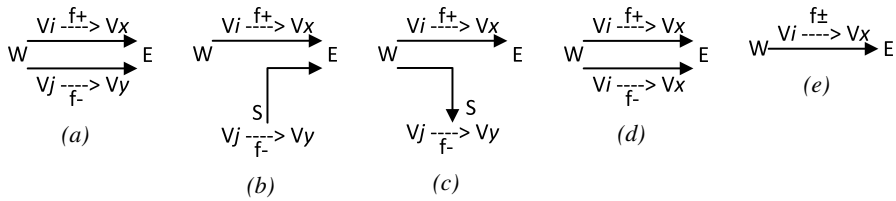


Figure 3.3: Alternative ways to propagate a flit through the DDRNoC router datapath.

VC) in DDR (*DDR_VC_en* and *DDR_LINK_en* enabled). Finally, a flit sent in SDR is depicted in Figure 3.3e.

In parallel to the ST of (up to) two flits, their control information ($FC+$ and $FC-$) is forwarded separately. As explained next, both $FC+$ and $FC-$ are switched during the first half of a cycle and traverse the link during the second half. This is possible as the minimum ST and LT delay in a DDRNoC router is half a cycle.

3.2 Timing

Despite operating at SDR or DDR, a flit always spends half a clock cycle in each datapath stage (ST or LT), separated by another half a clock cycle which is used for route computation. In DDR mode, a flit utilizes either the high or the low phase of the clock. We call a flit using the high clock phase $flit^+$ and one using the low phase $flit^-$. A $flit^+$ is switched during the first half of a cycle and registered by the negative edge triggered output register $Reg-$, it uses again the high phase of the next cycle for LT and is stored at the input VC of the downstream router in the next negative clock edge. Similarly, a $flit^-$ uses for both ST and LT two consecutive low clock phases, it is registered in the positive edge triggered output register $Reg+$ and is stored in a VC

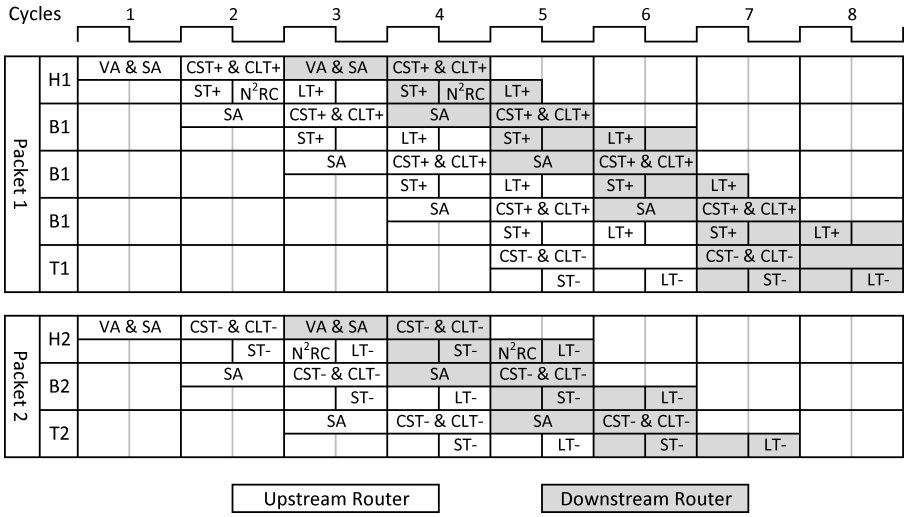


Figure 3.4: Timing of two packets traversing two hops through the same input and output ports (different VCs) of two routers.

buffer of the downstream router in the next positive clock edge. Finally, a flit switched at SDR ($flit^{\pm}$) propagates in a manner similar to $flit^{+}$.

Besides data, a flit usually carries additional control bits that indicate flit type, allocated VC, or the computed next route. In a baseline SDR router, these control bits pass through ST and LT together with the data of a flit. On the contrary, in a DDRNoC router, the control bits of flit(s) travel ahead of the data so as to enable the SA in the downstream router a cycle earlier and reduce packet latency. During ST of flits, $flit^{+}$ and $flit^{-}$, their control bits $FC+$ and $FC-$ perform both ST and LT (half a cycle each). In effect, during the cycle $flit^{+}$ and $flit^{-}$ perform LT, their control bits are already in the downstream router and are considered in the SA and VA, saving one cycle. We call the above technique *control forwarding*. Control forwarding is possible in our router architecture for two reasons: firstly, due to the fact that switch and link traversal have a latency of half a cycle, and secondly, because a flit performs ST and LT in two consecutive cycles (instead of a single cycle). A flit does not enter a new datapath stage in the middle of a cycle to avoid misalignment with the SA in the downstream router, which anyway starts at the positive clock edge and takes an entire cycle to complete. Although the data of a flit are routed slower than they potentially could, sending the control of the flits ahead, enables the downstream router to start SA a cycle earlier recovering the wasted time.

In order to perform the NRC in parallel with the VA and SA in a router with control forwarding, the entire destination address of a packet should be forwarded together with the rest of the control bits (earlier than the actual header flit). That would be quite costly as it would require for a 16×16 2D-mesh, 16 more link wires for the two flits in flight. To avoid this, we perform the NRC in the output of the upstream router, effectively, performing a routing computation for two hops ahead (Next-Next Route Computation: N^2RC). A flit takes half a cycle to traverse a link, during either the high or low clock phase, but it is sitting idle in an output register ($Reg+$, $Reg-$) for the other half of the cycle. N^2RC is performed during this idle half cycle. The N^2RC module is placed after the output registers and before the link multiplexer. When a

head flit is registered in the output after ST, N^2RC is performed for half a cycle, the result of the N^2RC overwrites the (two) respective header bits and is sent to the link multiplexer before LT. Note that the two output registers are triggered in different clock edges, so the same N^2RC module can be used by one of them in each half of the cycle.

For the header flit, SA speculates on the VA allocation result. Thereby, the VA can be performed in parallel with the SA. The header flit zero-load latency is then reduced by one cycle when the speculation is successful.

Figure 3.4 illustrates the timing of two DDRNoC routers, putting all the above together. In the example, a packet of five flits and a packet of three flits traverse the same two routers. In the first router, VA and SA is performed during the first cycle for the head flits of the two packets. Subsequently, ST is performed for the header flits (ST^+ and ST^-), each using half of a cycle. In parallel, during cycle 2, control information for the head flits is traversing the switch (CST^+ and CST^- for $flit^+$ and $flit^-$, respectively) and the link (CLT^+ and CLT^-). H1 traverses the switch in the high phase of clock 2 and is stored in the output register $Reg-$ at the falling edge of cycle 2 enabling N^2RC to be performed in the second half of cycle 2. Similarly, H2 traverses the switch in the low phase of clock 2, it is stored in the output register $Reg+$ at the rising edge of cycle 3 enabling N^2RC to be performed in the first half of cycle 3. LT is then performed by H1 and H2 after their N^2RC during the first and second half of cycle 3, respectively. In parallel during the same cycle (cycle 3), SA and VA are performed in the downstream router for the two header flits, as their control information has arrived a cycle earlier. The subsequent flits of the two packets follow, sharing the datapath of the two routers, having one cycle latency per ST and LT stage and throughput of two flits per cycle. Note, that packet 2 is two flits shorter than packet 1 and therefore the last two flits of packet 1 are routed in DDR mode through the two routers, similar to the example of Figure 3.3d.

3.3 Zero-Load Latency Analysis

The zero-load latency (ZLL) in a DDRNoC[1] is equal to one cycle for the first SA plus two cycles per hop for the head flit and an additional half a cycle for each of the remaining flits[2]:

$$ZLL_{DDRNoC} = 1 + 2 \times hops + (N-2)/2 \; cycles \tag{3.1}$$

where, *hops* is the number of hops traversed by the packet, and N the number of flits per packet.

Moreover, the ShortPath router which uses dynamic pipeline bypassing to reduce pipeline stages to 2 (VA, SA and ST in one cycle and LT in the other) has a zero load latency of:

$$ZLL_{ShortPath} = 2 \times hops + N - 1 \; cycles \tag{3.2}$$

As mentioned in the introduction and confirmed in our evaluation, the ST/LT stages in a router can be clocked about 18% faster than ShortPath. The ST or LT delay

[1] We consider that the VC buffer size is sufficient to support the credit round-trip time, as explained in 3.4.3.

[2] In zero load, a single packet can send its flits in DDR having serialization latency of half a cycle per flit (Figure 3.3d). In case of contention, the serialization latency would increase at least to one cycle per flit.

is then 83% the delay of the ShortPath NoCs. Considering that the DDRNoC can have a clock period of 2× the ST (or LT) delay, then one DDRNoC cycle is equal to about 1.67 ShortPath cycles. So, for a packet with 3 flits, propagating between 2 and 32 hops, zero-load latency for the DDRNoC is about 53-66% longer than ShortPath.

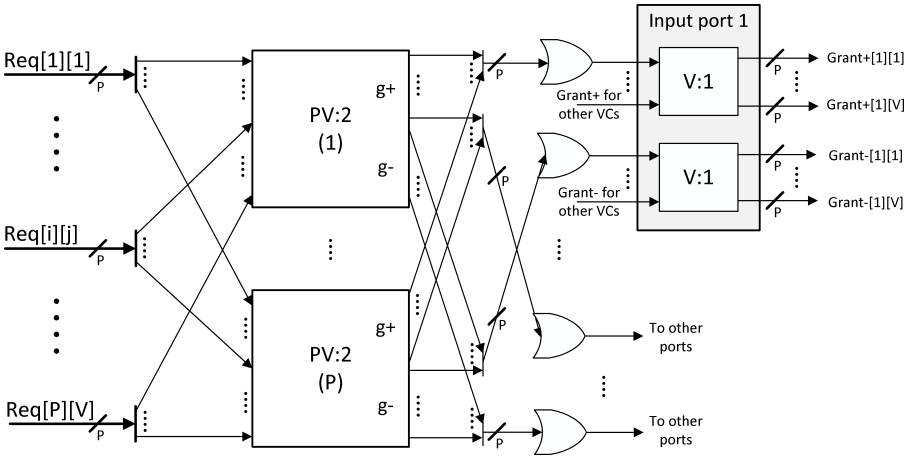
3.4 DDRNoC Control

3.4.1 Virtual Channel Allocation

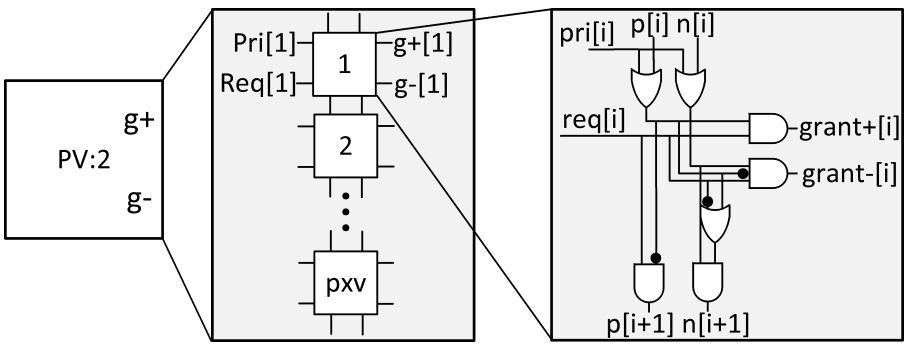
Despite the increasing load of the VA in a DDRNoC router, we choose to use the SDR baseline VA unmodified. That is a VA with round-robin priority-based output-first separable allocator as described by Becker and Dally [39]. For a router with P output ports and V VCs per port, the VA uses $PV:1$ and $V:1$ arbiters for the output and input arbitration, respectively, and takes one cycle to complete allocation. In DDR mode, up to two head flits per cycle may arrive at an input port of a router, which (in the worst case) would result in ten new VA requests per cycle, compared to five in the baseline. Although this may lower the VA matching quality, we did not observe any significant performance drawbacks in our experiments for packet sizes based on application driven workloads.

3.4.2 Switch Allocation

The DDRNoC routers, as well as the baseline SDR ones, use a single-cycle speculative SA to resolve contention among flits requesting the same input and output ports of the crossbar. The speculative SA gives higher priority to requests which have already been allocated a downstream VC. The DDRNoC SA, shown in Figure 3.5a is a modified version of the output-first separable allocator described in [39]. It makes two input and output arbitration decisions every cycle, one for the high ($g^+[i]$) and one for the low ($g^-[i]$) phase of the clock. The output arbiter for a single output port, shown in Figure 3.5b, accepts $P\times$V arbitration requests and a one-hot priority vector (which updates in round-robin) to generate two grant signals for each input request. At most one of these two grant signals would be asserted per input request. Moreover, at most one g^+ and one g^- signal would be asserted as a result of each output port arbitration. The second stage of the SA performs (input) arbitration among the VCs of an input port, which after the output arbitration might have two or more of its VCs granted access to the switch on the same half of the cycle. Two V:1 input arbiters per port are required, one to arbitrate among grants received for the high half of the cycle and the other among the grants received for the low half. After input arbitration a maximum of two grants (one per clock edge, $grant+$ and $grant-$) are asserted for each input port. The DDRNoC SA described so far is not yet able to allow two flits of the same packet to be switched in DDR mode. This is performed using additional logic appended to the above SA. In particular, based on the above grant decisions, a single packet, with an allocated downstream VC, will be allowed to send two flits through the switch in a cycle when all following conditions are met: (i) that packet has received a SA grant, (ii) more than one flits of the packet are available in the input VC, (iii) there are enough credits for them, (iv) no other packet has been granted the same input or output of the switch. After SA, the grant+ and grant- signals are registered and the DDR enable signals are generated. These are the signals that control the DDR mode



(a) DDRNoC dual grant Switch Allocator.



(b) PV:2 output arbiter.

Figure 3.5: Block diagram of the DDRNoC Switch Allocator.

in each VC buffer (DDR_VC_en), input port multiplexer (DDR_IN_en), output port multiplexer (DDR_OUT_en), and output (DDR_LINK_en).

3.4.3 Flow Control and Minimum Buffer Size

The DDRNoC uses credit based flow control. Since flits from two different input VCs can be granted access to the switch in a single cycle, credits for up to two different VCs need to be transmitted to the upstream router simultaneously. This is implemented using one wire per VC for sending credits[3]. Moreover, a single VC can also forward two flits of a packet in a single cycle. In order to send back two credits to a single VC in a cycle, one additional wire per port is used to inform the upstream router that the credit counter of the VC receiving credit(s) should be incremented by 1 or 2. A credit consumed after switch grant, can successfully be received back in a minimum of 4 cycles after SA of the flit in the downstream router. This means that using 4

[3]This is sufficient for 4 VCs per port considered in our implementation, but as the number of VCs per port increases, it becomes more scalable to send the ids of the two granted VCs, instead of a bitmask.

registers per VC would be sufficient to cover the credit-round-trip-time (t_{crt}). This is true however only if a single VC sends one flit per cycle. In case a single VC sends two flits at DDR (Fig. 3.3d) then it occupies 2 downstream buffers per cycle increasing the the minimum VC buffer size to 8 (although the t_{crt} remains the same). In comparison, the 3-stage baseline NoC has a t_{crt} of 5 cycles and uses VC buffers of 5 flits.

3.5 Discussion

We discuss below some of the DDRNoC design decisions and other alternatives. As opposed to a baseline SDR router, the critical path of DDRNoC router is moved to the datapath ($2 \times max(ST, LT)$) allowing the DDRNoC control to have significant slack (about 30% of the cycle). In our current design, this slack in the DDRNoC control is largely unexploited. Although the DDRNoC SA is more complex than the baseline SA, it is almost as slow as the VA block, which is the same for both the baseline and the DDRNoC. Consequently, more advanced SA and VA modules can be explored in the DDRNoC without affecting its cycle time. Another observation is that instead of operating at DDR using both clock edges, similar throughput could be achieved by a NoC that has a clock twice as fast as the DDRNoC clock and makes allocation decisions every two cycles using a pipelined SA. Such design would have many similarities with the proposed DDRNoC although it might be slightly less challenging to implement. However, offering two cycles for the allocation decisions, even if the allocators were pipelined, would affect the efficiency of the speculative SA. Moreover, the clock distribution power would be double than that of the DDRNoC and hence the total NoC power would be about 20% higher. Even compared to the baseline SDR NoC, the total power consumption of such a network would be expected to be about 9% higher due to a 45% faster clock. Moreover, packets in such a router would spend 4 cycles per hop because of increased router pipeline depth, which would require 6 registers per VC buffer to cover the t_{crt} of 6 cycles.

3.6 Summary

The DDRNoC architecture, introduced in this Chapter, uses routers with double-pumped datapath. It is based on the observation that conventional SDR 2D-mesh NoC routers have significant slack in their datapath stages. DDRNoC uses this slack by allowing two flits to share the same datapath within a cycle at DDR. It also employs control forwarding and speculative SA to reduce packet latency, which compared to SDR routers with pipeline bypassing still suffers at low injection rates.

Chapter 4

The FreewayNoC Architecture

The DDRNoC architecture presented in the previous Chapter operated its datapath at DDR in order to improve network throughput. It is based on the observation that clock frequency of previous SDR networks is defined by its control, while slack in the router's datapath reduces achievable network throughput. By operating the datapath at DDR, the DDRNoC is able to remove the slack present in the datapath stages, route flits at a rate defined solely by datapath delays and improve network throughput. However, the DDRNoC offers higher latency to packets compared to an SDR NoC. This is because DDRNoC operates at a lower clock frequency than an SDR network.

One simple way in which NoCs in the past have improved packet latency is by allowing incoming flits to bypass allocation stage, in the absence of contention, and directly initiate ST, followed by LT. This Chapter presents the FreewayNoC architecture, an enhancement of the DDRNoC architecture, which offers allocation bypassing capabilities. As opposed to the DDRNoC, FreewayNoC allows incoming flits to directly traverse the switch and link without undergoing an allocation step when their way to the output port is free, henceforth referred to as allocation bypassing (AB). FreewayNoC exploits a control forwarding (lookahead signaling) mechanism, similar to the one utilized in the DDRNoC, which allows the control of flits to precede the flit data, enabling allocation a cycle earlier; in our design control forwarding is also used for AB checks as well as for next-route computation (NRC). FreewayNoC uses a virtual channel selection (VS) mechanism at the output ports, rather than VC allocation in the DDRNoC, to better accommodate AB. It further restricts AB to flits propagating an in-network straight hop in order to minimize the overheads of bypassing logic. FreewayNoC achieves to not add any complexity to routers datapath preserving the DDR packet rate, while reducing packet latency. Without loss of generality, our FreewayNoC design considers a 2D-mesh network, with lookahead XY-routing, composed of routers with virtual-channels and credit-based flow control.

The top-level view of the FreewayNoC router is shown in Figure 4.1. The datapath is composed of two stages: Switch Traversal (ST) and Link Traversal (LT), separated by the input VC buffers and the two output registers (one for each clock edge multiplexed before the link). In addition, the FreewayNoC router has the following control blocks: a combined allocator composed of the Virtual Channel Select (VS) and a Switch Allocation (SA), Next-Route-Computation (NRC) and AB check logic at the input and output ports.

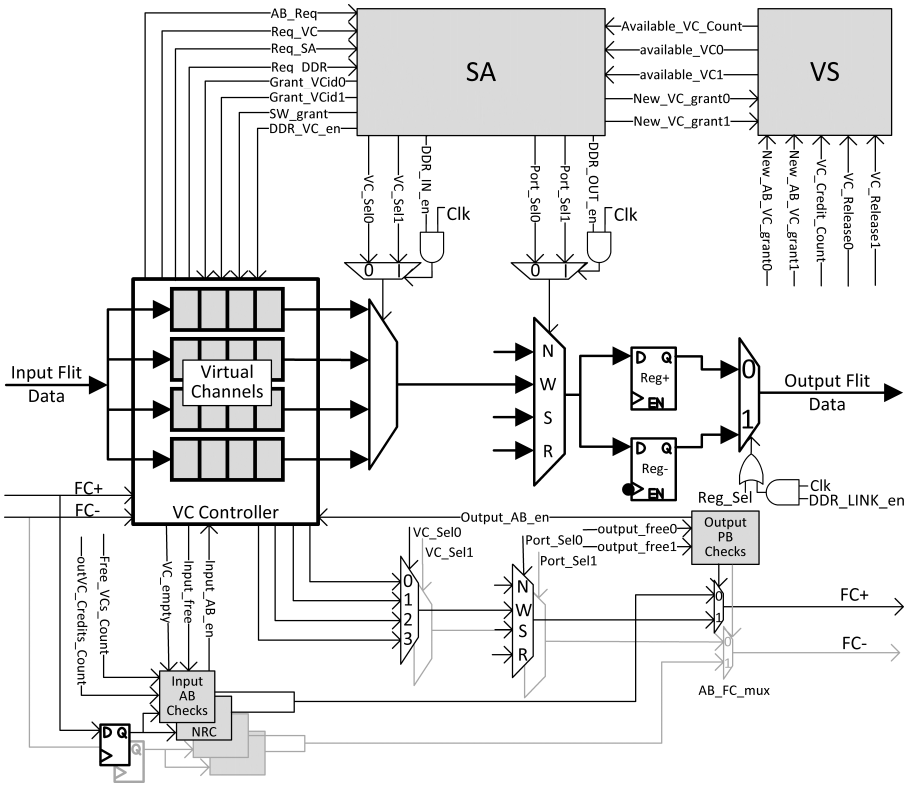


Figure 4.1: The FreewayNoC router architecture.

The rest of this Chapter is organized as follows. In the next Section, we describe the DDR datapath of the data flit, as well as the datapath of the forwarded control signals in a FreewayNoC router. In Section 4.2, we describe the timing details of flits and forwarded control information propagating through FreewayNoC routers, illustrated using a timing example. In Section 4.3 we analyze the zero-load latency of FreewayNoC. In Section 4.4, we describe the control modules of of a FreewayNoC router. Finally, in Section 4.5, we summarize the main aspects of the FreewayNoC architecture.

4.1 Router Datapath

The FreewayNoC router datapath has two parts: the datapath of the data flits and the datapath of the control bits forwarded separately to support control forwarding. Each of them is described separately below.

4.1.1 DDR Flit Datapath

The DDR flit datapath of a FreewayNoC router, illustrated in Figure 4.1, is similar to the DDRNoC router datapath described in Section 3.1. It also supports two flits per cycle per port, one at each phase of the clock and offers the same alternative ways to propagate flits at DDR and SDR, as presented in Figure 3.3. However, the control and

the timing of the flits differ in the FreewayNoC router compared to the DDRNoC, both when flits are routed regularly as well as in AB. The relative timings of switch and link traversal of flits and their control information is discussed in detail in Section 4.2

4.1.2 Datapath of Forwarded Control signals

The FreewayNoC also forwards control information of the two propagating flits to the downstream router one cycle ahead. The Forwarded Control (FC) uses a separate path (crossbar and link wires) to perform control switch traversal (CST) and control link traversal (CLT) in a single cycle as depicted in the bottom part of Figure 4.1. The FC bits carry the type of the flit (head, body, tail or a single flit packet), the id of the assigned downstream VC, the next route of the flits and the destination address. Moreover, there is a separate path for FC signals, which bypasses the regular CST connecting each input port with its straight output port (N to S, E to W and vice versa), except for the local port. This path is used to carry the FC signals of a bypassing flit to the output port through the *AB_FC_mux* and send them for CLT instead of the regular CST signals. Two sets of FC signals exist, one for the flit propagating in the high phase of the next cycle (*FC+*) and one for the flit propagating in the low phase (*FC-*), each set of FC signals uses a separate CST, CLT and bypass path.

The AB logic checks the respective input and output port availability, as well as for available VC and credits, to determine whether bypassing is possible. This is performed in parallel to the regular CST shown in Figure 4.1. Then, depending on the AB check, a 2:1 multiplexer is used to send the bypassed FC to CLT instead of the regular FC. This 2:1 multiplexer is the only addition to the FC path delay required for AB. Still, CST and CLT are not in the critical path because, as opposed to data ST and LT, they do not need a register between them and the FC path is substantially narrower than the datapath requiring a smaller crossbar.

4.2 Timing Example

The timing of a FreewayNoC router is described using an example of a five-flit packet traversing three routers, illustrated in Figure 4.2. In the source and destination router, the packet flits take a turn as they move from and to a local port, respectively. In our example, we consider that all routers are aligned and therefore flits passing through the second router go straight and have the opportunity for AB. Note that the LT and ST for flits that traverse the link in the high clock phase are marked in the example with “+”, and for flits that traverse the link in the low clock phase are marked with “-”.

In cycle zero, router-1 receives the forwarded control (FC) of the two first flits (H, B1) which carries flit types, assigned VC, next route and destination. The packet comes from the local port, so it takes a tur n and therefore there is no option for bypassing. Then, flits enter the allocation in cycle 1 using the next route information. In the same cycle. the head flit enables the NRC computation as well as the VC selection, which provides an output VC. In parallel, the data flits of H and B1 arrive and get stored in the input VC buffer in the first and second half of cycle 1, respectively. Considering that there are no other competing packets, both H and B1 flits get a grant for the switch at the end of cycle 1. Then, during cycle 2 the control information of the two flits can be forwarded to router 2 (CST & CLT). In the second half of cycle 2 the H flit traverses the switch of router-1 (ST+) and is registered in *Reg+* before

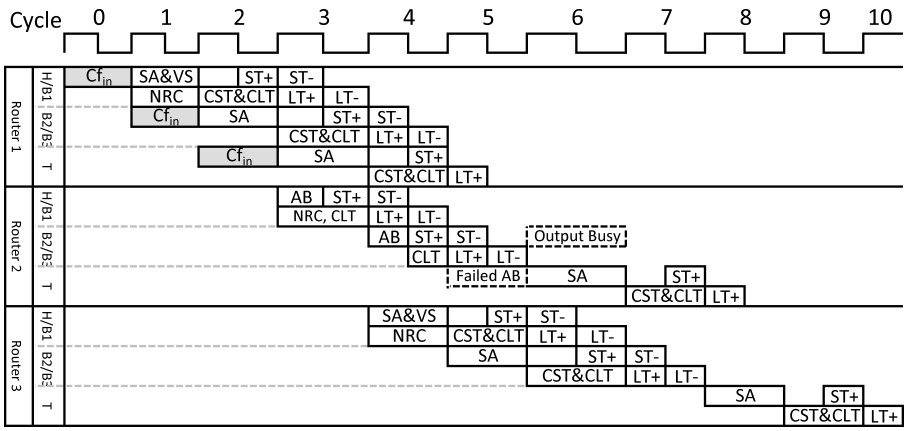


Figure 4.2: Timing diagram showing the flow of a five flit packet through three router of the FreewayNoC.

it performs LT in the first half of cycle 3 (LT+). B1 flit follows half a cycle later. It performs ST and LT in the first and second half of cycle 3, respectively (ST-, LT-). The remaining flits are routed through router-1 in a similar way.

Following again H and B1 flits, their control information arrives in router-2 at the end of cycle 2, enabling AB or allocation in cycle 3. AB check is positive, therefore the flits are not considered for allocation. As the first flit is a header, NRC and VC select (VS) are performed, too. NRC, VS, and AB check are performed in parallel to form the control information of the bypassing flits which will bypass the regular CST and traverse the link (CLT) to reach router-3 at the end of cycle 3. Bypassing flit H arrives in router-2 after the first half of cycle 3 and subsequently goes through ST and LT in the second half of cycle 3 and the first half of cycle 4. Bypassing flit B1 follows half a cycle later. Flits B2 and B3 pass through router-2 similarly with bypassing, however the tail flit finds the requested output busy and therefore has to go through allocation spending an additional cycle in router-2.

The control information for H and B1 arrive in router-3 at the end of cycle 3, enabling allocation in cycle 4 as they go to a local port and AB is not an option. Then, in cycle 5 their control is forwarded to the local port and in cycle 6 they traverse the local link. A cycle later B2 and B3 do the same. Finally, the tail flit arrives with an extra cycle delay due to unsuccessful bypassing in router-2.

4.3 Zero-Load Latency Analysis

The Zero-Load Latency (ZLL) of a packet in FreewayNoC is as follows: one cycle for the SA of the first hop as there is no control forwarding yet to cover it; all hops spend at least a cycle; turning hops require an additional cycle as AB is not allowed; the serialization latency is half a cycle per flit for all but the first 2 flits that arrive on the first cycle:

$$ZLL_{FreewayNoC} = 1 + hops + hops_{turn} + (N-2)/2 \; cycles \tag{4.1}$$

The ZLL of a packet in DDRNoC and ShortPath networks is given in Eq. 3.1 and 3.2, respectively. Moreover, the ZLL of an ideal network with two stages per hop (ST

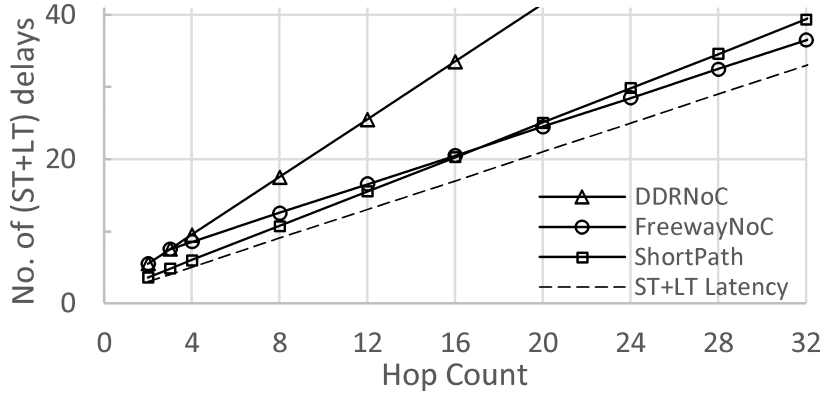


Figure 4.3: ZLL analysis of FreewayNoC, DDRNoC and ShortPath with respect to the number of hops versus minimum packet latency considering ST+LT delay per hop.

and LT) can also be derived from the ShortPath ZLL equation, although its cycle time would be shorter.

As shown in Section 7, the FreewayNoC cycle time is equal to the DDRNoC cycle and about 68% longer than the ShortPath cycle. Moreover, the minimum hop-latency for a NoC router, without any control overhead, is ST + LT. Considering ST and LT are balanced, the cycle time of such a network would be ST (or LT), i.e. 50% of the FreewayNoC cycle.

Taking into account these cycle times and the above ZLL equations we perform a sensitivity analysis of the ZLL with respect to the number of hops a packet traverses depicted in Figure 4.3. We assume a 3-flit packet and that the packet besides the two local port turns also takes a third one (X to Y or vice versa) if the hop count allows. The DDRNoC ZLL scales worse than the other two networks and is up to 50% higher for large number of hops. For low hop counts, ShortPath is better than FreewayNoC, i.e., 35% better for 2-3 hops, but above 18 hops FreewayNoC exhibits lower packet latency, i.e., 2.2% and 7.8% for 20 and 32 hops, respectively. It is worth noting that FreewayNoC scales to the number of hops better than ShortPath. Actually, FreewayNoC scalability is equal to the ST+LT hop latency. It can be observed that FreewayNoC ZLL has a constant offset of 3 to 4 ST+LT delays, depending on the number of turns, with respect to the minimum ST+LT hop latency. This enables FreewayNoC to approach a ZLL that is only 10% higher than the minimum ST+LT hop latency for 32 hops.

4.4 Router Control

4.4.1 Combined VC and Switch Allocation

The FreewayNoC utilizes a combined allocator to grant the requesting flits access to the switch and to assign available output VCs to the head flits of new packets, which are then inherited by the packet. It is composed of a VC Selection (VS) and a Switch Allocation (SA) module as depicted in the top part of Figure 4.1.

The VS maintains a bit-vector per output port keeping track of the free VCs (VCs currently not assigned to a particular packet) in the downstream router. A free VC with

available credits constitutes an *available VC*. The VS keeps a count of the available VCs per output port as shown in Figure 4.4. Moreover, the bit vector indicating available VCs undergoes a V:2 round-robin priority arbitration to select up to two available VCs which are provided to new packets when their head flits win SA, as illustrated in Figure 4.5.

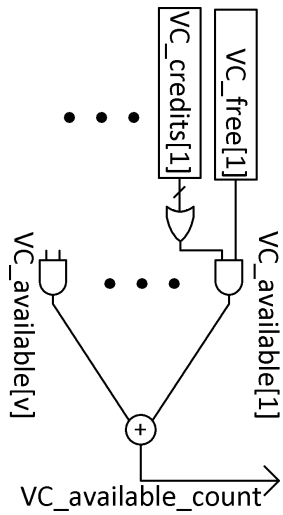


Figure 4.4: VC availability check and count

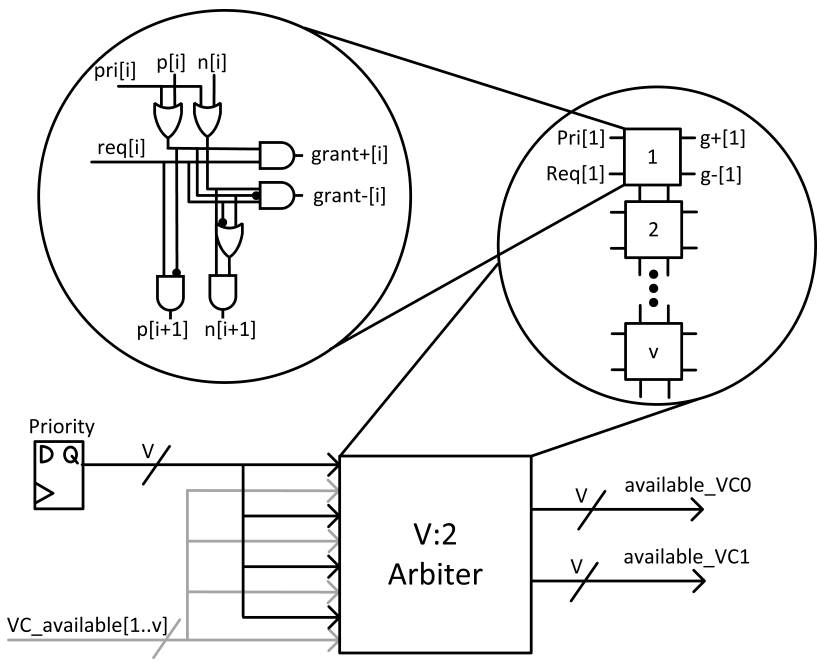


Figure 4.5: FreewayNoC VC selection

Parallel to the VS, the SA receives requests for switch access from flits going to particular outputs. In case of a head flit request, the allocation logic only considers it if VS indicates one or more available output VCs, otherwise the request is ignored. If the head flit is granted access to the switch, then it is also assigned an available output VC, selected by the VS arbiter, which is then marked as busy. Note that the VS arbitration and the SA are performed in parallel. A non-head flit request, is first checked for availability of credits at the already assigned output VC before it is considered by the SA. For a router with P ports and V VCs per port, this requires *(P-1)V:1* multiplexing on the credit counters. Besides filtering out requests that lack a VC or credits, SA ignores requests from flits that are allowed to bypass the allocation stage. So the logic that determines whether a flit can perform allocation-bypassing provides input to the SA. This logic is parallel to the the logic that computes the number of available output VCs and communicates their credits. All remaining requests undergo switch allocation using a separable output first dual grant switch allocator based on *P PV:2* output arbiters (OA) and $2 \times P$ *V:1* input arbiters (two per input port) with round-robin priority arbitration. Head flits selected after OA for a particular output port are at most two and less than *VC_available_count* for that output. The switch allocator design is identical to the one used in the DDRNoC. The SA can grant the two phases of the next cycle to flits of different or same packets.

4.4.2 Allocation Bypassing

The FreewayNoC reduces end-to-end packet latency by allowing packets to skip the SA stage under low load. At the same time it also aims for the critical path of the router to be defined only by the datapath and be independent of the control logic. FreewayNoC achieves this by simplifying the bypass mechanism, by providing bypass paths for the FC signals in the router as well as by operating all control components of the bypass logic in parallel.

Having full AB support from one input port to any other output port requires multiplexing of output port status bits (busy, credit and available VC counters) to verify whether the incoming flits can safely bypass SA in the current cycle. These checks introduce control logic in the datapath which defines the clock period and goes against one of the design goals of the FreewayNoC. Moreover, arbitration or kill logic would also be needed to either resolve contention or disable multiple flits attempting Ab in the same clock cycle. To avoid the latter problems, FreewayNoC supports AB only for flits going straight through the router, i.e., move along one direction. Then, the output port to which each input port allows AB is already known. This means that registered status bits at the output (and input) port can be directly used to decide if AB can occur. Finally, contention resolution is not required at the output port as it only allows bypassing from one input port.

After receiving the FC bits of incoming flits, the input port determines the eligibility of an incoming flit to bypass by checking whether: *(a)* The flit is going straight (i.e. not turning); *(b)* The input VC of the flit is empty; *(c)* For incoming $flit^+$ ($flit^-$), the ST^+ (ST^-) timeslot for the input multiplexer of the crossbar is not allocated to any input VC; *(d)* For a head flit the output port has available VCs (VC Selector signal: *Available_VC_Count*$\neq 0$); *(e)* For a non-head flit, the allocated output VC has credits.

In case these input AB checks are successful, the new FC of the bypassing flit(s) is updated with the following, before it is forwarded to the *AB_FC_mux* at the output port: (i) the NRC result computed using the destination included in the incoming FC,

(ii) an output VC-id provided by the VS. Although, NRC, VS, and input AB checks are performed in parallel based on registered information, they would add delay to the FC path (CST & CLT) and would make it slower. FreewayNoC remedies this by providing a bypass path around CST from each input port (except the local port) to its opposite output port as shown in Figure 4.1. The input AB checks, NRC and VS now safely occur in parallel to each other and in parallel to the normal CST path after which the FC bits reach the *AB_FC_mux* shown in Figure 4.1.

Parallel to input AB checks, the output AB check uses output port availability to set the select of *AB_FC_mux*. If the output port is free, the output AB check allows the bypassing FC to undergo CLT after passing input AB checks. It also informs input VC controller, SA and VS about a successful bypass.

After a successful input and output AB check, the SA aligns the input and output multiplexers of the crossbar to the input VC from where the bypassed incoming flits will undergo ST during the second half of the current cycle or the first half of the next, as shown in the example of Figure 4.2, cycles 3-4, router 2. The FreewayNoC performs the above checks in the clock half that precedes the arrival of the incoming flit[1] and the time margin is sufficient. As soon as the flit arrives, it has been decided whether it will wait in VC buffers for SA or it will immediately proceed to ST.

A normal SA allocation request is always created when an incoming flit is announced by its preceding FC bits. Then, the SA checks the availability of output VCs (for head flits) or credits (non head flits) and in parallel the input AB checks are performed. If the input AB checks pass, the VC controller speculates that output AB checks will also pass and sets the *AB_Req* to block the respective SA request before it enters the arbitration stages. This adds only a small delay of a 2-input AND gate after the downstream buffer availability checks are performed by the SA. In essence, an SA request that passes the input AB check is canceled speculatively without confirming that the output AB check is also positive. We opted for this design decision to avoid creating a longer path before SA arbitration. We have observed that this speculation has negligible impact on performance because: *(i)* under low load the speculation is mostly correct (output AB check is positive), *(ii)* under high load the input AB checks are not successful and thus their respective SA requests are seldom canceled, *(iii)* in case of mis-speculation, other flits can still allocate the output port reducing the chance of bandwidth waste.

4.4.3 Next Route Computation

Next Route Computation (NRC) is based on the destination carried by the forwarded control (FC) of a flit after the FC is registered. In case the flit is allowed to perform AB, the result of the NRC is put to the new FC that bypasses the regular FC. Otherwise, the NRC result is registered and used in the regular FC a subsequent cycle. The NRC is performed in parallel to the AB check so it does not add to its delay.

4.4.4 Flow Control and Minimum Buffer Size

FreewayNoC uses credit based flow control, with one wire per VC indicating to the upstream router the release of VC credits and an additional wire per port to signal

[1]For an FC that arrives at a rising clock edge, its first out of the two data flits, $flit^+$, is registered in an input VC at the next falling edge, and can perform ST (ST^+) in the second half of the cycle. AB has completed its checks in the first half of the cycle.

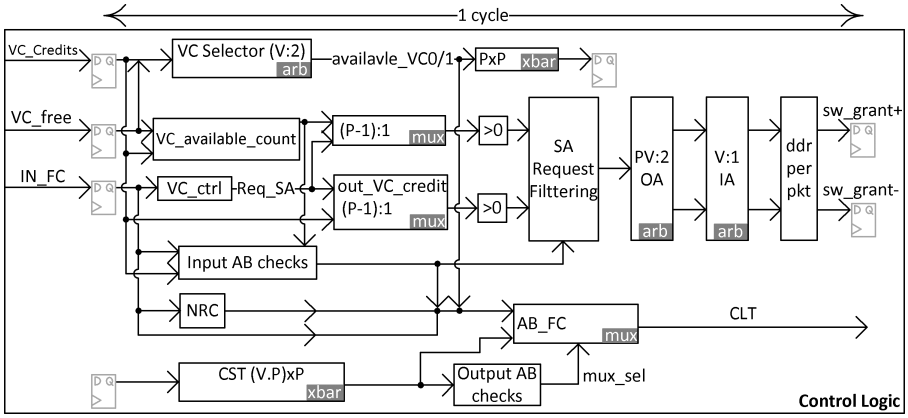


Figure 4.6: FreewayNoC control path analysis.

the release of two credits of one VC. Moreover, in case of a regular flit propagation, credits can be reused after four clock cycles. However, between nodes propagating flit in AB mode, credits can be reused after two clock cycles because the flits do not require SA pipeline stage in either of the two routers.

4.4.5 Control Path Analysis

Figure 4.6 illustrates the various register to register paths of the control logic. The FreewayNoC architecture offers an entire cycle for these paths to complete, as opposed to the datapath parts (ST, LT) which have half a cycle available to transfer a flit. It can be observed that, Input AB check, NRC, and VS selection are in parallel and give input to the CLT. They are also in parallel to the other checks of the SA needed for filtering SA requests before SA arbitration. Finally, the CST is in sequence with the output AB check, the AB_FC_mux and the CLT, while VC selection arbitration is in sequence with a $P \times P$ crossbar for sending the output VC-ids to the input ports.

4.5 Summary

The FreewayNoC router architecture achieves a performance solely dependent on datapath delays. FreewayNoC routes packets at DDR maximizing throughput at a rate defined by the longest datapath stage (ST or LT). It further provides simplified allocation bypassing to improve latency, which per hop is at best equal to the sum of the ST and LT delays. FreewayNoC matches the throughput of previous DDR NoC and achieves a zero-load latency that scales to the number of hops better than previous state-of-the-art NoCs and equally well with a network that has no control overheads.

Chapter 5

The HighwayNoC Architecture

The previously presented DDR NoC design (FreewayNoC) offered DDR datapaths to improve NoC throughput and allocation bypassing support to reduce packet latency. However, it only allowed flits propagating an in-network straight hop to bypass the allocation stage, in order to reduce the complexity of bypassing logic. It offered no bypassing support for turning flits, i.e., flits entering and exiting the network through the local port, as well as flits performing an in-network XY (or YX) turn. However, turns to and from the local port represent a special case, because local ports use shorter links, relative to inter-router links, to connect to the network interface placed in close proximity. So, local links have some slack because of their short length, which can be used to support allocation bypassing for flits entering and exiting the network through these links.

This Chapter presents the HighwayNoC router architecture, which enhances the allocation bypassing support offered by the previous FreewayNoC architecture to also include the local port. HighwayNoC uses a DDR datapath to improve network throughput and allows incoming flits to directly traverse the switch and link without undergoing an allocation step when their way to the output port is free, henceforth referred to as allocation bypassing (AB). Contrary to the FreewayNoC designs, this is possible for all hops except the in-network turns, including entering and exiting the network. HighwayNoC also forwards control information of routed flits to the downstream router a cycle before the flits, to start earlier the allocation of datapath resources as well as the AB checks and next-route computation. In parallel to allocation, a VC is selected out of the available free ones and given to the (header) flit which wins the allocation. Without loss of generality, our HighwayNoC design considers a 2D-mesh network, with look-ahead XY-routing, composed of routers with virtual-channels and credit-based flow control.

The top-level view of the HighwayNoC router is shown in Figure 5.1. The datapath is composed of two stages: Switch Traversal (ST) and Link Traversal (LT), separated by the input VC buffers and the two output registers (one for each clock edge), which are multiplexed before the link. Each stage can handle two flits per cycle, one at the high and one at the low phase of the clock. In addition, the HighwayNoC router has the following control blocks: a combined allocator composed of the Virtual Channel Select (VS) and a Switch Allocation (SA), Next-Route-Computation (NRC) and AB check logic at the input and output ports.

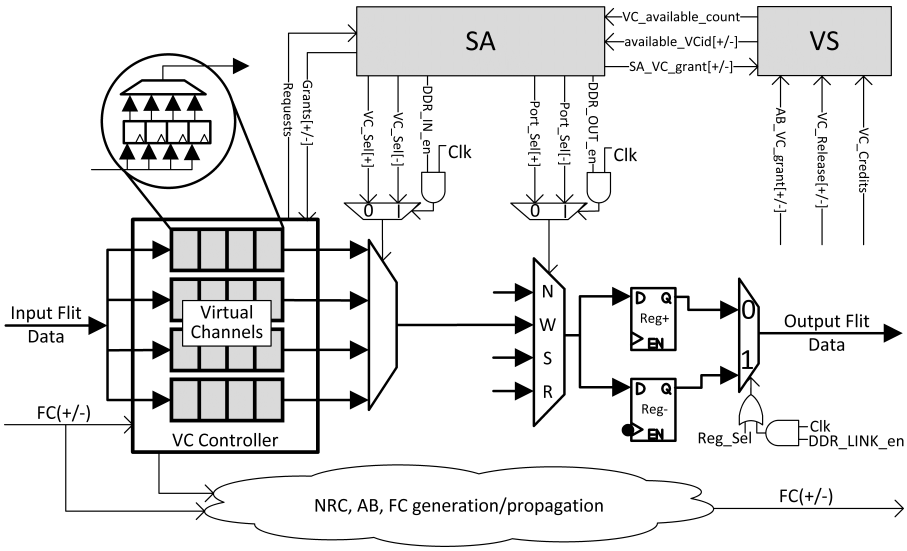


Figure 5.1: The HighwayNoC router architecture.

The rest of this Chapter is organized as follows. In the next Section, we describe the DDR datapath of the data flit, as well as the datapath of the forwarded control signals in a HighwayNoC router. In Section 5.2, we describe the timing details of flits and forwarded control information propagating through HighwayNoC routers, illustrated using a timing example. In Section 5.3 we analyze the zero-load latency of HighwayNoC. In Section 5.4, we describe the control modules of of a HighwayNoC router. Finally, in Section 5.5, we summarize the main aspects of the HighwayNoC architecture.

5.1 Router Datapath

The HighwayNoC router datapath has two parts: the datapath of the data flits and the datapath of the control bits forwarded separately to support control forwarding. Each of the two is described separately below.

5.1.1 DDR Flit Datapath

The DDR flit datapath of a HighwayNoC router, illustrated in Figure 5.1, is similar to the FreewayNoC router datapath described in Section 4.1.1. It also supports two flits per cycle per port, one at each phase of the clock and offers the same alternative ways to propagate flits at DDR and SDR, as presented in Figure 3.3.

In summary, each of the four types of multiplexers[1] along the HighwayNoC datapath can be controlled separately to operate in DDR mode in order to allow two flits per cycle to pass, one at each clock phase. Then, as illustrated in Figure 3.3, the HighwayNoC datapath supports the following cases of DDR: (a) DDR per packet,

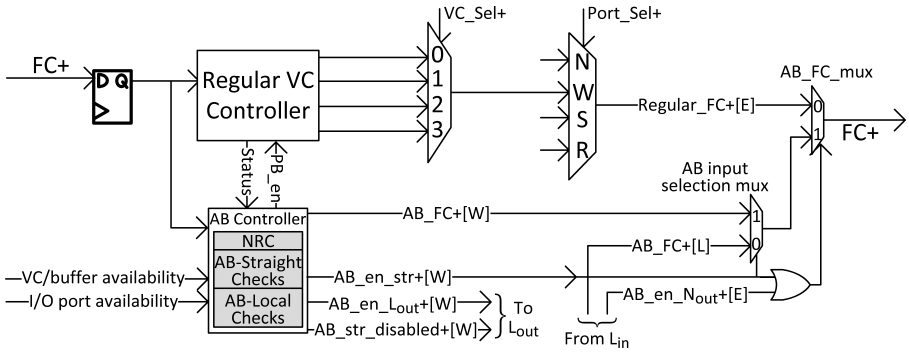
[1]These multiplexers are the following: the multiplexer of a VC buffer, the multiplexer for the input arbitration, the one for the output arbitration, and the multiplexer that selects one of the two output registers before the link.

sending two flits from the same input VC to the same output, (b) sending two flits of different VCs of the same input to a single output, (c) sending two flits of different VCs of the same input to two different outputs and (d) sending two flits of different input ports to a single output. In addition, a datapath part can operate at SDR, then, the select of the respective multiplexer will remain unchanged through the entire cycle.

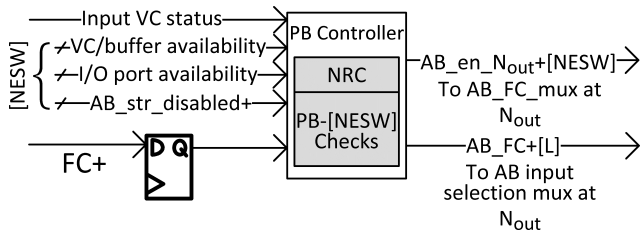
5.1.2 Datapath of Forwarded Control Bits

Control information for each of the two outgoing flits is forwarded to the downstream router one cycle ahead. This mechanism is termed as control forwarding [42]. This forwarded control (FC) uses a separate path (crossbar and link wires) from the datapath to perform control switch traversal (CST) and control link traversal (CLT) in a single cycle as depicted in the lower part of Figure 5.1 and elaborated in Figure 5.2a. Two sets of FC signals exist, one for the flit traversing the link during the high phase of the next cycle ($FC+$) and one for the flit propagating in the low phase (FC-). Each set of FC signals uses separate CST and CLT paths. Moreover, as shown in the lower part of Figure 5.2a, within the router there are additional paths for FC signals to support AB. These paths bypass the regular CST and send FC of bypassing flits, which either (i) traverse an in-network straight hop, (ii) enter the network or (iii) exit the network. In the first case, each non-local input port (N_{in}) sends FC of bypassing flits to its straight non-local output port (N_{out}), in particular West to East (shown in Figure 5.2a), North to South and vice versa. In the second case, FC of bypassing flits goes from the local input port (L_{in}) to each N_{out}. In the third case, each N_{in} sends FC to the local output port (L_{out}). During AB, these bypass paths carry FC signals to the output port for CLT through the *AB_FC_mux*.

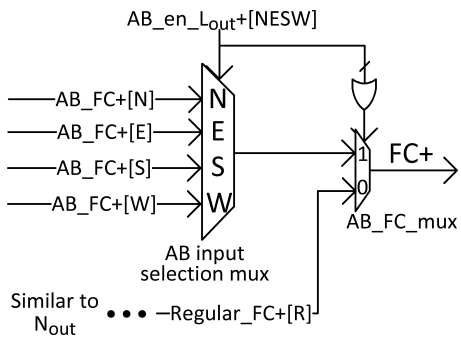
The AB logic checks the respective input and output port availability, as well as for available VC and credits, to determine whether bypassing is possible. These AB checks are performed in parallel to the regular CST. A successful AB check at an input port will enable the FC of the bypassing flit to be sent through the 2:1 *AB_FC_mux* multiplexer directly to CLT overriding the regular FC, as shown in Figure 5.2a. Note that flits coming from the N_{in} as well as from the L_{in} may want to bypass to the same output; in that case the in-network flit has priority as explained in Section 5.4.2 and selected by the 2:1 multiplexer *AB_input_selection_mux*. The two paths of the forwarded control signals, namely ⟨CST+CL⟩ and ⟨AB_checks+CLT ⟩ are not in the critical path of the router. To their advantage is that, unlike the data ST and LT, FC signals do need to be registered at the output port because they are not sent at DDR and in addition the FC path is substantially narrower than the data-path requiring a smaller crossbar. As shown in Figure 5.2, the local input and output ports have different AB support than the in-network ports. When bypassing from the L_{in} to the network, bypassing FC signals generated after AB checks (shown in Figure 5.2b) are multiplexed with the bypassing FC signals of the N_{in} using the *AB_input_selection_mux* as discussed above. Finally, when bypassing to the L_{out}, a 4:1 multiplexer is needed before the 2:1 *AB_FC_mux* (shown in Figure 5.2c) to select one of the N_{in} for bypassing using static priority. This adds latency to the bypassing FC path which is however compensated by the shorter CLT of the L_{out}; that is because there is a shorter link (relative to inter-router links) connecting the local output port of a router to the network interface. In case the local link is not short enough, as previously assumed, AB at the local port cannot be performed. This would have a negative impact to the packet latency of the HighwayNoC and in the worst case may fall back to the latency of the



(a) Regular and bypassing FC+ paths from a non-local input port (West) to its straight non-local output port (East) in HighwayNoC.



(b) Bypassing FC+ path and enable signals originating from the AB controller at the local input port.



(c) Multiplexing of regular and bypassing FC+ signals at the local output.

Figure 5.2: AB controller and FC+ paths at the HighwayNoC input and output ports. FC- paths are identical to FC+ paths.

FreewayNoC, if all local links need to be long.

The FC signals contain the following fields: two sets of VC-ids and NRC results (one for each incoming flit), flit type and finally destination addresses of header flits required for the NRC when bypassing as explained later. In our implementation, we allow only one header flit to bypass per cycle in order to reduce the number of FC bits; then only one destination address needs to be carried. In addition, when non-header flits are transmitted, the NRC field is not used; then the NRC bits are

reused to distinguish between body and tail flits.

5.2 Timing

The timing of a HighwayNoC router is described using an example of a five-flit packet traversing four routers from source to destination node, illustrated in Figure 5.3. In our example, we consider that Router 1, Router 2 and Router 3 are aligned and therefore flits passing through the second router go straight and have the opportunity for AB. Router 4 is not aligned with Router 1 and Router 2, therefore flits entering Router 3 cannot bypass SA as they need to turn in order to propagate towards Router 4. Note that the LT and ST for flits that traverse the link in the high clock phase are marked in the example with "+", and for flits that traverse the link in the low clock phase are marked with "-".

In cycle zero, router-1 receives the forwarded control (FC) of the two first flits (H, B1) which carries flit types, assigned VC, next route and destination. The packet comes from the local port, so depending on the AB checks, it can either undergo or bypass SA stage in cycle 1. AB check performed in the first half of cycle 1 is positive, therefore the flits are not considered for allocation. In parallel, the data flits of H and B1 arrive and get stored in the input VC buffer in the first and second half of cycle 1, respectively. As the first flit is a header, NRC and VC select (VS) are performed, too. NRC, VS, and AB check are performed in parallel to form the control information of the bypassing flits which will bypass the regular CST and traverse the link (CLT) to reach router-2 at the end of cycle 1. In the second half of cycle 1 the H flit traverses the switch of router-1 (ST+) and is registered in *Reg*+ before it performs LT in the first half of cycle 2 (LT+). B1 flit follows half a cycle later. It performs ST and LT in the first and second half of cycle 2, respectively (ST-, LT-). The remaining flits are routed through router-1 in a similar way.

Following again H and B1 flits, their control information arrives in router-2 at the end of cycle 1 while the data flits are received in cycle 2. These flits are routed through router-2 in a manner similar to router-1 because they are passing straight through the router and can bypass the allocation stage. Flits B2 and B3 pass through router-2 similarly with bypassing, however the tail flit finds the requested output busy and therefore has to go through allocation spending an additional cycle in router-2. It is important to mention that depending on the checks which disable AB, switch allocation request for the tail flit can be sent either in cycle 4 (as in this example) or in cycle 5. For more details on AB, see Section 5.4.2.

Following again H and B1 flits, their control information arrives in router-3 at the end of cycle 2 while the data flits are received in cycle 3. Packet route in the received FC informs router-3 that incoming flits need to turn to be directed towards router-4. This disables AB in router-3 for the incoming packet; instead, SA is performed in cycle 3. In the second half of cycle 3, NRC is also performed using the destination address in the header flit received during the first half of cycle 3. The results of SA and NRC are both ready at the end of cycle 3. After successful SA, FC bits undergo CST and CLT in cycle 4, whereas the corresponding H and B1 flits undergo LT in cycle 5. Remaining flits (B2, B3 and T) of the packet pass through router-3 in a similar manner.

The control information for H and B1 arrive in router-4 at the end of cycle 4 while the control information for the B2 and B3 arrive at the end of cycle 5. Now these flits

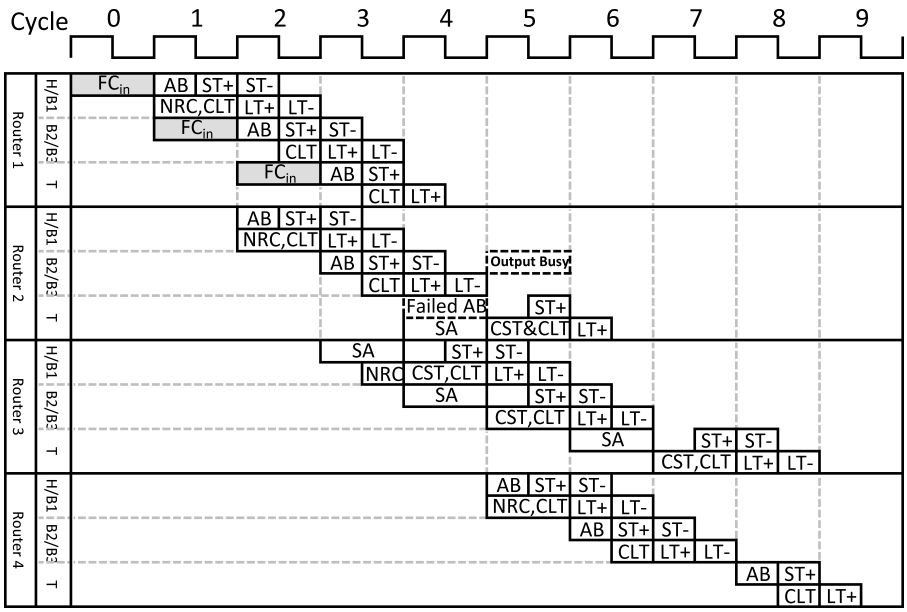


Figure 5.3: Timing diagram showing the flow of a five flit packet through four routers of the HighwayNoC.

have to be routed to the local output port. Successful AB check leads to these flits propagating through router-4 in a similar way as router-1. Finally, the tail flit arrives with an extra cycle delay due to unsuccessful bypassing in router-2.

5.3 Zero-Load Latency Analysis

Next, the Zero-Load Latency (ZLL) of HighwayNoC, FreewayNoC and ShortPath networks are analyzed and compared to the minimum ST+LT latency for the hops (described earlier in Section 4.3). The networks have the same topology, number of VCs and a datapath pipelined in two stages: ST and LT[2]. The analysis takes into account the maximum operating frequency of each network, implemented as described in Chapter 7.

The ZLL of a HighwayNoC packet is as follows: FC signals are registered at local input at rising clock edge; each hop takes a cycle except turning hops that take two because there is no AB; the serialization latency is half a cycle per flit:

$$ZLL_{HighwayNoC} = hops + hops_{turn} + N/2 \; cycles \tag{5.1}$$

The ZLL of FreewayNoC and ShortPath networks is given in Equation 4.1 and 3.2, respectively. Taking into account the cycle time of these networks and their ZLL equations, we perform a sensitivity analysis of the ZLL with respect to the number of hops a packet traverses. We consider a 3-flit packet and assume it always

[2]Although a network with single stage datapath (ST and LT performed in one cycle) would offer slightly lower latency, it would achieve only half of the throughput and therefore is considered a worse option.

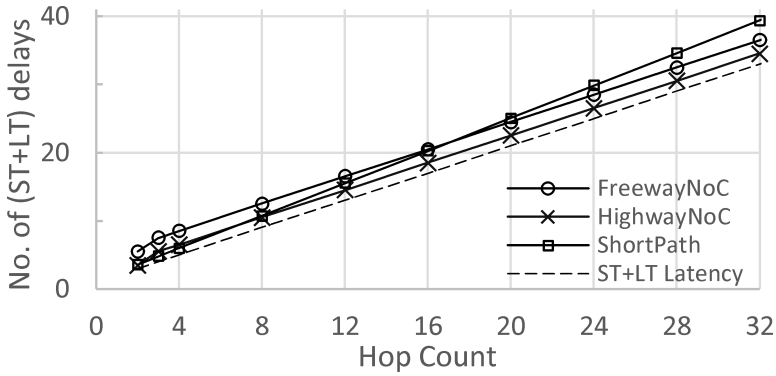


Figure 5.4: ZLL analysis with respect to hop count with the HighwayNoC, the FreewayNoC and the ShortPath networks versus an ideal network that has a delay of ST+LT per hop.

takes an xy-turn if the hop count allows. As shown in Figure 5.4, the ShortPath network scales worse than the FreewayNoC and the HighwayNoC and has 20% higher zero-load latency compared to the minimum ST+LT latency. This is because ShortPath introduces control logic in its critical path which leads to slower clock. FreewayNoC scales better than ShortPath because its datapath operates at DDR to have a critical path independent of control logic and offers limited allocation bypassing support. With higher number of hop, ZLL of FreewayNoC is 7% lower compared to ShortPath. HighwayNoC scales even better than FreewayNoC. It is worth noting that HighwayNoC scales to the number of hops better than ShortPath. Actually, HighwayNoC scalability is equal to the minimum ST+LT latency. It can be observed that HighwayNoC ZLL has a constant offset of 1.5×(ST+LT) delays with respect to the minimum ST+LT latency. Two thirds of this offset is due to the considered turn. The remaining $\frac{1}{3}$ of the offset is due to the bypassing checks required before the first hop. This second overhead can be hidden in cases where the control of a packet can be generated before the actual data arrive; this is common in network interfaces [53] as well as in cache accesses where tag matching is faster than reading data [54]. The fact that HighwayNoC ZLL has a constant offset relative to the ideal ZLL enables it to be only 4.4% slower than the ideal network for 32 hops at zero load, or only 1.5% when there is no turn in the path.

5.4 Router Control

The control of the HighwayNoC router is described next. First the DDR allocation algorithm is described, which is simplified compared to previous DDR NoCs. Then, the AB control is detailed, which is extended to support bypassing at the local ports. Subsequently, the route computation and flow control are discussed. Finally, the delay of the control paths is analysed.

5.4.1 Combined VC and Switch Allocation

The HighwayNoC utilizes a combined allocator to grant requesting flits access to the switch and assigns available output VCs to head flits of new packets. It is composed

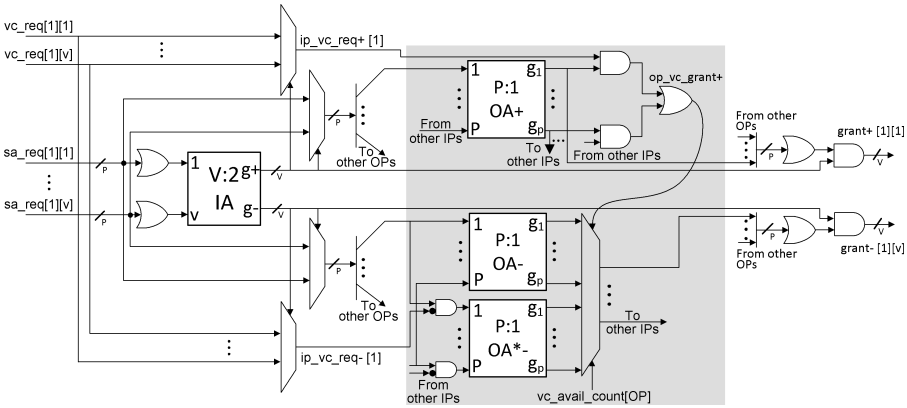


Figure 5.5: HighwayNoC dual grant input first Switch Allocator

of a VC Selection (VS) and a Switch Allocation (SA) module as depicted in the top part of Figure 5.1. The implementation of VS is similar to that of the FreewayNoC VS, described in Section 4.4.1. Moreover, the mechanism with which the SA in the HighwayNoC removes allocation requests with insufficient downstream credits or VCs and requests from flits allowed to bypass allocation, is also similar to the FreewayNoC and described in Section 4.4.1. However, the HighwayNoC utilizes separable input first dual grant switch allocator as opposed to the output first dual grant allocator in the DDRNoC and the FreewayNoC networks. Input first allocators reduce complexity, area and power costs while offering throughput similar to output first allocators [39]. The HighwayNoC SA is DDR and can grant the two phases of the next cycle to flits of different or same packets.

The HighwayNoC SA shown in Figure 5.5 utilizes a dual grant *V:2* round-robin priority input arbiter (IA), identical to the arbiter used for VC selection in Figure 4.5, to select at most two requesting input VCs per input port, one for high phase ($g^+[i]$) and one for low phase ($g^-[i]$) of the clock. The input VCs selected by IA for the high clock phase for all the input ports send arbitration requests to *P P:1* round-robin priority output arbiters (OA^+). The grants by the IA for the high clock phase are combined with grants by OA^+ to determine whether requests by input VCs are granted for high clock phase and to generate grant+ signals for input port VCs.

The output arbitration for low clock phase grants OA^- is different from output arbitration for high clock phase. This is because OA^+ gets priority over OA^- to the available output VCs. So, when there is only one output port VC available (vc_available_count=1 in Figure 4.4) and OA^+ gets it (op_vc_grant+ in Figure 5.5 is set) then OA^- should not grant a head flit request. This creates a dependence between the final output of the OA^+ and the beginning of OA^-. In order to optimize for clock speed and parallelize output arbitration for high and low clock phase, we use two sets of *P:1* round-robin priority output arbiters per output port for the low clock phase, one for all $g^-[i]$ requests including header flits (denoted as OA^-) and one for only body/tail flit $g^-[i]$ requests (denoted as OA^{*-}). Both arbiters operate in parallel and the output of one of them is selected depending on the op_vc_grant+ signal. In case op_vc_grant+ is set then at the low phase of the clock only non-header flits should be granted the switch (OA^{*-} outputs are selected) otherwise all types of flits can be granted (OA^- outputs are selected) and the grants are combined with $g^-[i]$ from IA to produce final

grants for input port VCs at the low clock phase.

The SA described so far is not yet able to allow two flits of the same packet to be switched in DDR mode. This is accomplished using additional logic appended to the above SA as presented in Chapter 3 for the DDRNoC architecture. Briefly, when an input VC receives an SA grant a second packet flit can follow if available under the condition that (i) there is enough space in the downstream VC buffer and (ii) the required input and output ports have not been promised by the SA to another flit.

5.4.2 Allocation Bypassing

The HighwayNoC reduces end-to-end packet latency by allowing packets to skip the SA stage in the absence of contention. At the same time, it also aims for the critical path of the router to be defined only by the datapath and remain independent of the control logic. HighwayNoC achieves this with a simplified AB mechanism, offered in all but the in-network turning paths, which would require excessively complex logic.

Offering complete AB from each input port to any other output port would require multiplexing of output port status bits (busy, credit and available VC counters) to verify whether the incoming flits can safely bypass SA in the current cycle. These checks would introduce control logic in the datapath, which defines the clock period and goes against one of the design goals of the HighwayNoC. Moreover, arbitration or kill logic would also be needed to either resolve contention or disable multiple flits attempting to concurrently bypass to the same output. To avoid these problems, HighwayNoC supports AB only for:

- in-network flits that go straight through the router, i.e., move along one direction, and therefore turning flits do not compete with them for bypassing;
- flits that enter the network, having lower priority compared to in-network bypassing flits;
- flits that exit the network, exploiting the shorter link traversal to perform a fixed arbitration between them.

This requires two AB check modules at each non-local input port (N_{in}) to determine bypassing feasibility to the straight non-local output port (N_{out}) and to the local output port (L_{out} and one AB check module at the local input port (L_{in}) for bypassing to any N_{out} as shown in Figure 5.2.

Each input port determines eligibility of its incoming flits to bypass by performing AB checks based on flit route indicated in the received FC bits. As shown in Figure 5.6, a set of some initial bypassing checks determine whether an allocation request should be sent to SA. These checks verify that:

- the flit traverses in a direction which supports bypassing;
- there are no buffered flits in the input VC which will receive the incoming flit (to ensure in-order delivery of flits in a packet);
- The input port is free, i.e. for incoming $flit^+$ ($flit^-$), the ST^+ (ST^-) timeslot for the input multiplexer of the crossbar is not allocated by the SA to any input VC;

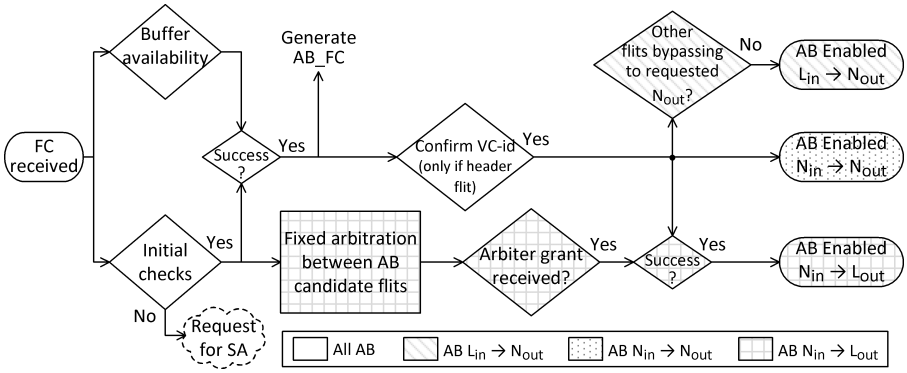


Figure 5.6: Checks performed in order to guarantee conflict free propagation of bypassing flits.

- The output port is free, i.e. for incoming $flit^+$ ($flit^-$), the ST^+ (ST^-) timeslot for the output multiplexer of the crossbar (which corresponds to LT^+ (LT^-) from the output port) is not allocated by the SA to any input port.

During this phase, our HighwayNoC implementation additionally checks that up to one header flit bypasses per input port per cycle. As explained in Section 5.1.2, this restriction is imposed only to limit the number of FC bits.

These *initial checks* are performed based on already registered status information in the router input and output and require little logic[3]. Passing these initial checks does not guarantee successful bypassing, however, they are used to filter out SA requests by incoming flits, holding back those that pass the initial checks. Performing a complete AB check before SA would be too slow to fit in the targeted cycle time. However, using these initial AB checks is still more efficient than naïvely allowing all SA requests to be processed and then to kill the granted ones which also pass complete AB checks, like [29], because this would affect throughput.

As shown in Figure 5.6, parallel to the initial AB checks, the received FC bits of incoming head and body/tail flits are used to check downstream *buffer availability* by accessing counters of available VCs and credits, respectively. Higher priority access to available downstream VCs is given to bypassing header flits already in the network over those coming from the local input. However, none of the bypassing header flits has priority to available VCs over non-bypassing flits.

Upon successful initial bypassing and buffer availability checks, the new FC of the bypassing flit(s) is generated, before it is forwarded to the *AB_input_selection_mux* at the output port as shown in Figure 5.2. The fields of the new FC are computed as follows:

- The next route is updated in the FC of a header flit after it is computed using the destination address carried by the received FC. The next route of a body/tail flit is copied from the VC state register.

- The output VC-id of a header flit is provided by the VS and that of a body/tail flit by a VC state register.

[3] A V:1 multiplexer is needed to check whether the input VC is empty and a P-1:1 multiplexer only at L_{in} to check whether the requested N_{out} is free.

- Finally, the flit type and the destination address bits (only for header flits) are copied from the received FC.

After the generation of the bypass_FC signals, bypassing header flits need to finally confirm they did not violate priority over non-bypassing header flits when getting a VC.

Additional bypassing checks need to be performed for flits that enter or exit the network, as depicted in Figure 5.6.

Multiple incoming flits may fulfil the initial bypassing and buffering criteria to bypass to the L_{out}. Therefore, bypassing to the L_{out} calls for conflict resolution between the non-local inputs. In order to resolve conflicts, HighwayNoC uses two *P-1:1* fixed priority arbiters, one for each clock phase ($FC^{+}s$, $FC^{-}s$). The delay of these arbiters can be easily accommodated exploiting the slack present in the shorter L_{out} link. An arbitration request is only sent to these arbiters when the received FC satisfies initial bypassing checks as shown in the left part of Figure 5.6.

Finally, bypassing from the L_{in} to a N_{out} is allowed only when there is no other flit, already in-network, bypassing to the same output. In order to fit our time-budget, this condition is not fully checked for in-network header flits. In that cases, it is assumed that an in-network flit is bypassing before its VC-id is confirmed. This may result in unnecessarily missed opportunities for bypassing, which yield to a negligible (less than 1%) increase in latency at high injection rates.

Although, NRC, VS, and all the above input AB checks are performed in parallel based on registered information, they would add delay to the FC path (CST & CLT) and would make it slower. HighwayNoC remedies this by providing a bypass path around CST between each input-output bypassing pair. The input AB checks, NRC and VS now safely occur in parallel to each other and in parallel to the normal CST path after which the FC bits reach the *bypassing_input_selection_mux* as shown in Figure 5.2.

After a successful AB check, the SA aligns input and output multiplexers of the crossbar to the input VC of the bypassing flit to enable ST during the second half of the current cycle or the first half of the next. Figure 5.3 shows AB examples in Router-1 from the L_{in} to a N_{out}, in Router-2 from a N_{in} to its straight N_{out}, and in Router-4 from a N_{in} to the L_{out}.

5.4.3 Next Route Computation

Next Route Computation (NRC) is performed only for header flits and its result is stored in the input VC state registers to be used by the subsequent flits of the packet. As discussed in Section 5.2, NRC is performed at different timing for bypassing and non-bypassing flits. Bypassing flits have their destination address carried by the FC. Then NRC is performed after FC is registered and before the new FC is created and sent to the downstream router. Figure 5.3 shows an example of NRC in Router-1 for bypassing header flit in cycle 1.

Non-bypassing flits that are transmitted in the low phase of the clock use the same approach as bypassing flits in order to have NRC ready together with allocation results. On the contrary, non-bypassing flits sent during the high phase of the clock are treated differently. For these flits, the FC does not need to carry their destination address. The header flit arrives during the first half of the cycle and carries its own destination address which is registered at the falling edge of the clock. Then, NRC is performed

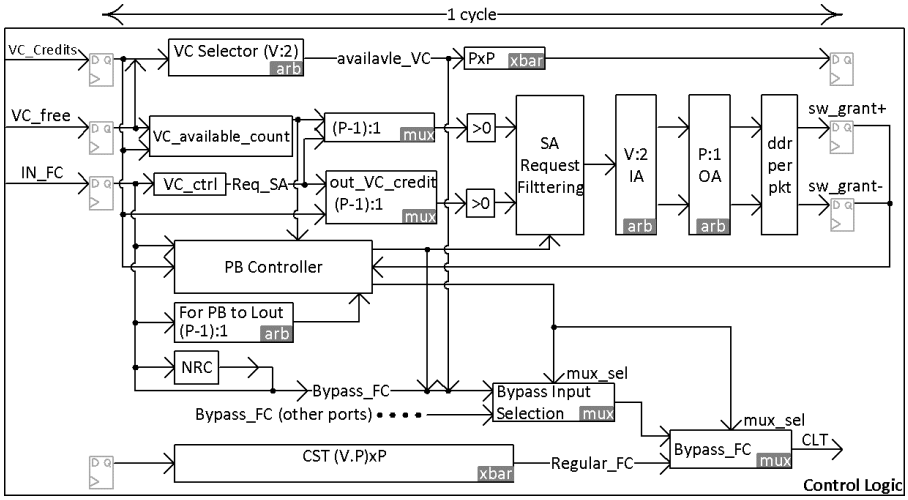


Figure 5.7: HighwayNoC control path analysis.

during the second half of the cycle to be ready together with their allocation result. Such a scenario is illustrated in cycle 3 of Figure 5.3 for Router-3.

The tight time budget of the bypassing logic, which has NRC of bypassing flits in sequence with the bypassing multiplexers and CLT, limits the routing algorithm alternatives. So, when a bypassing path is chosen, the routing algorithm needs to be simple to fit in the available time budget (about 5-6 FO4 stages). In our implementation we use XY routing, but any other (dimension-order) routing algorithm with similar complexity could also be used.

However, when a flit is not bypassing then half a cycle is available for NRC, which allows more complex routing algorithms to be implemented. On one hand, AB is enabled when there is no congestion. On the other hand, advanced algorithms yield most of their performance benefits when there is contention and congestion. Therefore, HighwayNoC could use more advanced algorithms when a header flit is routed normally and fall back to dimension-order routing when bypassing, as long as deadlocks are avoided.

5.4.4 Flow Control and Minimum Buffer Size

HighwayNoC uses credit based flow control, with one wire per VC indicating to the upstream router the release of VC credits and an additional wire per port to signal the release of two credits of one VC. Moreover, in case of a regular flit propagation, credits can be reused after four clock cycles. However, between nodes propagating flit in AB mode, credits can be reused after two clock cycles because the flits do not require SA pipeline stage in either of the two routers.

5.4.5 Control Path Analysis

Figure 5.7 illustrates the various register to register paths of the control logic. The HighwayNoC architecture offers an entire cycle for these paths to complete, as opposed to the datapath parts (ST, LT) which have half a cycle available to transfer a flit. It

can be observed that, AB controller, NRC, and VS selection are in parallel and give input to the CLT. They are also in parallel to the *P-1:1* fixed priority arbiters needed to resolve conflicts among bypassing requests to the local output port and the buffer availability checks of the SA needed for filtering allocation requests before input arbitration. Finally, the CST is also parallel to the AB controller and the Bypass_-Input_Selection_mux and in sequence with the AB_FC_mux and the CLT, while VC selection arbitration is in sequence with a $P \times P$ crossbar for sending the output VC-ids to the input ports.

5.5 Summary

The HighwayNoC router architecture offers an average packet latency and throughput which depends solely on its datapath delays and remains independent of control delays. HighwayNoC routes packets at DDR maximizing throughput at a rate defined by the longest datapath stage (ST or LT). Its packet latency can be as low as the sum of the ST and LT delays for all hops except the in-network turns. Mostly due to its slower turns, HighwayNoC adds a constant latency overhead of 1.5 cycles to each end to end packet transition. Still, its zero-load latency scales to the number of hops equally well with an ideal network that has no control overheads.

Chapter 6

The FastTrackNoC Architecture

The previously presented DDR NoC architecture (HighwayNoC) operated its datapath at DDR, in order to improve network throughput, and offered allocation bypassing support for flits propagating an in-network straight hop as well as flits enter and exiting the network through the local port, in order to reduce packet latency. Allocation stage can be bypassed because it is used to resolve contention among competing flits and is not needed in the absence of contention. Following the same line of thought, we may also be able to bypass the ST stage in some cases. ST stage is needed to change the routing dimension of a flit. It can be bypassed when a flit traverses a hop without changing its dimension. Bypassing the ST stage, however, requires fast bypass paths from the input port to the requested output port. In order to optimize for the common case, these bypass paths can be provided from every network input port to its straight output port, because a packet mostly propagates straight and has at most one in-network XY-turn.

This Chapter presents the FastTrackNoC architecture which offers an extra bypassing path between every input port and its opposite output as shown in Figure 6.1. This enables flits to proceed directly to the link without passing through the switch, thereby reducing latency. FastTrackNoC datapath uses both clock edges, i.e., it is able to route up to two flits per port per cycle at Dual Data Rate (DDR), to maximize its switching rate.

In a regular mode of routing a flit, FastTrackNoC spends two cycles per hop. More precisely, one cycle for control, i.e., allocation, next route computation (NRC), and other checks, half a cycle for switch traversal (ST) and half a cycle for link traversal (LT) as shown in Figure 6.2. Using lookahead signalling, forwarded control (FC) information arrives at the downstream router a cycle before the flits to enable the control stage a cycle before flits arrive. Depending on contention and the flit direction, the FastTrackNoC router can offer two faster ways to route a flit. A non-turning flit arriving at the head of VC-0 can use the FastTrack (FT) path if the output is free and proceed directly to link traversal (LT) spending half a cycle per hop. If that is not possible, incoming flits may still be able to bypass the control (allocation) stage and directly traverse switch and link when their way to the output is free. This is possible for all VCs and hops including entering and exiting the network, except of in-network turns, and allows flits to be routed with a latency of one cycle per hop.

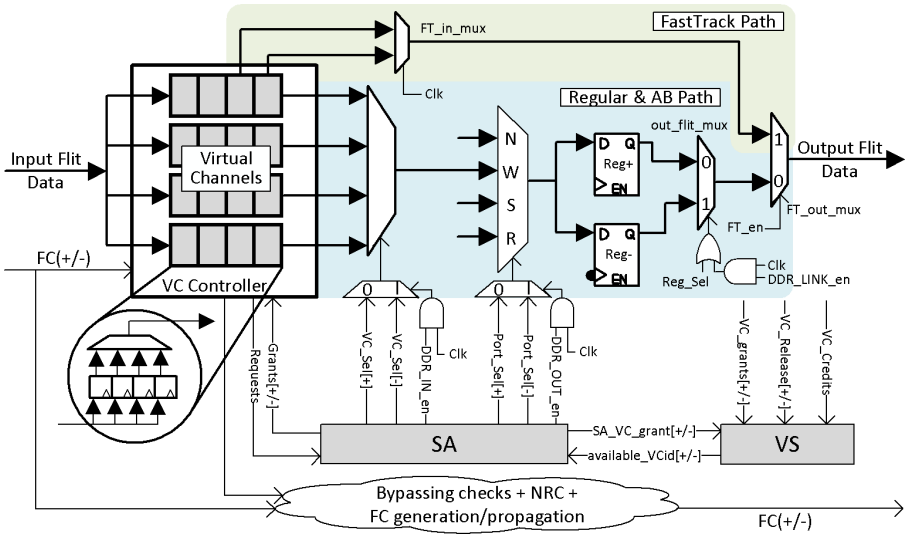


Figure 6.1: FastTrackNoC router architecture; west input to east output.

Without loss of generality, our FastTrackNoC design considers a 2D-mesh network, with lookahead XY-routing, composed of routers with virtual-channels and credit-based flow control. The datapath, illustrated in Figure 6.1, is composed of two stages: Switch Traversal (ST) and Link Traversal (LT), separated by the input VC buffers and the two output registers (one for each clock edge), which are multiplexed before the link. Alternate bypass paths also exist to support FT traversal by multiplexing flits stored in input VC-0 on the output link. Each stage can handle two flits per cycle, one at the high and one at the low phase of the clock. In addition, the FastTrackNoC router has the following control blocks: a combined allocator composed of the Virtual Channel Select (VS) and a Switch Allocation (SA), Next-Route-Computation (NRC) and bypassing check logic at the input and output ports.

The rest of this Chapter is organized as follows. In the next Section, we describe the DDR datapath of the data flit, as well as the datapath of the forwarded control signals in a FastTrackNoC router. In Section 6.2, we describe the control modules of a FastTrackNoC router. In Section 6.3 we analyze the zero-load latency of FastTrackNoC. In Section 6.4, we illustrate the timing details of flits and forwarded control information propagating through FastTrackNoC routers using a timing example. Finally, in Section 6.5, we summarize the main aspects of the FastTrackNoC architecture.

6.1 Router Datapath

The FastTrackNoC router datapath has two parts: the datapath of the data flits, which is further divided to the regular and the FastTrack path, and the datapath of the control bits forwarded separately to support lookahead signalling. Each of them is described separately below.

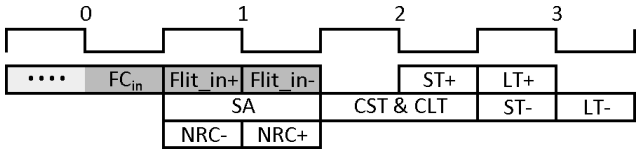
6.1.1 DDR Flit Datapath

The datapath of the flits carries flits from an input to an output port. It is composed of two parallel intra-router paths, namely, the regular datapath and the FastTrack datapath, which are multiplexed at the output before the link.

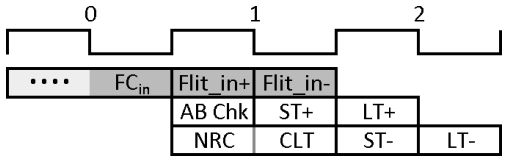
Regular flit path: The regular datapath is composed of input VCs, crossbar switch, output registers and link wires and it is illustrated in Figure 6.1. This path is similar to the regular datapath of DDR NoC routers presented in previous Chapters. It can transfer two flits per cycle per port, one at each phase of the clock. At the input side, each register in a VC buffer can be loaded selectively, either at the negative or the positive clock edge, to allow the VC to store up to two incoming flits per cycle. The multiplexer of the input VC buffer along with the input and output multiplexers of the crossbar then transfer flits stored in the input VCs to the requested output port. These multiplexers operate at DDR and can transfer up to two flits from one or two different VCs in a single clock cycle, i.e., one flit per clock phase. This is accomplished by changing the select signals of these multiplexers during each clock phase. After the output multiplexer of the crossbar, flits are registered in one of the two output registers ($Reg+$ and $Reg-$), one using the positive clock edge and the other the negative. Then, the contents of these registers are multiplexed to the link and sent to the downstream router at DDR.

FastTrack flit path: The FT datapath is parallel to the crossbar switch and the output registers, as shown in Figure 6.1, and transfers flits in FT mode. FT mode allows incoming flits to bypass SA and ST stages, in the absence of contention, and directly traverse the link in half a clock cycle, as shown in Figure 6.2c. It offers FT support to flits traversing an in-network straight hop; in particular, from west to east (shown in Figure 6.1), north to south and vice versa. FT support is limited to flits that arrive at the head of input VC-0. Connecting the FT path to only the head of VC-0 minimizes the logic of the path and thus the delay added to the link as it avoids multiplexing among (VC) buffer registers and input multiplexing. This restriction is however independent of the allocated output VC. More precisely, the FT path propagates only incoming flits stored at the head, registers 0 and 1, of an otherwise empty input VC-0 buffer. So, VC-0 FIFO implementation prioritizes storing its incoming flits in registers 0 and 1 when it is empty (contrary to a simple circular FIFO for other VCs) to facilitate FT traversal. As a consequence, selecting incoming flits for FT requires a 2:1 multiplexer (FT_in_mux) rather than $(B \times V) : 1$, where B is the buffer size of a VC and V the number of VCs. The FT_in_mux operates at DDR using a gated-clock for select to allow the FT path to send two flits in a cycle using both clock edges. It is worth noting that, SDR NoC router would support ST bypassing with an FT path that starts only from the head register-0 of VC-0 and thus FT_in_mux would not be needed. At the other end of the FT path, a 2:1 multiplexer (FT_out_mux) is used to multiplex the FT and the regular path of the router to the link. This is a *2:1* rather than *(P-1):1* multiplexer as FT traversal is restricted to only in-network straight hops.

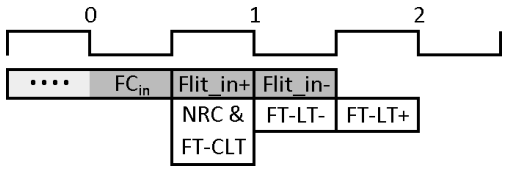
The two aforementioned optimizations at the two ends of the FT path, reduce its delay, which is in the critical path of the router as shown in Section 7.2. More precisely, the latency added to the LT delay is the delay of the two 2:1 multiplexers (FT_in_mux, FT_out_mux) plus the wire delay from input to opposite output. This can affect either the maximum network clock frequency or the link length. Our implementation opts for the latter to ensure that the FastTrackNoC offers maximum switching rate and hence maximum throughput.



(a) Regular routing mode. Cycle-0: forwarded control (FC) signals arrive. Cycle-1: allocation and NRC performed, incoming flits buffered. Cycle-2: new FC signals go through ST (CST) and LT (CLT). Second half of Cycle-2 and Cycle-3: the two flits (+/-) perform ST and LT.



(b) Allocation Bypassing (AB) mode. Cycle-1, first half: NRC and checks for bypassing allocation. Cycle-1, second half: new FC is sent through the link and ST of first flit (+). Cycle-2: ST of the second flit (-) and LT of both flits.



(c) FastTrack (FT) mode. Cycle-1, first half: FT checks, NRC and FC sent through the link. Cycle-1, second half: LT of first flit via FT path. Cycle-2, first half: LT of second flit via FT path.

Figure 6.2: Modes of routing a flit in a FastTrackNoC router.

6.1.2 Datapath of Forwarded Control Signals

The FastTrackNoC uses lookahead signalling [42] to forward control information ahead of the flits. This forwarded control (FC) information is received by the downstream router, half or one full cycle, before the arrival of the flits using separate link wires. This information not only allows allocation stage in a router to be overlapped with LT stage of the upstream router, saving a cycle in regular mode of flit traversal, but it also informs downstream input VC registers to store subsequent incoming flits at the appropriate clock edges. There are two sets of FC signals, one for the flit traversing the link during the high phase of the clock cycle (FC+) and one for the flit traversing in the low phase (FC-). Within the router, there are three parallel alternative FC paths, one for each of the routing modes, namely the regular, allocation bypassing (AB), and FT mode. The signals of the three FC paths are generated by the respective controller (for regular, AB and FT routing)and are multiplexed on the output link as shown in Figure 6.3.

Regular Forwarded Control Paths: The regular forwarded control (FC) path transfers once every cycle control information of up to two flits (FC+, FC-) routed in regular mode in the two phases of the subsequent cycle. When a flit at the input port is granted access to the switch by SA, its FC signals first undergo a switch traversal, called control switch traversal (CST), to reach the requested output port and then they are multiplexed on the link for a control link traversal (CLT). CST and CLT together take one complete cycle, as shown in Figure 6.2a and are not pipelined as opposed to

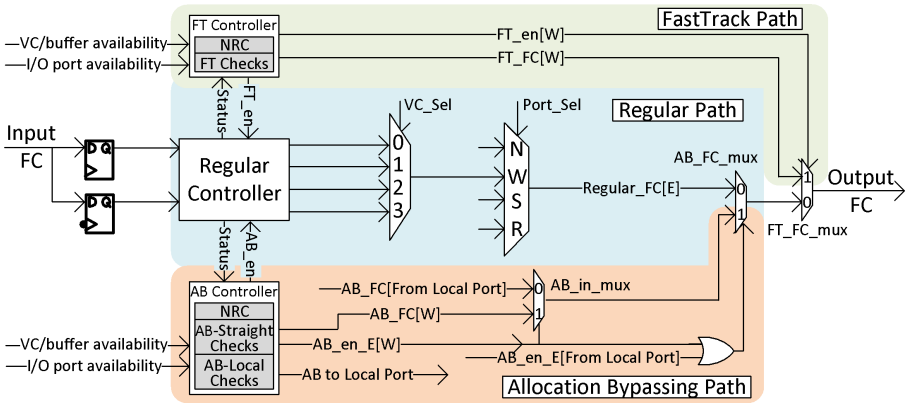


Figure 6.3: Datapaths of the forwarded control (FC) signals for the regular, AB and FT routing modes. FC+ is depicted, FC- paths are identical.

the ST ad LT of the flits.

Forwarded Control Paths for Allocation Bypassing: Control information of flits traversing the router in Allocation Bypassing (AB) mode is also forwarded once a cycle using separate FC paths in the router, which skip CST and are multiplexed with the regular FC path at the output port. CST is not needed for this path because, as in the HighwayNoC, AB is supported for non-turning flits, or flits that enter or exit the network. For non-turning flits, each non-local input port sends FC of respective flits to its straight non-local output, as shown in Figure 6.3 for west to east. Flits entering the network, can also bypass allocation with a lower priority compared to concurrently arriving in-network flits, then, their FC signals go from the local input port to each output and are multiplexed with the FC of in-network AB flits (using *"AB_in_mux"*). Finally, for exiting flits, each input sends FC to the local output port where an arbitration is performed with static priority and FC of winning input is selected using an additional 4:1 multiplexer. This adds extra delay to the local output FC path, which is however compensated by its shorter local link traversal.

Forwarded Control Paths for FastTrack Mode: FC signals of incoming flits routed in FT-mode are forwarded through separate FC paths in the router. They are generated in the FT controller, detailed in Section 6.2.3, and carried to the opposite output, where they are multiplexed last with the regular and AB FC paths through a 2:1 multiplexer (*FT_FC_mux*) before they traverse the link. The FC path for FastTrack mode is a half cycle path and can be active during either of the two phases of a cycle. This requires that a negative edge triggered register is added parallel to the regular positive edge triggered one to store FC sent in the high pulse of the clock, as shown in Figure 6.3. It is worth noting that the path for the FT-mode is the most critical out of the three FC paths because it is a half cycle path and it includes the LT delay. Besides the above 2:1 multiplexer and the link delay, this path also includes the delay of the FT controller, which has a latency of about two 2:1 multiplexers as is discussed in Section 6.2.3. The FT controller used at every non-local input port checks whether incoming flits can continue in FastTrack mode, generates the FC signals and allows them to go out through the link. As explained in detail in Section 6.2.3, FT controller relies on two bits of registered information to minimize its logic: (i) one bit indicating that

the required router resources are available to support FastTrack mode for in the next half cycle, and (ii) a bit carried by received FC indicating that the incoming flit(s) can continue in FastTrack mode. The logic AND of the two bits determines whether FT mode can be turned on and allows the FC signals of this path to traverse the link. One more module that is in the critical path of generating the new FC signals in FT mode is the Next Route Computation (NRC). As explained in Section 6.2.4, FastTrackNoC simplifies NRC for FT traversal reducing its delay by hard-coding the comparison of the destination address with next-node id.

FC Signals Format: The FC signals at each port contain the following fields: two sets of VC-ids and NRC results (one for each incoming flit), encoded flit types of two flits, destination addresses of header flits required for the NRC when bypassing (as explained in Section 6.2.4), a bit to indicate if FC is valid in negative or positive clock edge (*FT_Active*), and a bit indicating whether incoming flits can continue routing in FT mode (*FT_Capable*). Similar to the HighwayNoC, various optimizations are performed to reduce the number of FC bits. In particular, only one header flit is allowed to bypass per cycle, in order to require only one destination address in the FC bits, moreover, when non-header flits are transmitted, the unused NRC bits are reused to distinguish between body and tail flit types.

6.2 Router Control

The control of the FastTrackNoC router includes a combined DDR allocator, the control logic for FT and AB modes, as well as the next route computation and flow control logic.

6.2.1 Combined VC and Switch Allocation

The FastTrackNoC utilizes a combined allocator which grants requesting flits access to the switch and assigns available output VCs to head flits of new packets. It is composed of a VC Selection (VS) and Switch Allocation (SA) module and is similar to the HighwayNoC allocator. In a nutshell, the VS performs V:2 fixed priority arbitration of available VCs (unassigned downstream VCs with credits) to select up to two VCs which are provided to new packets in the next cycle, when their header flits bypass SA or ST stages or win SA. The VS uses fixed priority arbitration and gives to packets higher priority access to output VC-0 which supports FT traversal, thereby, improving chances for flits to be routed in the FT-mode. Parallel to the VS, the SA receives requests for switch access from flits going to particular outputs. A head flit SA request is considered only if the VS indicates one or more available downstream VCs. Similarly, a non-head flit SA request is first checked for availability of credits in the assigned output VC. Besides filtering out requests that lack a VC or credits, SA also ignores requests from flits that propagate using FT or AB modes. All remaining requests undergo allocation using a separable input-first dual grant switch allocator, similar to the one used in the HighwayNoC presented in Chapter 5. In the first stage of the allocator, input arbitration is performed on requests from input VCs. This uses P *V:2* arbiters, one per input port. After input arbitration, up to two winning requests (one for each clock phase) from each input port, undergo *P:1* output arbitration using two different sets of output arbiters, one per clock phase. Finally, SA uses additional logic to allow two flits of the same packet to be granted access to the switch in DDR mode.

When an input VC receives an SA grant, a second flit of the same packet, if available, can follow under the condition that (i) there is enough space in the downstream VC buffer and (ii) the required input and output ports have not been promised by the SA to another flit.

6.2.2 Allocation Bypassing Controller

AB-mode allows incoming flits to bypass SA stage in three cases: when flits go straight in the network, when they enter or exit the network. AB is not offered for in-network turns as this would require arbitration among concurrently arriving flits and would be too slow. An AB check module is located at each input port and determines eligibility of incoming flits to bypass the SA stage. The AB checks are split in two types: (i) initial AB checks, and (ii) buffer availability checks.

Initial AB checks use simple logic on registered information to filter out requests that otherwise would enter SA. For a flit to successfully pass the initial checks is a necessary but not sufficient condition for AB and includes the following:

- The flit traverses in a direction that supports AB.
- There are no buffered flits in the input VC that receives the flit (to ensure in-order delivery of flits in a packet).
- FT-checks failed and the flit cannot be sent in FT-mode.
- The input and output ports are free, i.e., the respective ST and LT slot is not allocated elsewhere by SA.

Parallel to the initial AB checks, the received FC bits of incoming head and body/tail flits are used to check downstream buffer availability by accessing counters of available VCs and credits, respectively. Higher priority access to available downstream VCs is given to AB header flits already in the network over those coming from the local input. However, none of the AB header flits has priority to available VCs over non-bypassing flits.

Additional bypassing checks need to be performed for flits that enter or exit the network. Bypassing from the local input is allowed only when there is no other flit, already in-network, eligible to bypass to the same output. For flits exiting the network, a *P-1:1* fixed priority arbiter is used to resolve contention if multiple concurrently incoming flits fulfill the criteria to bypass to the local output port. The delay of these arbiters is compensated by the shorter local output link.

All AB checks and generation of new bypassing FC signals occur in parallel to each other and in parallel to the normal CST path after which they are multiplexed as shown in Figure 6.3. Finally, after a successful AB check, the crossbar is set to allow the bypassing flit to traverse the switch during the second half of the current cycle or the first half of the next, as shown in Figure 6.2b.

6.2.3 FastTrack Controller

FastTrack mode is possible for in-network, non-turning flits arriving at the head of VC-0. Therefore, FastTrack checks are carried in all, except the local input ports and have a very tight time budget, as explained in Section 6.1.2. The time constraint is met by having most of the checks performed and registered before the clock ticks. In particular, two registered bits of information are used as shown in Figure 6.4: (i) one

bit in the forwarded control of an incoming flit, called *FT_capable*, which indicates whether the flit is capable to continue in FT mode and (ii) another bit, called *FT_ready* that indicates the router, for the particular input-output combination, is ready to allow FT traversal of a flit.

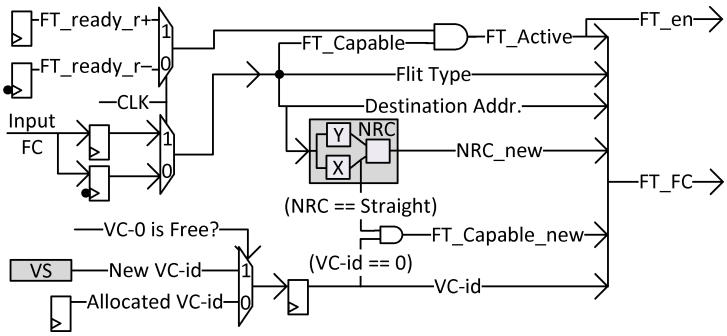


Figure 6.4: FT Controller updates the FastTrack version of the FC information and sets FastTrack enable signal (FT_en) before CLT.

FT_capable: The incoming flit can perform an FT traversal (*FT_capable* set) when the following conditions are met: there is valid incoming flit, its NRC result indicates it is not turning, and it uses VC-0 to be stored. This should be checked for up to two incoming flits arriving in a cycle to identify if one or both of these flits can traverse the hop in FT-mode. Note that for a flit that performs multiple hops in FT-mode, a new *FT_capable* signal needs to be generated before going forward (*FT_capable_new* in Figure 6.4). FastTrackNoC simplifies the (pre-)computation of *FT_capable* by allowing only flits of the same packet to use FT-path within the same cycle, therefore having to deal with only one VC selection and one direction-check using a hard-coded value. Another simplification is that the same *FT_capable* signal is used for both flits in a cycle, disabling the option of only one out of two valid flits to be FT capable. This last constraint costs 1% in average packet latency. The above reduce the complexity of generating a new *FT_capable* to a 2-bit AND gate with the inputs indicating that (i) VC-0 is granted and (ii) the flit direction is straight. The later signal is generated by the NRC and takes as input half of the destination address for the dimension currently traversed. Overall, the longest path of the FT controller starts from the registered FC signals and includes (i) a 2:1 multiplexer and (ii) the NRC (or half of NRC logic plus the AND-gate for *FT_capable_new*), summing up to roughly the delay of two 2:1 multiplexers.

FT_ready: The particular input and output of a router is able to support a FT traversal (*FT_ready* set) when the following conditions are met:

- Input VC-0 is empty for the next cycle to store the up to two flits at the head of VC-0 (registers 0 and 1).
- The input port is free for a complete cycle.
- The output port is free to send forwarded control (FT) signals through the link for the respective slot.
- The output port is free for the respective slot.
- Free downstream VCs will be available in case incoming flit is a header.
- At least two credits of the allocated output VC will be available for non-header

incoming flits.

It is worth noting that the FC signals of FT-propagated flits may arrive at either clock edge. So, the router prepares two *FT_ready* bits, registered separately in a positive and a negative-edge triggered register, as shown in Figure 6.4, and uses the appropriate one according to the FC timing.

In summary, FastTrack is enabled when incoming flit(s) are capable of FastTrack and the router is ready to support it.

6.2.4 Next Route Computation

Next Route Computation (NRC) is performed only for header flits and its result is stored in the input VC state registers to be used by the subsequent flits of the packet. Its timing differs for flits routed in regular, AB and FT modes, as shown in Figure 6.2a. Bypassing (FT and AB) flits have a tighter time budget and therefore rely on their FC signals to carry their destination address. As FC bits increase the link width, FC carries only one destination address for only one of the two flits it represents. This decision limits each input to support bypassing of up to one header flit per cycle. Then, NRC of bypassing flits is performed after their FC is registered and before the new FC is created and sent to the downstream router. On the other hand, regularly routed header flits can carry their destination address in their data. Therefore, routing two incoming header flits in regular mode, as shown in Figure 6.2a, uses the destination address carried by the FC for the NRC of the latest arriving flit, and the destination address carried in the header flit for the earliest arriving one.

When bypassing (allocation or ST), NRC is in the critical path and the timing constraints are very tight, especially in FT-mode. So, when a bypassing mode is selected (FT or AB), the routing algorithm needs to be simple to fit in the available time budget (approximately 5-6 FO4 stages). In our implementation, we use XY routing, but any other (dimension-order) routing algorithm with similar complexity would suffice. To further simplify NRC for FT mode, FastTrackNoC implementation exploits the fact that with simple dimension-order routing, a single downstream router-ID can be hard coded at each input of a particular router corresponding to its opposite output port. This minimizes the NRC logic for FT-mode, e.g. for an 8×8 networks to a 6-input logic, as shown in Figure 6.4. This is not applicable to AB-mode because it needs to support flits sent from the local input port to any of the outputs and therefore the router-ID is not known a priori and therefore cannot be hardcoded.

On the contrary, when a flit is not bypassing then half a cycle is available for NRC, which allows more complex routing algorithms to be implemented. On one hand, allocation-bypassing and ST-bypassing are enabled when there is no congestion. On the other hand, advanced algorithms yield most of their performance benefits when there is contention and congestion. Therefore, FastTrackNoC could use more advanced algorithms when a header flit is routed in regular mode and fall back to dimension-order routing when bypassing, as long as deadlocks are avoided.

6.2.5 Flow Control and Minimum Buffer Size

FastTrackNoC uses credit based flow control, with one wire per VC indicating to the upstream router the release of VC credits and an additional wire per port to signal the release of two credits of one VC. Except for the FT- mode, all other modes of traversal

have one complete cycle available to transmit the credit upstream and to update the credit counters. FT-mode is different because it is activated in the middle of a clock cycle and it has only half of a cycle available to transmit the credit upstream and to update the credit counters, which is insufficient. In this case the credit is registered locally and then transmitted upstream in the next cycle. Finally, the latency of credit reuse differs for different routing modes: (i) in regular mode it is four cycles, (ii) in AB mode it is two cycles because flits skip allocation, and (iii) in FT mode, it is two cycles determined by the delay of credits transmission.

6.3 Zero-Load Latency Analysis

In order to show the potential of FastTrackNoC, we analyze the Zero-Load Latency (ZLL) of FastTrackNoC, and compare it with the best DDR and SDR networks, HighwayNoC and ShortPath [30], with parameters presented in Table 2.1 and used further in our evaluation in Chapter 7. The above are also compared with the theoretical minimum latency of a direct connection, registered at every hop, as describe in Chapter 1. Note that not registering such a direct connection at every hop could offer lower single flit latency, but high serialization latency and thus is considered worse. The analysis takes into account the minimum clock period of each network, 42 FO4 for FastTrackNoC and HighwayNoC, and 25 FO4 for ShortPath, as well as the theoretical latency of a direct connection that is 15.6 FO4 per hop, considering 2.1 mm tile dimension in 28 nm as explained in Chapter 7.

The ZLL of a FastTrackNoC packet is as follows: FC is received at local input at rising clock edge; in-network straight hops take half a cycle because of FT support - here an extra half a cycle is required when FC signals are received at falling clock edge at an odd hop to exit the network and need to wait until the next cycle to be considered; then, entering and exiting the network takes an additional half a cycle each because of using AB; in-network turns take 1.5 cycles more than a FT hop because they require regular-mode routing; again an extra half cycle is also required when FC is received at falling clock edge at an odd turning-hop and need to wait until the next cycle; finally, the serialization latency is half a cycle per flit. Considering all the above the worst case FastTrackNoC ZLL is:

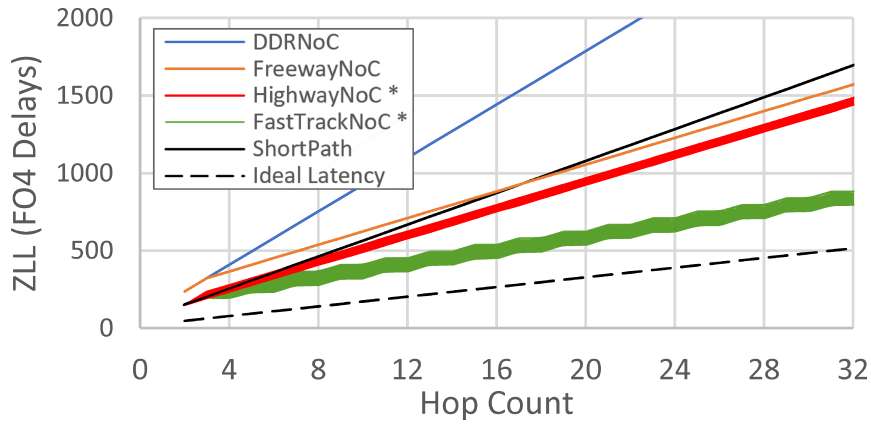
$$ZLL_{FastTrackNoC} = 1 + \lceil hops/2 \rceil + 1.5 \times hops_{turn} + odd\_hops_{turn}/2 + N/2 \; cycles$$

The ZLL of a packet in HighwayNoC and ShortPath networks is presented in Equations 5.1 and 3.2, respectively. Finally, the ideal latency of a direct connection is one cycle per hop, requiring one hop less than the previous cases because there is no delay in entering a network. In addition, it has serialization latency of one cycle per flit:

$$Ideal\ Latency = hops + N - 2 \; cycles$$

Taking into account the cycle time of these networks and the above ZLL equations, we perform a sensitivity analysis of the ZLL, measured in FO4 delays, with respect to the number of hops a packet traverses. We consider a 3-flit packet and for FastTrackNoC we show the margins of both the worst case ZLL describe above, as well as the best case of not having a turn. As shown in Figure 6.5, the ShortPath network scales worse than HighwayNoC due to its slower switching rate and its latency is up to 3.3× higher than the ideal latency for large number of hops. HighwayNoC operates its datapath at DDR offering higher switching rate, but cannot bypass ST offering 2.9×

higher ZLL than the ideal at high hop counts. Employing ST bypassing, FastTrackNoC reduces ZLL to half compared to HighwayNoC and ShortPath, closing the gap with the ideal direct connection to only 1.6-1.75× longer latency for 32 hops. More importantly, the ZLL of FastTrackNoC scales with the number of hops substantially better than the best previous SDR and DDR networks promising significant latency gains for large network sizes.



*A range of ZLLs is plotted, as ZLL for a particular number of hops varies depending on particular conditions, i.e., presence and timing of a turning hop.

Figure 6.5: ZLL analysis with respect to hop count of the FastTrackNoC, HighwayNoC and ShortPath NoCs vs. an ideal pipelined connection.

6.4 Timing Example

A timing example is described next of a 5-flit packet traversing three routers in a FastTrackNoC network illustrated in Figure 6.6. The example does not include entering or exiting of the packet and consider all three routers are aligned across one dimension. As a consequence, flits passing through the routers are not turning and hence have the opportunity to use all three modes of traversal (FT, AB and regular mode). The example is contracted with some artificial contention (marked in red color) to demonstrate all three routing modes. Note that LT and ST for flits that traverse the link in high clock phase are marked with "+", and for flits that traverse the link in low clock phase are marked with "-".

In cycle 0, router-1 receives the FC for the first two flits (H, B1), which carries flit types, assigned VC, next route, destination and an asserted FT_Capable bit indicating the flits are eligible for FT traversal. H and B1 flits arrive in a non-local input and are eligible to be routed both in FT and AB mode. This is checked in the beginning of cycle-1 and FT-mode is followed as it has higher priority allowing H and B1 to bypass both SA and ST stages. The data of H and B1 flits are stored in registers 0 and 1 of the empty input VC-0 buffer in the first and second half of cycle 1, respectively. As the first flit is a header, NRC is performed and an output VC-id is acquired from registered output of VC select (VS). Additionally, the *FT_Capable* bit is set because output VC-0 is allocated to the packet and it traverses an in-network straight hop in the downstream

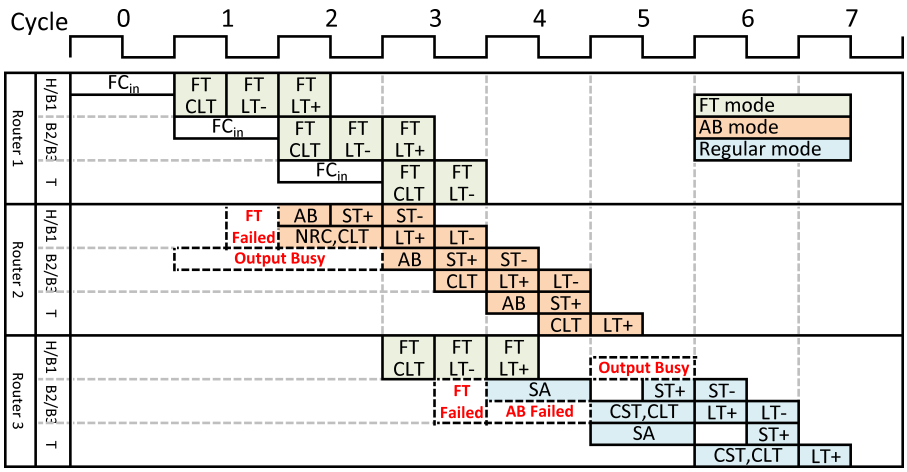


Figure 6.6: Timing of 5-flit packet through three FastTrackNoC routers.

router. Moreover, the *FT_Active* bit is set to indicate output FC signals should be registered at the falling edge of cycle 1 in router-2. After new FC is generated in router-1 using the result of NRC and the allocated output VC, it traverses the link to reach router-2 at the falling edge of cycle 1. In the second half of cycle 1, the H flit stored in VC-0, register 0 uses FT path to traverse the link. B1 follows half a cycle later and the remaining flits are routed through router-1 similarly in cycles 2 and 3.

Following again H and B1 flits, their control information arrives in router-2 at the falling edge of cycle 1 while the data flits are received in the second half of cycle 1 and first half of cycle 2. During the second half of cycle 1, it is assumed that these flits are unable to initiate FT-mode in router-2 because of a busy output port. The flits attempt to utilize the AB-mode in the next cycle. In first half of cycle 2, AB check for these flits is successful, indicating no contention on router switch and link, allowing SA to be bypassed. In parallel, to AB check, NRC is performed, output VC-id acquired from VS and the *FT_Capable* bit is set because once again output VC-0 has been allocated to the packet and it traverses an in-network straight hop to the downstream router. The updated FC signals bypass the regular CST and go directly to control link traversal (CLT) reaching router-3 at the beginning of cycle 3. In the second half of cycle 2, the H flit traverses the switch of router-2 (ST+) and is registered (in *Reg+*) before it performs LT in the first half of cycle 3 (LT+). B1 flit follows half a cycle later. It performs ST and LT in the first and second half of cycle 3, respectively (ST-, LT-). The remaining flits of the packet cannot use the FT-mode in router-2 either because the output port is busy transferring the previous flits of this packet, e.g., H and B1 flits use LT of router-2 in cycle 3, which would be needed for a FT traversal of B2 and B3. As a consequence, the remaining flits of the packet are also routed through router-2 as H and B1 flits using AB-mode.

Finally, the control information of H and B1 flits arrives in router-3 at the end of cycle 2 while their data flits are received in cycle 3. The *FT_Capable* bit in the received FC is set and the router-3 is FT capable for the respective input-output ports. Therefore, these flits are routed using FT-mode in a manner similar to router-1. Updated FC signals for H and B1 are generated and traverse the link (CLT) in the first half of cycle

3. H and B1 flits traverse router-3 and LT in FastTrack in the second half of cycle 3 and first half of cycle 4, respectively. Assuming lack of credits in the used VC downstream, flits B2 and B3 cannot be routed using the FT-mode. Then, they attempt AB in cycle 4, which also fails because output port appears busy during cycle 5. Allocation requests for flits B2 and B3 are then sent to SA in cycle 4. After successful SA, the FC of these flits undergo switch traversal and link traversal (CST, CLT) in cycle 5. Subsequently, the actual B2 flit undergoes ST in the second half of cycle 5 (ST+) and LT in the first half of cycle 6 (LT+) and B3 flit follows half a cycle later. Finally, T flit received in the first half of cycle 5 cannot bypass any router-3 stage because the used VC still contains previously buffered flits. So, flit T requests SA and is routed similar to B2 and B3 flits.

6.5 Summary

This Chapter described the FastTrackNoC router architecture. FastTrackNoC exploits the fact that the direction of traffic through a NoC router is biased. It offers a bypassing path between each input port and its preferred output and in the absence of contention allows to skip switch traversal stage. In the best case, FastTrackNoC allows incoming flits to directly proceed to link traversal after they are buffered at the input substantially, reducing packet latency. FastTrackNoC applies ST bypassing to a NoC with DDR datapaths to improve its switching rate and hence increase network throughput.

Chapter 7

Evaluation

Now that the design details of the proposed DDR NoC architectures have been presented, we compare the performance, area and power consumption of these networks with state-of-the-art ShortPath [30] network. ShortPath improves network throughput by simplifying allocation to increase clock frequency and reduces minimum hop latency by implementing dynamic pipeline bypassing. This makes ShortPath one of the fastest single data rate (SDR) NoCs in the literature.

In this Chapter, we first discuss the experimental setup and present post place-and-route (P&R) implementation results for operating frequency and silicon area of the networks being analyzed. Then we evaluate and compare the performance and energy characteristics of DDR NoC architectures using synthetic traffic as well as traces of application-driven workloads.

7.1 Experimental Setup

In order to reliably evaluate the proposed DDR networks and compare them to the ShortPath network, they were fully implemented in Register Transfer Level (RTL) abstraction to accurately determine their operating frequency, area and power consumption. A simple 4-stage Baseline SDR NoC [24] was also designed and implemented for performance evaluation during the early phase of the DDRNoC design. All networks were also modeled in SystemC at a cycle accurate level to obtain performance results for longer simulations. Table 7.1 lists the implementation details of all NoC routers.

The networks were implemented in 28nm CMOS FDSOI (Fully Depleted Silicon on Insulator) 1.10V standard cell technology. The designs were placed and routed with Cadence Design Systems Innovus Implementation System 17.1. We considered square tiles with 2.1 mm long sides based on the Chip multiprocessors parameters used by Sewell et al., after scaling down to 28 nm (CPU core with 32kB L1 I- and D-cache and a 512kB L2 cache slice) [55]. The length of the local links is considered to be 0.5 mm. Finally, the registers in the datapath support clock gating to reduce power consumption when idle.

Power analysis is performed by simulating post-P&R netlists of the NoCs in QuestaSim with back-annotated delays. Then, gate-level switching activity for each router in a network is recorded in a VCD file which is then used to get power estimates of the entire NoC using Synopsys PrimeTime PX.

Table 7.1: Implementation parameters of the evaluated NoC architectures.

Design	DDRNoC	FreewayNoC	HighwayNoC	FastTrackNoC	ShortPath [30]	Baseline [24]
Router architecture	2-stages: ST, LT. Control forwarding				4-stages: VA, SA1, SA2/ST, LT	4-stages: VA, SA, ST, LT
ST bypassing	No			From non-local input to non-local straight output	No	No
Allocation bypassing	No	From non-local input to straight output	(i) Non-local input to straight output (ii) Non-local input to local output (iii) Local input to non-local output		Dynamic pipeline bypassing from all inputs to all outputs	No
Flow control	Credit based flow control					
Link width	128b data, 4b+1b credits FC:-2×3b flit type, 2×2b VC-id, 2×2b N^2RC **Total:-** 147	128b data, 4b+1b credits FC:-2×3b flit type, 2×2b VC-id, 2×2b NRC, 2× 6b dest. addr. **Total:-** 159	128b data, 4+1b credits FC:- 3b flit type[1], 2×2b VC-id, 2×2b NRC, 6b dest.[2] **Total:-** 150	128b data, 4+1b credits FC:- 1b FT_Active, 1b FT_Capable, 3b flit type [1], 2×2b VC-id, 2×2b NRC, 6b dest.[2] **Total:-** 152	128b data, 4b flit-credits, 1b pkt-credit, 3b flit type, 2b VC-id **Total:-** 138 bits	128b data, 4b flit-credits, 3b flit type, 2b VC-id **Total:-** 137 bits
Cycles (FO4 delay) per hop	2 (84)	Min: 1 (42), Max: 2 (84)	Min: 1 (42), Max: 2 (84)	Min: 0.5 (21), Max: 2 (84)	Min: 2 (50), Max: 4 (100)	Min: 3 (93), Max: 4 (124)
VC Config.	4 VCs per input port.					
Buffer size	5-flit flip-flop based VC buffers. [3]					
VC allocator	Output-first separ. alloc., RR, PV:1 out-arb., V:1 in-arb.	VC Select with V:2 arbiter, RR priority.		VC Select with V:2 arbiter, fixed priority.	V:1 in-arb., P:1 out-arb. VC alloc. ReqQ depth = 8.	Output-first separable alloc., RR, PV:1 out-arb., V:1 in-arb.
Switch allocator	Speculative output-first separ. alloc., RR priority, PV:2 out-arb., V:1 in-arb.	Out-first separable alloc., RR priority, PV:2 out-arb., V:1 in-arb.	Input-first separable allocator, RR priority, V:2 in-arb., P:1 out-arb.		2-stage pipelined: SA1: V:1 in-arb., SA2: P:1 out-arb. SA ReqQ depth = 2.	Speculative output-first separable alloc., RR priority, PV:1 out-arb., V:1 in-arb.
Routing	XY routing with N^2RC	XY routing with NRC				

[1]For non-header flits, NRC bits are reused to distinguish body/tail flits.
[2]This destination address field indicates the destination address of one header flit to be used for NRC.
[3]This is sufficient for ShortPath as it has a credit-round-trip-time of 5 cycles and for DDR NoCs as they handle up to 5 flits packets.

Table 7.2: Post place & route implementation results.

Voltage	Design	Area (# Gates)	Max. Freq. (GHz)	FO4 delay
1.10 V	DDRNoC	167 185	1.53	21*
	FreewayNoC	177 984	1.53	21*
	HighwayNoC	170 688	1.53	21*
	FastTrackNoC	178 654	1.53	21*
	ShortPath	157 810	2.56	25
	Baseline	161 290	2.1	31
0.95 V	DDRNoC	134 879	1.15	28*
	Baseline	135 564	1.76	37

* Half cycle FO4 delay is reported for DDR NoC routers, which is the delay for a ST in DDRNoC, FreewayNoC and HighwayNoC networks and for the FastTrack path in FastTrackNoC. Moreover, for the considered tile size of $2.1 \times 2.1 mm^2$, the delay of FastTrack path and ST is similar.

Performance is measured by injecting synthetic traffic as well as application-driven traffic into the network. Synthetic traffic injects data packets of 80 bytes (5 flits) and control packets of 16 bytes (1 flit) with the following traffic patterns: (i) uniform random (UR), (ii) hotspots (HS) with 25% of the traffic going to 4 hotspots, one at each NoC corner, and the rest of the traffic being uniform random, (iii) nearest neighbor (NN), and (iv) bit reverse (BR). In addition, traffic traces based on application driven workloads are obtained using SynFull [56]. These traces capture the application behavior of PARSEC [57] and SPLASH-2 [58] benchmarks including messages generated by the cache coherence protocol and message dependencies. Simulations run until completion, all below 100 million cycles, generating control and data packets of 16 and 80 bytes, respectively, for a 32 node (4×8) network. In these experiments, average packet latency is measured per benchmark.

7.2 Implementation Results

Table 7.2 summarizes the post P&R results of a single router for all networks at 1.10V, as well as results for the DDRNoC and the Baseline network at 0.95 V [4]. At 1.10 V, balancing ST and LT delays, the delay of the DDRNoC, FreewayNoC and HighwayNoC datapath stages is 327 ps. The target cycle time of these networks is then double the above delay as two flits per cycle traverse ST (or LT). Post P&R results confirm that these networks have a clock period of 654 ps because they do not have control in the critical path as it fits in the target clock period. For the FastTrackNoC the critical path is the FastTrack path, shown in Figure 6.1, which is 327 ps (similar to ST delay). Thus the FastTrackNoC also has a clock frequency of 1.53 GHz, similar to other DDR networks. According to Psarras et al., ShortPath's control logic is in the critical path; more precisely its cycle time is defined by the delays of credit-check, an N:1 arbiter (for the second part of SA), two 2:1 multiplexers and the crossbar delay [30]. Our physical implementation of the ShortPath router in 28 nm technology confirms the above, indicating that the maximum operating frequency of ShortPath is 2.56 GHz. The Baseline network has a clock period of 476 ps (2.1 GHz), with VA in the critical path. Baseline network clock is slower than ShortPath because ShortPath simplifies VA and SA by using allocation request queues [30]. At 0.95 V,

[4] Implementation at 1.10 V is always assumed for all networks, unless otherwise stated.

the DDRNoC operates at 1.15 GHz and it does not have control logic in the critical path, whereas Baseline network operates at 1.76 GHz with VA in its critical path.

At 1.10 V, DDRNoC requires up to 6.3% more gates compared to a ShortPath router, mostly due to extra bits required in the paths for forwarding control information, an additional register-multiplexer pair per output port and more complex allocation logic. FreewayNoC, which implements simplified allocation bypassing (AB), requires an additional 6% gates. HighwayNoC, which enhances AB support in the the FreewayNoC to also include the local port, requires 4% less gates than the FreewayNoC because it simplifies VA and SA. FastTrackNoC, which supports both SA and ST bypassing, requires 4.7% and 13.3% more gates than the HighwayNoC and ShortPath routers, respectively, because it requires intra-router bypass paths for flits and forwarded control bits. Finally, the SDR Baseline requires 2.5% more gates compared to ShortPath because it uses more complex VA and SA.

Operating Frequency for Higher Radix Routers

Our implementation focuses on 2D-mesh networks. However, interesting questions arise regarding the impact of the router's radix on the operating frequency of the design. For instance, it is crucial for the performance of the proposed DDR networks that the critical path of a router remains in the datapath. In order to verify that, we implemented the DDR NoC routers and the ShortPath router with radices 3×3, 5×5, 9×9, and 17×17 keeping the cell/area utilization constant to 50% and measured their maximum operating frequency. Figure 7.1 plots the results showing the clock period of the ShortPath and, for better comparison, half of the clock period of the proposed DDR networks; that is for all the networks the latency of a flit performing a switch traversal. As it can be observed, the frequency of the DDR networks scales better with higher radices than the ShortPath. More precisely, ShortPath and DDR networks get affected by the higher wire congestion in higher radix routers, however, the critical path of the DDR networks still remains in the datapath (crossbar) and is decoupled from the control. For a DDR NoC router, the control logic does not become slower than its datapath at any router radix, therefore, its operating frequency depends only on the way the datapath delay scales. On the contrary, ShortPath's critical path includes parts of control (credit-check and switch allocation) and ST (output multiplexing) in sequence. The delay of these components of a ShortPath router increases with higher router radices (at least linear to the number of inputs), and results in worse scalability compared to the proposed DDR NoCs.

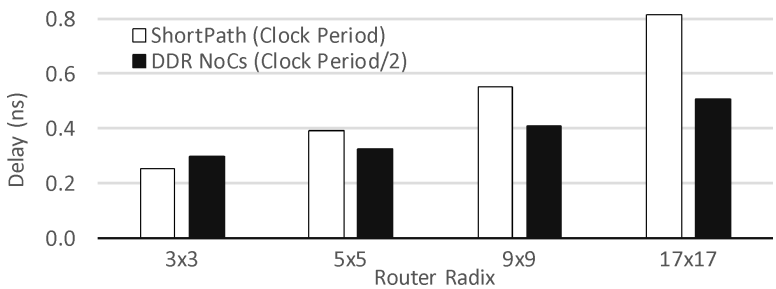


Figure 7.1: Scalability of ShortPath's clock period and, for better comparison, half clock period of a DDR NoC to the router radix.

7.3 Performance Evaluation

7.3.1 Performance with Synthetic Traffic

The performance of DDR and ShortPath networks of 8×8, 16×16 and 32×32 sizes is measured using four different traffic patterns: (i) uniform random (UR), (ii) hotspots (HS), (iii) nearest neighbor (NN), and (iv) bit reverse (BR). All networks are operated at their maximum clock frequencies, as presented in Table 7.2.

Figures 7.2a, 7.3a, 7.4a and 7.5a show the performance of 8×8 networks for UR, HS, BR and NN traffic patterns, respectively. The throughput of the DDR networks is 16-20% higher compared to the ShortPath network for all traffic patterns. DDR NoCs gain throughput because their DDR datapath can switch every half a clock cycle (327 ps) as opposed to ShortPath network datapath which switches every cycle (390 ps). At low injection rates, average latency of packets through the DDRNoC is 58-63% higher than the ShortPath network, because minimum hop latency for DDRNoC (84 FO4 delays) is higher than the ShortPath (50 FO4 delays), as presented in Table 2.1. FreewayNoC, which implements AB for flits propagating an in-network straight hop, offers 17-25% lower and 21-35% higher packet latency compared to DDRNoC and ShortPath, respectively, for all traffic patterns except NN where it offers latency similar to DDRNoC because traffic sent to immediate neighbours cannot utilize simplified AB. HighwayNoC, which extends AB to also support the local port, reduces average packet latency by 16-24%, compared to the FreewayNoC, for all traffic patterns except NN where latency is reduced by 33%. Compared to ShortPath, HighwayNoC has 1-3% higher packet latency. FastTrackNoC reduces packet latency by 11-13% compared to HighwayNoC for UR, HS and BR traffic patterns. This is attributed to the FT-mode traversal supported in FastTrackNoC. As indicated in Figure 7.6a for UR traffic, FastTrackNoC uses FT-mode in 45-50% of the hops and AB mode for another 36-39%, while HighwayNoC offers only AB mode for 83-85% of the cases. For NN traffic, the average packet latency through FastTrackNoC is similar to the HighwayNoC because traffic is sent to immediate neighbours and cannot utilize FT-mode of traversal. Similarly, compared to ShortPath, FastTrackNoC packet latency is 9-11% lower for UR, HS and BR traffic and for NN traffic average latency is 3% higher.

Figures 7.2b, 7.3b, 7.4b and 7.5b show the performance of 16×16 networks for UR, HS, BR and NN traffic patterns, respectively. The throughput of the DDR networks is 15-19% higher compared to the ShortPath network for all traffic patterns. At low injection rates, average latency of packets through the DDRNoC is 37-55% higher than the ShortPath network, for all traffic patterns. FreewayNoC offers 33-35% lower and 4-10% higher packet latency compared to DDRNoC and ShortPath, respectively, for all traffic patterns except NN where it offers latency similar to DDRNoC (which is 53% higher than ShortPath). HighwayNoC packet latency is 8-12% lower than FreewayNoC and 4-5% lower than ShortPath for traffic patterns with high average hop count (UR, HS and BR). For NN traffic, HighwayNoC latency is 30% lower compared to FreewayNoC and 4% higher compared to ShortPath. FastTrackNoC packet latency is 17-21% lower than HighwayNoC and 20-25% lower than ShortPath for traffic patterns with high average hop count (UR, HS and BR). For NN traffic, FastTrackNoC latency is similar to the HighwayNoC and 4% higher than ShortPath. We observe that the latency of FreewayNoC, HighwayNoC and FastTrackNoC improves relative to DDRNoC and ShortPath with increasing network size where flits have a higher

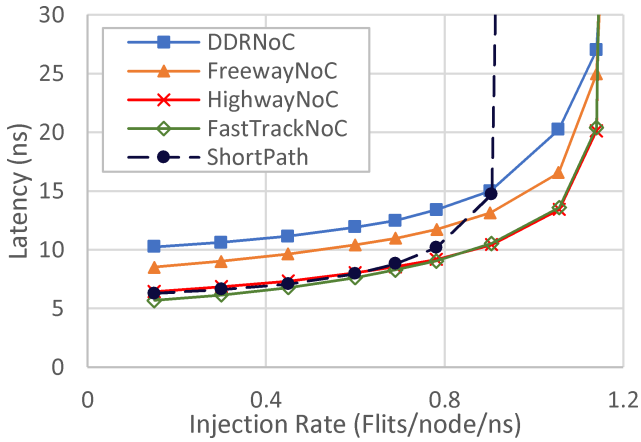
average hop count. This confirms our ZLL analysis in Sections 5.3 and 6.3, which showed that packet latency in FreewayNoC scales better with increasing hop count compared to DDRNoC and ShortPath. Similarly, HighwayNoC latency scales better than FreewayNoC and FastTrackNoC latency scales better than HighwayNoC.

Figures 7.2c, 7.3c, 7.4c and 7.5c show the performance of 32×32 networks for UR, HS, BR and NN traffic patterns, respectively. Here, again, the throughput of the DDR networks is 16-19% higher compared to the ShortPath network for all traffic patterns. At low injection rates, average latency of packets through the DDRNoC is 54-70% higher than the ShortPath network, for all traffic patterns. FreewayNoC offers 38-45% lower and 3% higher to 4% lower packet latency compared to DDRNoC and ShortPath, respectively, for all traffic patterns except NN where it offers latency similar to DDRNoC (which is 54% higher than ShortPath). HighwayNoC packet latency is 7-12% lower than FreewayNoC and 10% lower than ShortPath for traffic patterns with high average hop count (UR, HS and BR). For NN traffic, HighwayNoC latency is 32% lower compared to FreewayNoC and 4% higher compared to ShortPath. FastTrackNoC packet latency is 30-32% lower than HighwayNoC and 38-40% lower than ShortPath for traffic patterns with high average hop count (UR, HS and BR). For NN traffic, FastTrackNoC latency is similar to the HighwayNoC and 4% higher than ShortPath.

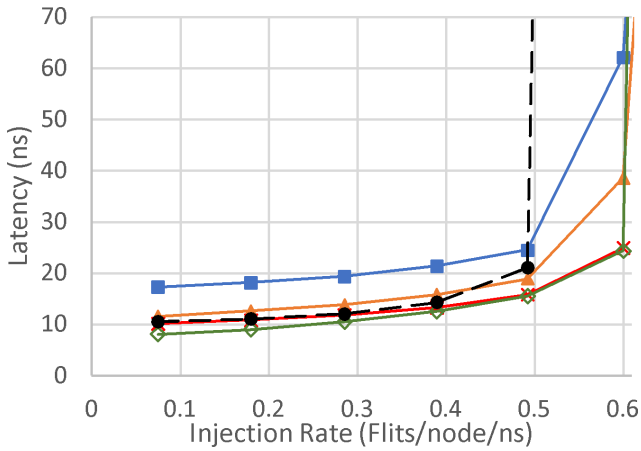
Figure 7.6 summarizes for different network sizes, the percentage of hops that bypass some router stage in the four competing NoCs which support bypassing. In particular, for FreewayNoC and HighwayNoC the AB-mode bypassing hops are measured, for FastTrackNoC the FT-mode and AB-mode bypassing hops and for ShortPath hops which bypass one or two of the 4 stages (including LT) are recorded. FreewayNoC has the lowest percentage of bypassing cases compared to other networks because it does not support AB for in-network turns and when entering and exiting the network. HighwayNoC and FastTrackNoC have lower percentage of accumulated bypassing cases compared to ShortPath as they support bypassing only for non-turning flits, however their bypassing yields higher latency gains and so offers lower average packet latency. That is especially evident for FastTrackNoC where the FT-mode is a large fraction of its recorded bypassing cases, especially for larger networks and lower injection rates. For instance, it is work noting that in a 32×32 FastTrackNoC 45-80% of the traffic can use the FastTrack path in low and medium injection rates.

7.3.2 Performance Comparison against Networks with Longer Links

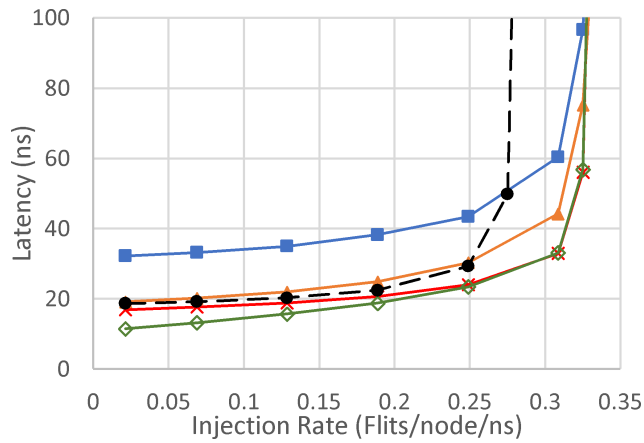
In the previous evaluation, the considered tile size in 28 nm technology is $2.1 \times 2.1 \text{mm}^2$ as explained in Section 7.1, which fits a regular size core including L1 and its L2 slice. FastTrackNoC is able to support this tile size without reducing its maximum operating frequency, restricting the router to $\frac{1}{16}$ of the tile area. In other words, for $2.1 \times 2.1 \text{mm}^2$ tiles the FastTrackNoC LT delay, including the FastTrack logic overhead, is similar to the ST delay and therefore results in a well balanced design. In comparison SMART NoC considers 1mm tile dimension in 45nm [26]. Still, NoC routers with slower switching rate than DDR networks (1/327ps) could support longer links. For example, ShortPath could handle up to 4.6mm tile dimensions, without pipelining the link.



(a) Network size: 8×8

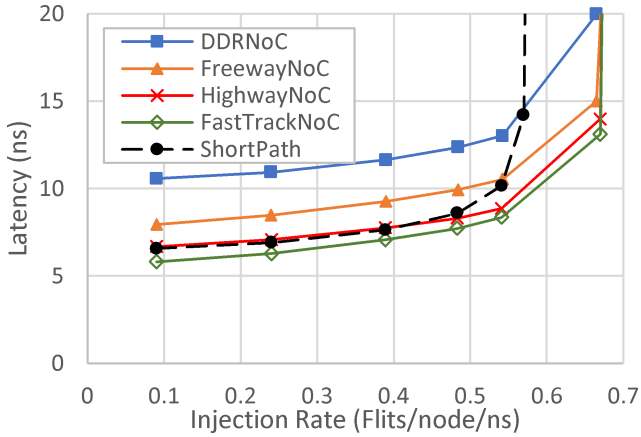


(b) Network size: 16×16

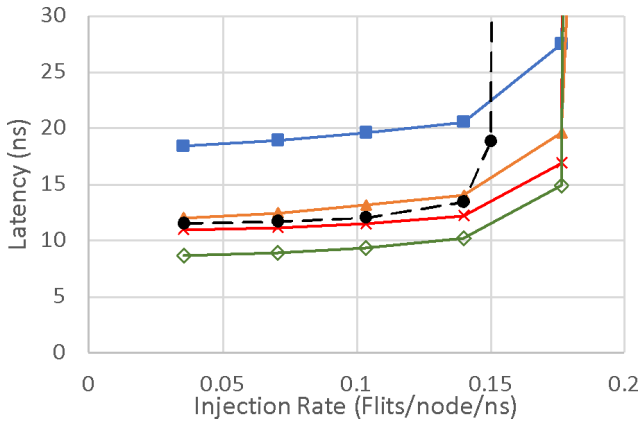


(c) Network size: 32×32

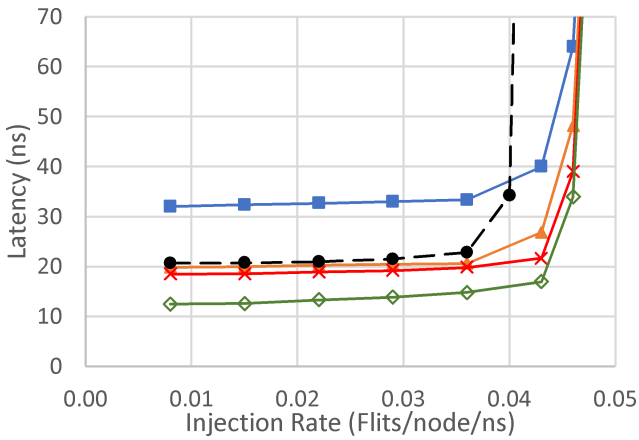
Figure 7.2: Average packet latency with uniform random (UR) traffic pattern for 2D-mesh networks of different sizes.



(a) Network size: 8×8

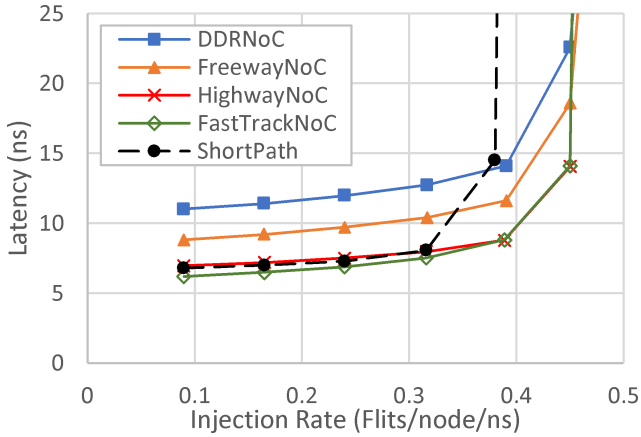


(b) Network size: 16×16

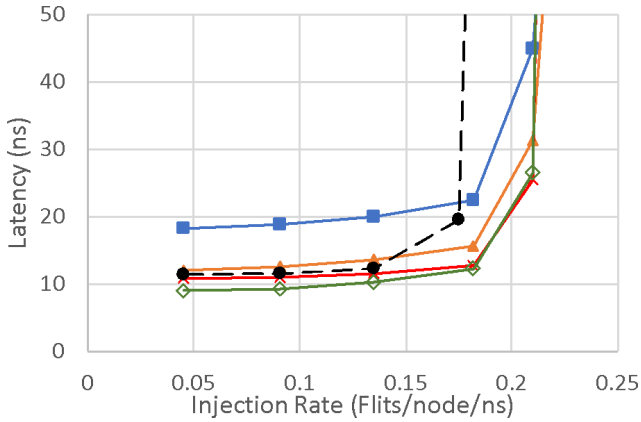


(c) Network size: 32×32

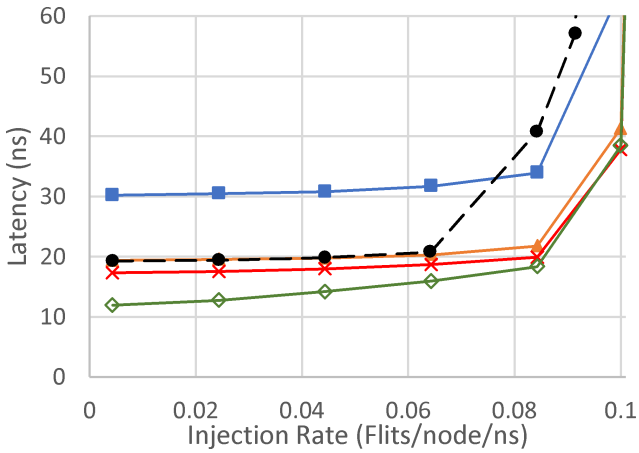
Figure 7.3: Average packet latency with hotspot (HS) traffic pattern for 2D-mesh networks of different sizes.



(a) Network size: 8×8

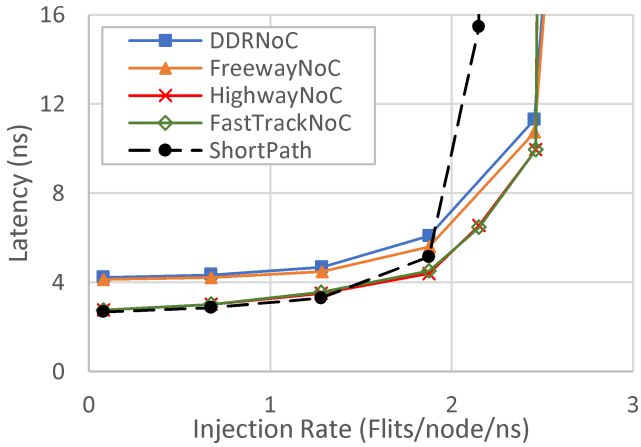


(b) Network size: 16×16

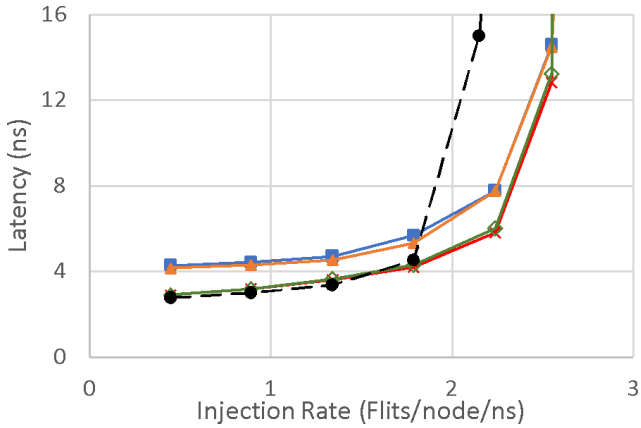


(c) Network size: 32×32

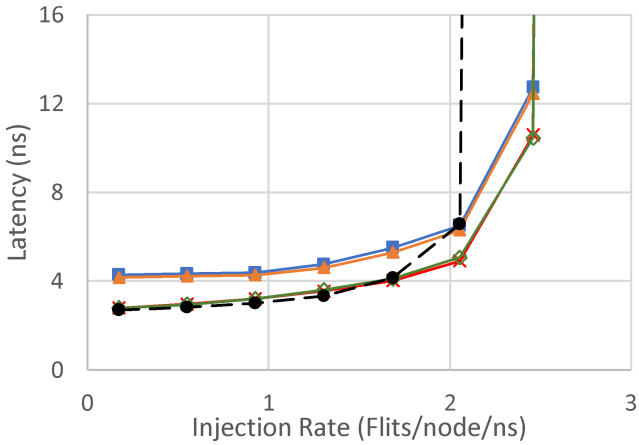
Figure 7.4: Average packet latency with bit reverse (BR) traffic pattern for 2D-mesh networks of different sizes.



(a) Network size: 8×8

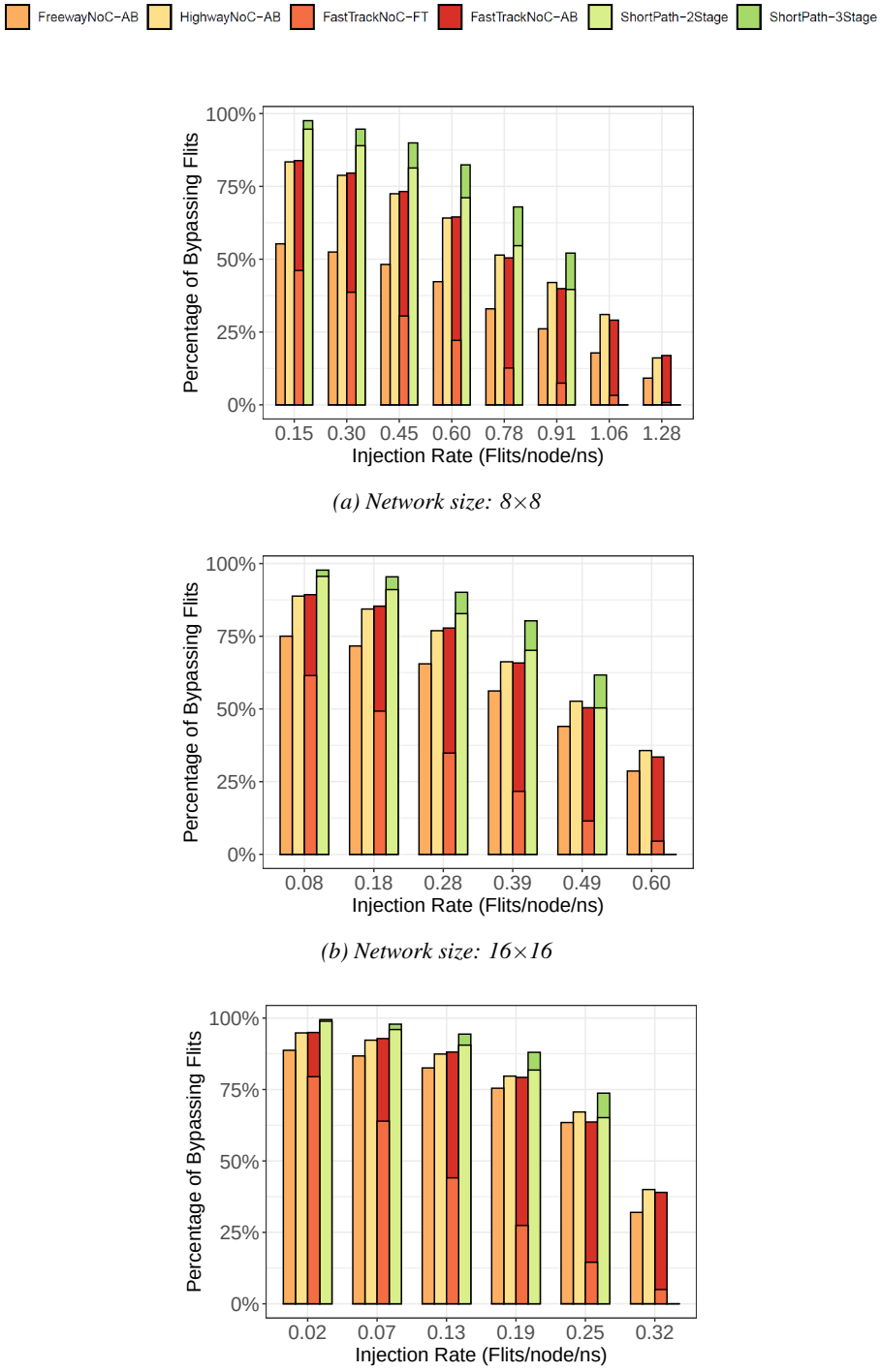


(b) Network size: 16×16



(c) Network size: 32×32

Figure 7.5: Average packet latency with nearest neighbour (NN) traffic pattern for 2-mesh networks of different sizes.



(a) Network size: 8×8

(b) Network size: 16×16

(c) Network size: 32×32

Figure 7.6: Percentage of bypassing flits in FreewayNoC, HighwayNoC, FastTrackNoC and ShortPath networks for UR traffic. Bars are not shown for ShortPath network after saturation.

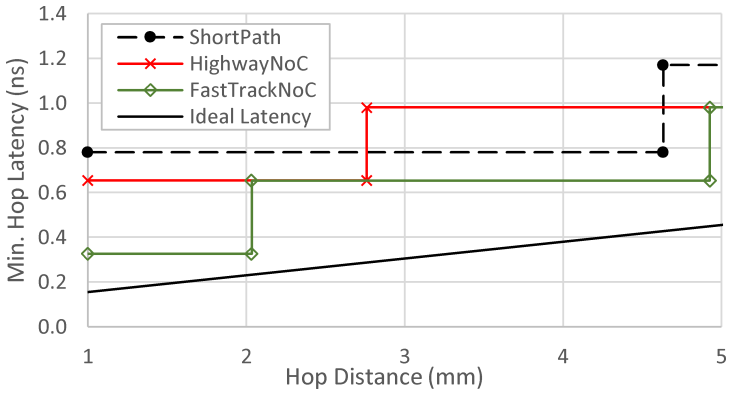
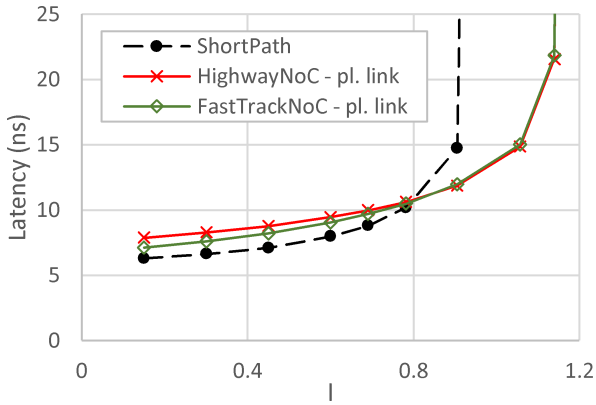


Figure 7.7: Minimum hop latency with respect to hop distance for HighwayNoC, FastTrackNoC and ShortPath networks as well as for an ideal un-pipelined link with registered end-points.

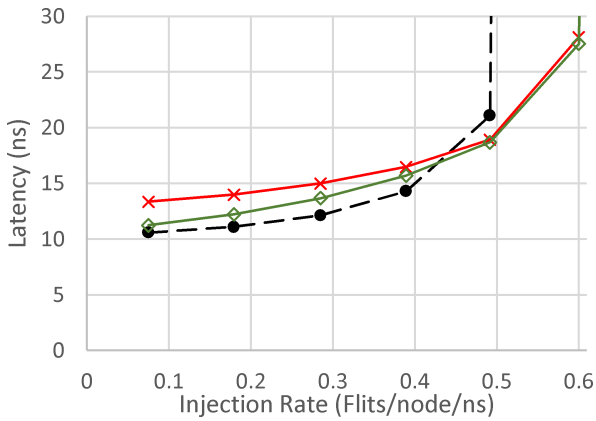
In order to transfer flits over a longer hop, links need to be pipelined to avoid slowing down network clock frequencies and to maintain their throughput. Under this assumption, Figure 7.7 shows the minimum hop latency of the networks for different tile dimensions (hop distances). Although FastTrackNoC would need to pipeline its link for tile dimensions smaller than the other networks, it can still offer lower minimum hop latency over long distances due to its ST bypassing support.

Next, we evaluate the networks considering that the LT delay is equal to the ShortPath cycle time (390ps); this would add roughly 2.5 mm to the link length leading to tile size of $4.6\times4.6\text{mm}^2$. Then, the HighwayNoC and the FastTrackNoC networks require one DDR pipeline stage on the inter-router flit and FC link wires, as indicated in Figure 7.7. Wires transmitting credits upstream are not pipelined because they operate at SDR and can traverse tile dimension in a single cycle. Similarly, link wires at the local port are also not pipelined considering there is a shorter link (relative to inter-router links) connecting the local port of a router to the network interface.

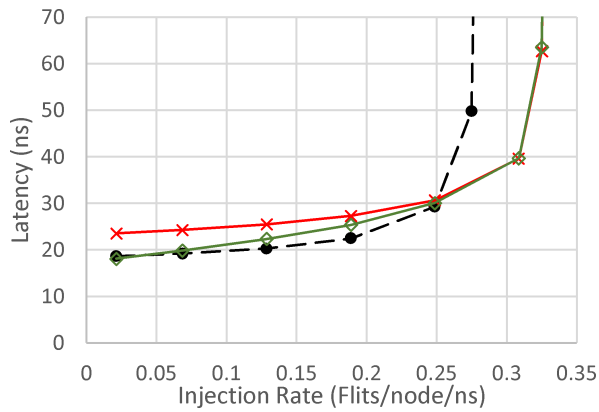
Figure 7.8 shows the performance of the HighwayNoC, FastTrackNoC and ShortPath with long links for 8×8, 16×16 and 32×32 networks and with uniform random traffic. The throughput of the DDR NoCs remain unaffected because they operate at the same clock frequency as when the links are shorter and not pipelined. So, HighwayNoC and FastTrackNoC have 17-19% higher throughput compared to ShortPath network. The latency of the DDR networks increases because flits need to spend an extra half a clock cycle per hop. Increase in latency is more pronounced at low injection rates where additional delay of pipelined link causes average packet latency to become higher than the ShortPath for 8×8 and 16×16 networks. Average latency of HighwayNoC is about 25% higher than ShortPath at low injection rates for all three network sizes. Average latency of FastTrackNoC is 13% and 6% higher than ShortPath at low injection rates for 8×8 and 16×16 network sizes, whereas it is 3% lower for 32×32 network. In summary, pipelined longer links reduce the latency advantage of the FastTrackNoC over ShortPath and force it to have up to 13% higher latency in smaller network sizes. However, as the number of hops increase, FastTrackNoC latency scales better than ShortPath because flits can more effectively utilize the FT-mode of traversal.



(a) Network size: 8×8



(b) Network size: 16×16



(c) Network size: 32×32

Figure 7.8: Average packet latency for UR traffic through HighwayNoC, FastTrackNoC and ShortPath networks considering large tiles of size 4.6×4.6mm^2, which is defined by the cycle time of the ShortPath router (390 ps). Inter-router links are pipelined for HighwayNoC and FastTrackNoC.

7.3.3 Performance Comparison against Networks with DDR Links

Another existing technique for improving network performance has been to time-multiplex two SDR sub-networks on a shared link operating at DDR, called RapidLink [17, 32]. Assuming each link is fast enough to carry two flits in a single cycle without compromising network clock frequency, applying RapidLink to an SDR network, such as ShortPath, can improve network throughput. We compare the performance of RapidLink network with a FastTrackNoC design that exhibits similar router datapath modifications. More, precisely, we consider a FastTrackNoC with two sub-networks (denoted as FastTrackNoC-2SubNet); its routers are split into two sub-routers (each with 4 VCs) similar to RapidLink, while the links are replicated as they already operate at DDR and cannot be shared by the sub-routers. In 28nm technology used for our implementation, replicated links can be supported in the low resistance intermediate Bx metal layers using a small fraction of the available wires – 5% to 10% depending on spacing and shielding. Figure 7.9 shows the average packet latency through 8×8 2D-mesh RapidLink and modified FastTrackNoC networks with UR traffic pattern. Compared to the RapidLink version of ShortPath NoC, a FastTrackNoC with two sub-networks offers about 10% lower packet latency at low traffic injection rate and supports about 22% higher throughput. That is similar latency and throughput gain as a regular FastTrackNoC versus a regular ShortPath network, as presented in 7.3.1.

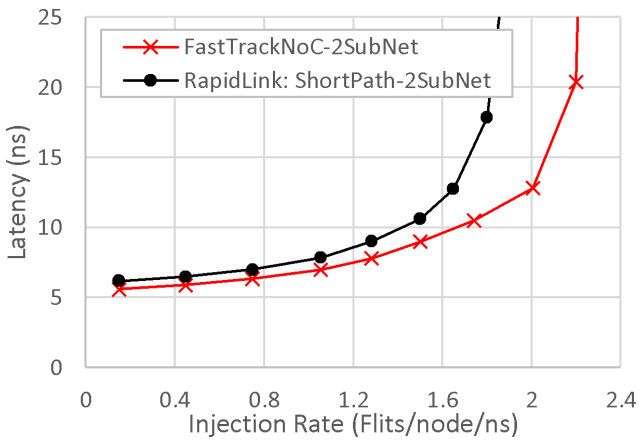
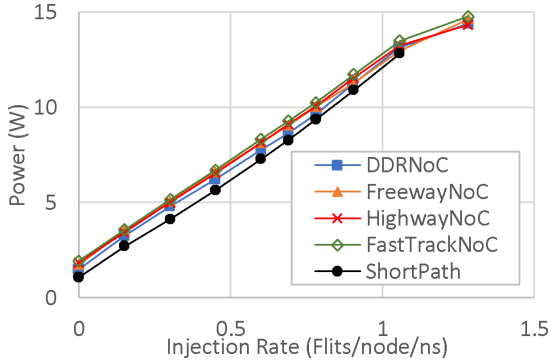


Figure 7.9: Average packet latency through RapidLink and FastTrackNoC with two sub-networks. Network size of 8×8 and UR traffic pattern is considered.

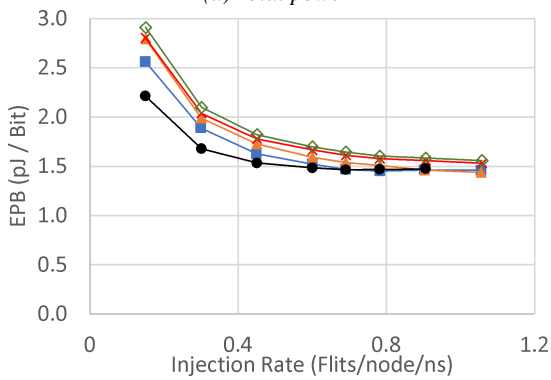
7.3.4 Energy Efficiency

Figures 7.10 and 7.11 summarizes the energy efficiency results on an 8×8 2D-mesh with UR traffic for DDRNoC, FreewayNoC, HighwayNoC, FastTrackNoC and ShortPath networks. The following are measured: total power, energy per transferred bit (EPB), energy-delay product (EDP), and energy throughput ratio (ETR).

Compared to ShortPath network, DDRNoC power is about 20% higher at low injection rates because it uses a DDR datapath, more complex control logic and lookahead signalling. However, at the saturation throughput of ShortPath, DDRNoC



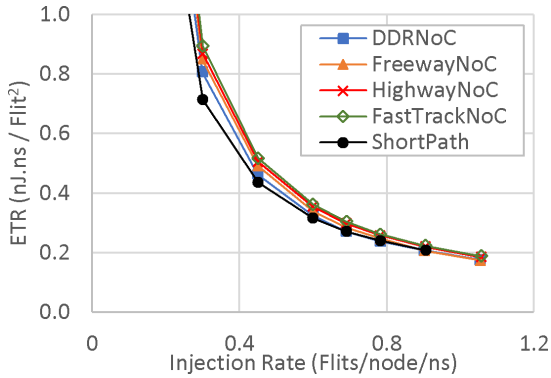
(a) Total power



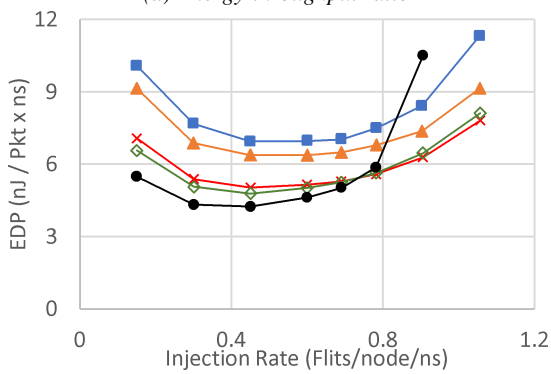
(b) Energy per bit

Figure 7.10: Power and energy per transferred bit costs of 8×8 DDR NoCs and ShortPath network with uniform random traffic.

power is only 2.3% higher. EPB and ETR with DDRNoC are up to 16% higher at low traffic but similar at the saturation throughput of ShortPath. EDP is up to 83% higher with DDRNoC at low injection rates because of its high minimum hop latency (84 FO4 delays) relative to ShortPath (50 FO4 delays). FreewayNoC power is up to 31% higher than ShortPath at low injection rates, but it is only 3% higher at the saturation throughput of ShortPath. EPB and ETR with FreewayNoC are up to 31% higher at low injection rate but similar at the saturation throughput of ShortPath. EDP is up to 67% higher with FreewayNoC at low injection rates. HighwayNoC power is up to 26% higher than ShortPath at low injection rates. However, at the saturation throughput of ShortPath, HighwayNoC power is only 5.7% higher. EPB and ETR with HighwayNoC are up to 26% higher at low injection rate and 5% higher at the saturation throughput of ShortPath. EDP is up to 28% higher with HighwayNoC at low injection rates. FastTrackNoC has up to 31% higher power than ShortPath at low injection rates because it has FastTrack bypass paths for flits and forwarded control bits. However, at the saturation throughput of ShortPath, FastTrackNoC power is only 5.2% higher as the two networks converge to the same percentage of bypassing cases. EPB and ETR with FastTrackNoC are up to 31% higher at low injection rate but similar at the saturation throughput of ShortPath. At the saturation throughput of FastTrackNoC, its ETR is 14% lower than ShortPath. EDP is up to 19% higher with



(a) Energy-throughput ratio



(b) Energy-delay product

Figure 7.11: Energy-throughput ratio and energy-delay product of 8×8 DDR NoCs and ShortPath network with uniform random traffic.

HighwayNoC at low injection rates.

Compared to DDRNoC, FreewayNoC consumes 10% more power at low injection rates and similar power at saturation throughput. This is also reflected in EPB and ETR, both of which increase for the FreewayNoC by about 10% compared to the DDRNoC. The EDP of FreewayNoC is up to 12% lower than DDRNoC at low injection rates because FreewayNoC supports AB for flits propagating a straight hop. HighwayNoC consumes up to 3% more power compared to FreewayNoC. This is also reflected in EPB and ETR, both of which increase by about 3% for the HighwayNoC. EDP of HighwayNoC is 48-56% lower compared to FreewayNoC. Compared to the HighwayNoC, FastTrackNoC power consumption is 5% higher when idle, 3.7% higher at low injection rates and 1.6% higher at high injection rates, mostly due to the FastTrack bypass paths. This is also reflected in the energy per bit and ETR, both of which increase by 1.6% to 3.7% for the FastTrackNoC. Finally, EDP of the FastTrackNoC is up to 8% lower at low injection rates, in spite of the higher power consumption, because of the significant latency reduction achieved by bypassing SA and ST stages. At the saturation throughput of FastTrackNoC, where FT-mode is active for less than 3% of the hops (as depicted in Figure 7.6a), EDP increases by about 3%.

7.3.5 Trading DDRNoC Performance Advantage for Energy Efficiency

During the early phases of the DDRNoC design we also performed physical implementation of SDR Baseline and DDRNoC networks at 0.95 V, in addition to 1.10 V, in order to analyze the pros and cons of DDRNoC when its performance is traded-off for energy efficiency. We implemented a 4-stage (VA, SA, ST and LT) SDR Baseline with speculative SA, lookahead routing, 4 VCs per input port and credit based flow control [24]. Area and frequency results of these networks after physical implementation at 1.10 V and 0.95 V are presented in Table 7.2. We compared the performance of low voltage implementations of these networks with their high voltage (1.10 V) counterparts, considering 8×8 2D-mesh networks and synthetic traffic. Performance evaluated in terms of average packet latency is presented in Figure 7.12 for UR traffic pattern. We further report total power consumption and energy efficiency results measured in terms of energy per transferred bit (EPB), energy delay product (EDP) and energy throughput ratio (ETR). These results are presented in Figures 7.13 and 7.14 for UR traffic pattern.

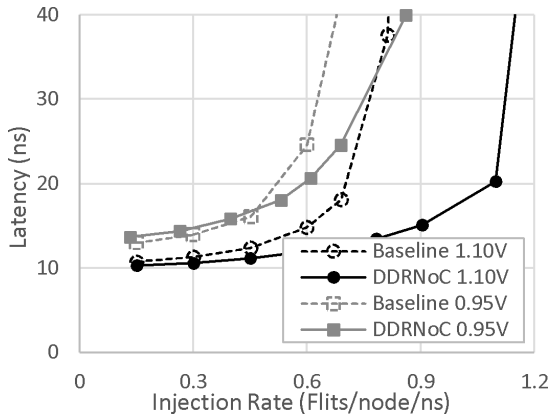
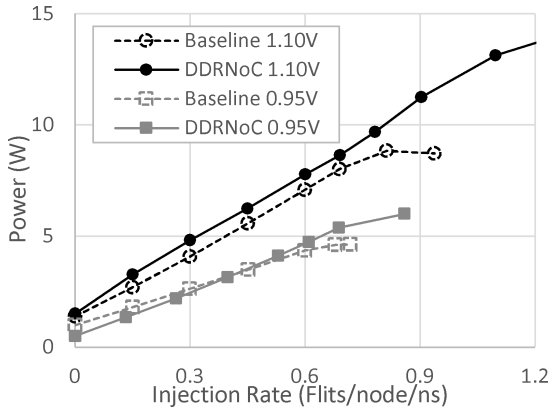


Figure 7.12: DDRNoC performance compared to the Baseline SDR NoC using 1.10 V and 0.95 V implementations of both networks. Performance is evaluated for UR traffic through 8×8 2D-mesh networks.

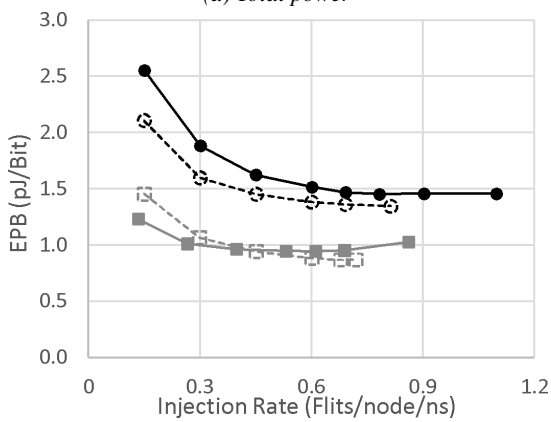
Compared to the SDR Baseline at 1.10 V, the DDRNoC at 1.10 V improves throughput by about 45% and offers up to 4% lower packet latency at low traffic injection rate. The power consumption of the DDRNoC at 1.10 V is up to 21% higher than the Baseline network at low traffic injection rate. At the Baseline saturation point, power consumption is up to 10% higher for the DDRNoC. At the maximum DDRNoC throughput, the 45% throughput gain costs about 51% more power than the Baseline. The DDRNoC has 10% to 21% higher EPB compared to the Baseline network. ETR is 17% higher (at low injection rate) to 26% lower (at high injection rate) for the DDRNoC. EDP is 16% lower at low traffic injection rate for the DDRNoC network.

For 0.95 V network implementations, DDRNoC improves throughput by about 30%, which represents a lower throughput gain compared to 1.10 V implementations due to a larger gap between operating frequencies of 0.95 V Baseline and DDRNoC routers. The DDRNoC at 0.95 V has up to 7% higher latency at low traffic injection rate, compared to the Baseline network at the same voltage. At low injection rates, the power

consumption of the DDRNoC is up to 45% lower than the Baseline network because of lower clock frequency of the DDRNoC. At the Baseline saturation throughput, power consumption is up to 16% higher for DDRNoC. At the maximum DDRNoC throughput, the 30% throughput gain costs about 31% more power than the Baseline. The DDRNoC has about 15% lower (at low injection rate) to 17% higher (at high injection rate) EPB compared to the Baseline network. ETR is about 7% higher (at low injection rate) to 8% lower (at high injection rate) for the DDRNoC. EDP is up to 11% lower at low traffic injection rate for the DDRNoC.



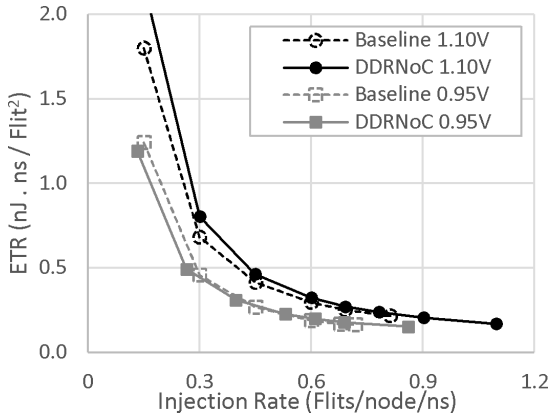
(a) Total power



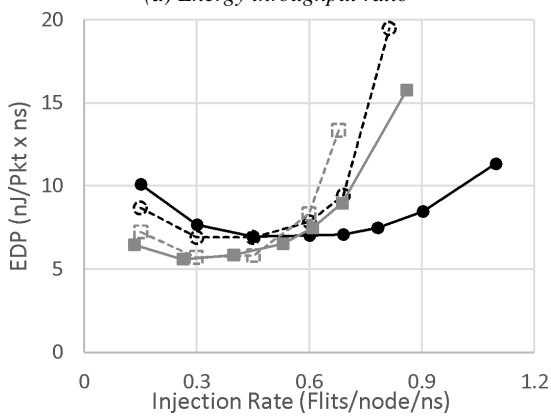
(b) Energy per transferred bit

Figure 7.13: Power consumption and energy per transferred bit results are presented for DDRNoC and Baseline SDR NoC using 1.10 V and 0.95 V implementations of both 2D-mesh networks of size 8×8. Networks are evaluated for UR traffic.

An interesting comparison is between the 1.10 V Baseline and the 0.95 V DDRNoC. This DDRNoC design point attempts to capitalize the slack present in the datapath gaining mostly energy efficiency rather than performance. Still, compared to Baseline network at 1.10 V, throughput of DDRNoC at 0.95 V is about 8% higher for UR traffic pattern, while packet latency is 27% higher due to its slower clock. On the other hand, power consumption of DDRNoC is 32% to 63% lower. This translates to 23% to 41% lower energy per transferred bit, 5% to 25% lower EDP, and 24% to 33% lower ETR



(a) Energy throughput ratio



(b) Energy-delay product

Figure 7.14: Energy-throughput ratio and energy-delay product results are presented for DDRNoC and Baseline SDR NoC using 1.10 V and 0.95 V implementations of both 2D-mesh networks of size 8×8. Networks are evaluated for UR traffic.

for the DDRNoC.

7.3.6 Evaluation with Application Driven Traffic

Using SynFull, we measure the average packet latency of 32-node (4×8) networks for various PARSEC and SPLASH-2 benchmarks. In order to stress networks throughput, in these experiments we considered 32-bit network datapaths, rather than 128-bits and SynFull packet generator step of 200ps (400ps for *fft* and *radix* benchmarks which saturate all networks). Average throughput does not provide any useful insight and therefore is not reported since for all networks the traffic generated by a benchmark is entirely delivered.

Figures 7.15 reports the average packet latency per benchmark as well as the geometric mean for each of the four DDR networks and the ShortPath network. Compared to ShortPath network, DDRNoC reduces average packet latency by about 4%. FreewayNoC and HighwayNoC networks offer 14% and 23% lower average

latency, respectively, compared to ShortPath. FastTrackNoC offers the lowest latency for all benchmarks. On average, FastTrackNoC reduces packet latency by about 26% compared to ShortPath due to its significant throughput advantage and competitive router latency. Compared to HighwayNoC, FastTrackNoC reduces packet latency by 3.7% due to its extended bypassing support. The average percentage of bypassing traffic in FastTrackNoC is similar to the HighwayNoC, however, 38% of this bypassing traffic bypasses both SA and ST stages using the FT-mode supported only by the FastTrackNoC. Moreover, FastTrackNoC offers about $\frac{3}{4}$ of the bypassing opportunities compared to ShortPath as it does not support bypassing for turning flits.

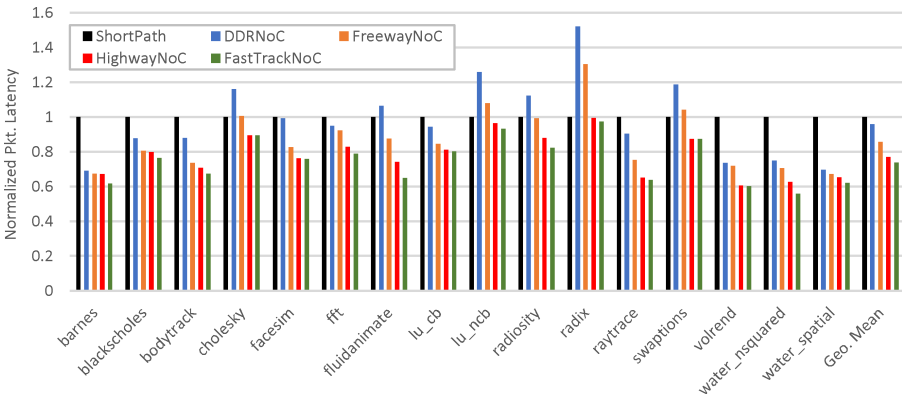


Figure 7.15: Normalized packet latency for application driven traffic.

7.3.7 Measuring the Impact of DDRNoC Throughput Gain on System Performance

In order to appreciate the impact of the improved DDRNoC network throughput to the application execution time and energy efficiency we finally performed full system simulations using GEM5 [59] system simulator, Ruby based memory subsystem models, and Baseline and DDRNoC 1.10 V models based on modified GARNET [60]. The detailed system configuration is presented in Table 7.3. We run medium-sized multi-threaded benchmarks from PARSEC-2.1 benchmark suite and measure the execution time of the parallel phase of the application called the Region of Interest (ROI).In order to estimate system power and EDP we used McPAT considering 28nm technology.

Figure 7.16 shows the normalized execution time with respect to the Baseline network of the ROI of 10 different PARSEC2.1 benchmarks. The largest network we could simulate within a reasonable time was a 4×4 2D-mesh. Despite the small network size we can observe that DDRNoC reduces application execution time by up to 11% and on average by 6% in the benchmarks used. As discussed in the introduction, the benefits of improving network throughput have been studied in more detail by Bakhoda et al. showing that doubling the network bandwidth improved execution time of applications up to 2× [61]. In addition, DDRNoC achieves lower EDP compared to the Baseline network by up to 20% and on average 10%, as shown in Figure 7.17.

Table 7.3: System and NoC configurations for Gem5 full system simulations.

System Parameters	
Cores	16 in-order Alpha ISA, 2.0GHz
Private L1 I/D cache	32 KB, 4 way set associative
Shared distributed L2 cache	8 MB, 8 way set associative, 16 directories
Cache line size	64 Bytes
Replacement policy	Pseudo-LRU
Cache coherency	MESI protocol
Topology	4x4 2D-mesh
Memory size	1 GB
Network Parameters	
Virtual networks and VCs	3 VNs, 2 VCs/VN 1 buffer per ctrl VC 4 buffers per data VC
Packet size	Ctrl: 2 flits Data: 18 flits
Links	32 bits, 1 cycle latency

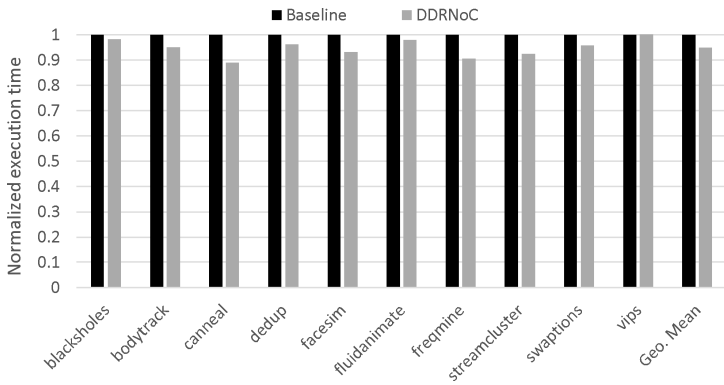


Figure 7.16: Normalized application execution time for PARSEC-2.1 benchmarks using full system simulations.

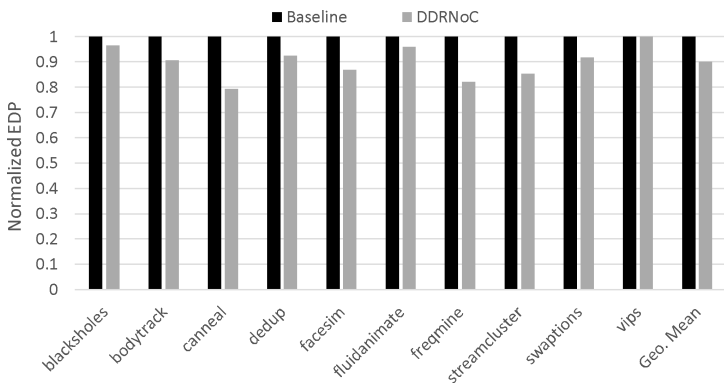


Figure 7.17: Normalized processor EDP for PARSEC-2.1 benchmarks using full system simulations.

7.4 Summary

This Chapter presents the physical implementation, performance evaluation, power and energy efficiency results of the DDR NoC architectures and compares them to the ShortPath network. The DDR NoCs routers require 6-13% more gates compared to the ShortPath network, because of extra bits required in the datapath for forwarded control signals, more complex allocation logic and inter-router bypass paths for flits and forwarded control signals. Critical path analysis of a DDR NoC router demonstrates that its clock frequency is defined by the datapath (crossbar) delay. As a result, our DDR NoCs offer 16-20% higher throughput compared to the ShortPath network. Moreover, DDR NoCs offer 50% higher to 40% lower average packet latency, compared to the ShortPath network, depending on the bypassing opportunities (SA and ST bypassing) the DDR NoC offers and the network size.

Chapter 8

Conclusions

Shrinking transistor geometries still allow integration of more cores on a chip increasing its performance potential. However, the performance of a single chip system is often affected by the performance of its on-chip interconnects [7, 13, 61]. Depending on the workload characteristics and communication needs, network latency and throughput can both be crucial for high performance. Moreover, some workloads may be more sensitive to packet latency, e.g. workloads that exhibit small transfers at low loads [4]. Others may be more sensitive to network throughput, for instance concurrent scale-out applications, which push the network close to its saturation point [4, 5]. As a consequence, the design of high-performance NoC architectures is essential for many-core scaling.

8.1 Summary

Chapter 3 introduced the **DDRNoC** router architecture which uses double-pumped datapaths in order to improve network throughput. It is based on the observation that conventional SDR 2D-mesh interconnects with VC flow control have significant slack in their datapath stages while their control logic defines the critical path. This slack in the datapath wastes NoC bandwidth. DDRNoC uses this slack to enable two flits to share the same datapath in a single clock cycle at DDR. Both the allocation logic and the datapath are modified to facilitate conflict free DDR transmission of flits. The DDRNoC offers significant throughput improvement over a SDR network by routing packets at a rate determined only by datapath stages ($2 \times max(ST, LT)$), rather than the control. Packet latency is improved at high injection rates due to reduced contention.

Chapter 4 introduced the **FreewayNoC** router architecture which improves over the DDRNoC by implementing simplified pipeline stage bypassing in order to reduce packet latency at low injection rates. It allows incoming flits to bypass the SA stage when required router resources are available. Offering full pipeline bypassing, from all inputs to all outputs, requires complex checks and it could introduce control logic in the datapath which defines the clock period. This goes against one of the key design goals. Therefore, FreewayNoC supports pipeline bypassing only for flits propagating an in-network straight hop.

Chapter 5 introduced the **HighwayNoC** router architecture which enhances pipeline bypassing support offered by the FreewayNoC. It not only allows bypassing of the SA stage for flits propagating a straight hop, similar to the FreewayNoC, but

also for flits entering and exiting the network through the local port. HighwayNoC is able to improve allocation bypassing, while manage contention among simultaneous incoming flits attempting to bypass at the same time, without adding any delay to the critical path of the DDRNoC router. Allocation bypassing candidate flits propagating an in-network straight hop can only face competition from simultaneous incoming flits entering the local input port. In this case priority is always given to in-network flits, avoiding the need for any arbitration. Moreover, HighwayNoC also exploits the fact that the local port of a router has a shorter (local) link traversal and uses the available slack for fixed priority arbitration to choose a bypassing flit among the candidates that exit the network. Furthermore, bypassing logic for flits that enter the network also fits without increasing the cycle time.

Chapter 6 introduced the **FastTrackNoC** router architecture which enables straight flits to bypass the ST stage as well as the SA stage similar to the HighwayNoC. FastTrackNoC capitalizes on the observation that with xy-routing a flit performs at most one in-network turn and offers a direct connection between each input port and its straight output thereby bypassing the ST stage and reducing hop latency when required conditions are met. In particular, an extra FastTrack datapath is added to a 2D-mesh router between the head of a specific Virtual Channel (VC) at each input port of the router and its preferred, straight output port. Then, incoming non-turning flits arriving at this specific VC have the opportunity to directly traverse the link using the FastTrack path when input and output ports are available.

8.2 Contributions

Chapter 3 introduced the DDRNoC architecture. It is based on the observation that clock frequency of typical SDR NoC routers is based on its control logic delays, while the slack present in the datapath wastes network bandwidth. In order to improve NoC throughput, DDRNoC operates it datapath at DDR and has a clock frequency defined only by datapath delays. The main contributions of the DDRNoC architecture are listed below:

- DDRNoC offers 17-20% higher throughput compared to one of the fastest SDR NoC (ShortPath).
- Compared to the ShortPath network, DDRNoC has up to 50% higher average packet latency, at low injection rates with synthetic traffic. This is mainly because of slower clock frequency of DDRNoC and lack of pipeline stage bypassing capabilities.
- DDRNoC reduces energy per transferred bit, energy delay product and energy throughput ratio due to a slower clock and higher network throughput.
- A low-voltage DDRNoC implementation reduces power consumption by up to 40%, at the cost of 26-40% higher latency, still however offering similar or better throughput when compared to a high-voltage implementation of a conventional SDR NoC. This low voltage implementation further improves energy efficiency, offering a substantially better energy-performance trade-off.

Chapter 4 introduced the FreewayNoC architecture, which allows incoming flits to bypass the allocation stage in order to reduce packet latency. FreewayNoC reduces

complexity of bypassing check logic by only allowing flits propagating an in-network straight hop to bypass allocation. Main contributions of the FreewayNoC architecture are listed below:

- FreewayNoC offers a minimum hop latency equal to the datapath delay ($ST + LT$), for all in-network non-turning hops. This is accomplished by allowing incoming flits to bypass the allocation stage in the absence of contention.
- FreewayNoC matches or improves the throughput offered by DDRNoC architecture and offers 17-20% higher throughput compared to ShortPath.
- FreewayNoC offers up to 25% lower packet latency compared to the DDRNoC for synthetic traffic patterns.
- FreewayNoC achieves a zero-load latency that scales to the number of hops better than DDRNoC and ShortPath networks, and equally well as a network that offers minimum $ST + LT$ hop latency. This is because, for a balanced router datapath (ST = LT), per hop packet latency with FreewayNoC is at best equal to the sum of the ST and LT delays.

Chapter 5 introduced the HighwayNoC architecture, which enhances the allocation bypassing capabilities of the FreewayNoC. It allows incoming flits to bypass allocation when entering and exiting the network through the local port, as well as when propagating an in-network straight hop. Main contributions of the HighwayNoC architecture are listed below:

- HighwayNoC offers packet latency as low as $ST + LT$ datapath delay for all hops except the in-network turns, including entry into and exit from the network through the local port.
- Throughput of HighwayNoC is similar to DDRNoC architecture and 17-20% higher than ShortPath network.
- HighwayNoC offers up to 3% higher packet latency compared to ShortPath for small network sizes (8×8), which it can be up to 10% lower for larger networks (32×32).
- HighwayNoC offers 16-33% lower packet latency compared to FreewayNoC for small network sizes (8×8) and 7-32% lower for larger networks (32×32).
- Zero-load latency of FastTrackNoC is lower than FreewayNoC and scales to the number of hops equally well as a network that offers minimum $ST + LT$ hop latency.
- HighwayNoC reduces the number of control wires per link by using a more efficient encoding and by restricting the number of bypassing header flits to one (rather than two) per port, per cycle.

Chapter 6 introduced the fourth and the fastest DDR NoC architecture, the FastTrackNoC, which offers intra-router FastTrack paths to allow incoming flits, propagating an in-network straight hop, to bypass ST and SA pipeline stages, when required conditions are met, to directly initiate LT and reduce packet latency. This is in addition to bypassing only the SA stage when incoming flits propagate straight or enter and exit the network. Main contributions of the FastTrackNoC architecture are listed below:

- FastTrackNoC is able to achieve a hop latency at best similar to the link traversal delay. It allows flits to propagate a hop in half a clock cycle when traveling straight, compared to one cycle for the HighwayNoC and the FreewayNoC and two cycles for the SDR networks.
- Throughput of FastTrackNoC is similar to DDRNoC and 17-19% higher compared to ShortPath.
- FastTrackNoC reduces packet latency by up to 32% and 40% (for 32×32 2D-mesh) compared to HighwayNoC and existing SDR networks, respectively, at low traffic injection rates.
- Zero-load latency of FastTrackNoC scales to the number of hops much better than HighwayNoC and ShortPath networks.

In summary, this thesis proposes DDR NoC architectures which reduce, compared to previous NoCs, the performance gap with ideal network in terms of throughput and latency, as shown in Figure 8.1. Compared to the fastest previous NoCs in the literature, ShortPath, the overall contributions of the DDR NoC architectures are listed below:

- They offer up to 20% higher throughput.
- They reduce average packet latency by up to 40%.
- Their packet latency is more scalable to the number of hops.

8.3 Future Work

There are several directions for future research which can improve and complement the work presented here. In the following, we identify and list some of them:

Interconnects in 3D Stacked Chips: Presently, the semiconductor industry is targeting interposer based 2.5D [62–64] and 3D [65–67] stacked chips in order to address scaling challenges of 2D chips. Such 2.5D and 3D chips utilize vertical wires called through silicon vias (TSVs) to establish communication between different layers. TSVs are slower and consume more area and power compared to 2D metal wires and pose a challenge for 3D NoC designs [68]. It would be interesting to analyze the contribution of delays of different components (control logic, ST, 2D wires and TSVs) of current 3D NoC routers to the clock period in order to figure out pipeline delay imbalances and inefficiencies in these designs. It is important to note that a NoC operates at maximal efficiency when pipeline stages of its routers are well balanced and its clock period is defined by the datapath delays. If some parts of 3D NoC router's datapath are faster than others, then they can be operated at DDR to improve network performance.

Support For Different Topologies: This thesis only evaluated 2D-mesh topologies of DDR NoC. However, other 2D NoC topologies also exist which offer benefits in terms of higher path diversity, lower packet latency and increased throughput. These topologies include 2D-torus, flattened butterfly, multidrop express channels [22] and express links [4] spanning multiple hops. The critical path for these topologies is also in the router control logic. Operating the datapath of these routers at DDR can offer significant throughput and latency improvements.

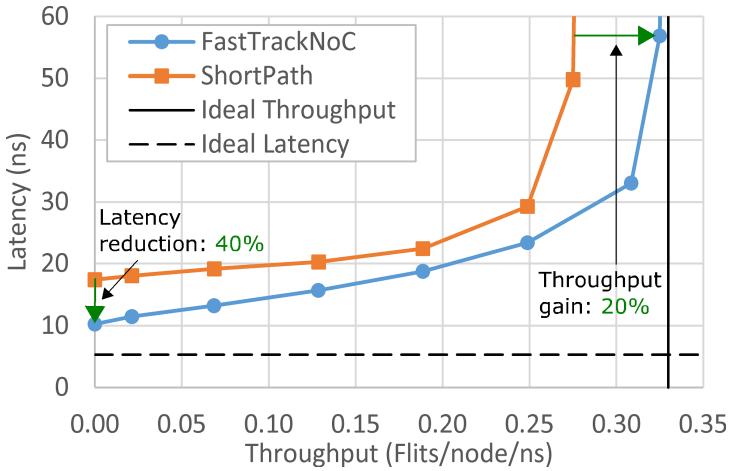


Figure 8.1: Throughput and latency of one of the fastest SDR networks, called the ShortPath network [30], and the final proposed DDR network which supports SA and ST bypassing, called the FastTrackNoC, is presented in comparison to the ideal network throughput and latency. Network size of 32×32 is considered with uniform random traffic of 1 and 5 flit packets. Ideal throughput is the saturation throughput of the ShortPath network when operating at a clock frequency defined solely by the datapath (ST and LT) delays. Ideal latency is for a (pipelined) direct connection between the source and the destination nodes, separated by a hop distance equal to the average hop count with uniform random traffic in a 32×32 2D-mesh network. The link for this direct connection is pipelined considering tile dimensions similar to ShortPath and it considers a packet containing 3 flits. All networks are operated at their maximum clock frequency.

Allocation Bypassing for In-Network Turns The DDR NoC routers do not support bypassing (SA or ST) for flits performing an in-network turn. Bypassing ST stage when turning is not possible because that would require more FastTrack bypass paths and slow down network clock frequency. However, it may be possible to implement allocation bypassing for incoming turning flits. This is because DDR NoC routers already have the forwarded control and flit datapaths which would be needed to forward data and control if turning flits are allowed to bypass allocation. However, satisfying tight timing constraints for these paths, along with more complex bypassing check logic for turns, is challenging. It would be interesting to explore if bypassing checks for turns can somehow be simplified to support allocation bypassing when turning while also satisfying the required timing constraints of DDR NoCs.

Bibliography

[1] H. Fu, J. Liao, J. Yang, L. Wang, Z. Song, X. Huang, C. Yang, W. Xue, F. Liu, F. Qiao, and et al., "The Sunway TaihuLight supercomputer: system and applications," *Science China Information Sciences*, vol. 59, no. 7, 2016.

[2] B. Bohnenstiehl, A. Stillmaker, J. Pimentel, T. Andreas, B. Liu, A. Tran, E. Adeagbo, and B. Baas, "KiloCore: A fine-grained 1,000-processor array for task-parallel applications," *IEEE Micro*, vol. 37, no. 2, pp. 63–69, 2017.

[3] A. Olofsson, "Epiphany-V: A 1024 processor 64-bit RISC system-on-chip," 2016.

[4] A. Psathakis, V. Papaefstathiou, N. Chrysos, F. Chaix, E. Vasilakis, D. Pnevmatikatos, and M. Katevenis, "A systematic evaluation of emerging mesh-like CMP NoCs," in *ANCS*, 2015, pp. 159–170.

[5] P. Lotfi-Kamran, B. Grot, and B. Falsafi, "NOC-Out: Microarchitecting a scale-out processor," in *2012 45th Annual IEEE/ACM Int. Symp. on Microarchitecture*, Dec 2012, pp. 177–187.

[6] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," in *ISCA*, 2011, pp. 365–376.

[7] S. Borkar, "How to stop interconnects from hindering the future of computing!" in *2013 Optical Interconnects Conf.*, May 2013, pp. 96–97.

[8] M. B. Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffman, P. Johnson, J.-W. Lee, W. Lee *et al.*, "The raw microprocessor: A computational fabric for software circuits and general-purpose programs," *IEEE micro*, vol. 22, no. 2, pp. 25–35, 2002.

[9] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar, "A 5-GHz mesh interconnect for a teraflops processor," *IEEE Micro*, vol. 27, no. 5, pp. 51–61, 2007.

[10] S. Bell, B. Edwards, J. Amann, R. Conlin, K. Joyce, V. Leung, J. MacKay, M. Reif, L. Bao, J. Brown *et al.*, "Tile64-processor: A 64-core SoC with mesh interconnect," in *IEEE Int. Solid-State Circuits Conf.*, 2008, pp. 88–598.

[11] P. Salihundam, S. Jain, T. Jacob, S. Kumar, V. Erraguntla, Y. Hoskote, S. Vangal, G. Ruhl, and N. Borkar, "A 2 Tb/s 6 x 4 mesh network for a single-chip cloud computer with DVFS in 45 nm CMOS," *IEEE J. of Solid-State Circuits*, vol. 46, no. 4, pp. 757–766, April 2011.

[12] J. Howard, S. Dighe, S. R. Vangal, G. Ruhl, N. Borkar, S. Jain, V. Erraguntla, M. Konow, M. Riepen, M. Gries, G. Droege, T. Lund-Larsen, S. Steibl, S. Borkar, V. K. De, and R. V. D. Wijngaart, "A 48-core IA-32 processor in 45 nm CMOS using on-die message-passing and DVFS for performance and power scaling," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 1, pp. 173–183, Jan 2011.

[13] B. K. Daya, C.-H. O. Chen, S. Subramanian, W.-C. Kwon, S. Park, T. Krishna, J. Holt, A. P. Chandrakasan, and L.-S. Peh, "SCORPIO: A 36-core research chip demonstrating snoopy coherence on a scalable mesh NoC with in-network ordering," in *Int. Symp. on Computer Architecuture*, ser. ISCA '14, 2014, pp. 25–36.

[14] B. Bohnenstiehl, A. Stillmaker, J. J. Pimentel, T. Andreas, B. Liu, A. T. Tran, E. Adeagbo, and B. M. Baas, "KiloCore: A 32-nm 1000-processor computational array," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 4, pp. 891–902, April 2017.

[15] J. Kim, J. Balfour, and W. Dally, "Flattened butterfly topology for on-chip networks," in *MICRO-40*, 2007, pp. 172–182.

[16] J. Balfour and W. J. Dally, "Design tradeoffs for tiled CMP on-chip networks," in *Int. Conf. on Supercomputing*, 2006, pp. 187–198.

[17] A. Psarras, S. Moisidis, C. Nicopoulos, and G. Dimitrakopoulos, "Networks-on-chip with double-data-rate links," *IEEE Trans. on Circuits and Systems*, vol. 64, no. 12, pp. 3103–3114, 2017.

[18] S. Rao, S. Jeloka, R. Das, D. Blaauw, R. Dreslinski, and T. Mudge, "VIX: Virtual input crossbar for efficient switch allocation," in *Design Automation Conf. (DAC)*, 2014, pp. 103:1–103:6.

[19] H. Kim, A. Vitkovskiy, P. V. Gratz, and V. Soteriou, "Use it or lose it: Wear-out and lifetime in future chip multiprocessors," in *MICRO-46*, 2013, pp. 136–147.

[20] A. K. Mishra, R. Das, S. Eachempati, R. Iyer, N. Vijaykrishnan, and C. R. Das, "A case for dynamic frequency tuning in on-chip networks," in *MICRO-42*, 2009, pp. 292–303.

[21] C. Nicopoulos, V. Narayanan, and C. R. Das, *Network-on-Chip Architectures - A Holistic Design Exploration*, ser. Lecture Notes in Elect. Eng. Springer, 2010, vol. 45.

[22] B. Grot, J. Hestness, S. W. Keckler, and O. Mutlu, "Express cube topologies for on-chip interconnects," in *HPCA*, Feb 2009, pp. 163–174.

[23] J. Kim, D. Park, T. Theocharides, N. Vijaykrishnan, and C. R. Das, "A low latency router supporting adaptivity for on-chip interconnects," in *Design Automation Conf. (DAC)*, 2005, pp. 559–564.

[24] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers, 2004.

[25] L.-S. Peh and W. J. Dally, "A delay model and speculative architecture for pipelined routers," in *Int. Symp. on HPCA*, 2001.

[26] C. O. Chen, S. Park, T. Krishna, S. Subramanian, A. P. Chandrakasan, and L. Peh, "SMART: A single-cycle reconfigurable NoC for SoC applications," in *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2013, pp. 338–343.

[27] A. Kumar, L.-S. Peh, P. Kundu, and N. K. Jha, "Express virtual channels: Towards the ideal interconnection fabric," *SIGARCH Comput. Archit. News*, vol. 35, no. 2, 2007.

[28] F. Alazemi, A. Azizimazreah, B. Bose, and L. Chen, "Routerless network-on-chip," in *IEEE Int'l Symp. on High Performance Computer Architecture (HPCA)*, 2018, pp. 492–503.

[29] T. Krishna, J. Postman, C. Edmonds, L.-S. Peh, and P. Chiang, "SWIFT: A swing-reduced interconnect for a token-based network-on-chip in 90nm CMOS," in *2010 IEEE International Conference on Computer Design*, 2010, pp. 439–446.

[30] C. N. Anastasios Psarras, Ioannis Seitanidis and G. Dimitrakopoulos., "ShortPath: A network-on-chip router with fine-grained pipeline bypassing," *IEEE Trans. on Computers*, vol. 65, no. 10, pp. 3136–3147, 2016.

[31] J. Xu, W. Wolf, and W. Zhang, "Double-data-rate, wave-pipelined interconnect for asynchronous NoCs," *IEEE Micro*, vol. 29, no. 3, pp. 20–30, May 2009.

[32] A. Psarras, S. Moisidis, C. Nicopoulos, and G. Dimitrakopoulos, "RapidLink: A network-on-chip architecture with double-data-rate links," in *2016 IEEE International Conference on Electronics, Circuits and Systems, ICECS*, 2016, pp. 93–96.

[33] A. Kumary, P. Kunduz, A. P. Singhx, L.-S. Pehy, and N. K. Jhay, "A 4.6Tbits/s 3.6GHz single-cycle NoC router with a novel switch allocator in 65nm CMOS," in *2007 25th International Conference on Computer Design*, 2007, pp. 63–70.

[34] S. Park, T. Krishna, C. H. Chen, B. Daya, A. Chandrakasan, and L. S. Peh, "Approaching the theoretical limits of a mesh NoC with a 16-node chip prototype in 45nm SOI," in *DAC Design Automation Conference 2012*, June 2012, pp. 398–405.

[35] M. Li, Q.-A. Zeng, and W.-B. Jone, "DyXY - a proximity congestion-aware deadlock-free dynamic routing method for network on chip," in *2006 43rd ACM/IEEE Design Automation Conference*, 2006, pp. 849–852.

[36] C. Wang, W.-H. Hu, and N. Bagherzadeh, "Scalable load balancing congestion-aware network-on-chip router architecture," *Journal of Computer and System Sciences*, vol. 79, no. 4, pp. 421–439, 2013, jCSS CADS 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0022000012001432

[37] G. P. Nychis, C. Fallin, T. Moscibroda, O. Mutlu, and S. Seshan, "On-chip networks from a networking perspective: Congestion and scalability in many-core interconnects," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, p. 407–418, Aug. 2012. [Online]. Available: https://doi.org/10.1145/2377677.2377757

[38] C. Wang, W.-H. Hu, and N. Bagherzadeh, "Congestion-aware network-on-chip router architecture," in *2010 15th CSI International Symposium on Computer Architecture and Digital Systems*, 2010, pp. 137–144.

[39] D. U. Becker and W. J. Dally, "Allocator implementations for network-on-chip routers," in *Conf. on HPC Net., Stor. and Analysis*, ser. SC '09, 2009, pp. 52:1–52:12.

[40] S. Liu, A. Jantsch, and Z. Lu, "A fair and maximal allocator for single-cycle on-chip homogeneous resource allocation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 10, pp. 2230–2234, 2014.

[41] Y. Lu, C. Chen, J. McCanny, and S. Sezer, "Design of interlock-free combined allocators for networks-on-chip," in *2012 IEEE International SOC Conference*, 2012.

[42] M. Galles, "Spider: A high-speed network interconnect," *IEEE Micro*, vol. 17, no. 1, pp. 34–39, Jan. 1997.

[43] Y. J. Yoon, N. Concer, M. Petracca, and L. P. Carloni, "Virtual channels and multiple physical networks: Two alternatives to improve NoC performance," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2013.

[44] J. Kim, W. J. Dally, and D. Abts, "Flattened butterfly: A cost-efficient topology for high-radix networks," *SIGARCH Comput. Archit. News*, vol. 35, no. 2, p. 126–137, Jun. 2007. [Online]. Available: https://doi.org/10.1145/1273440.1250679

[45] C. H. O. Chen, N. Agarwal, T. Krishna, K. H. Koo, L. S. Peh, and K. C. Saraswat, "Physical vs. virtual express topologies with low-swing links for future many-core nocs," in *NOCS*, 2010, pp. 173–180.

[46] T. Moscibroda and O. Mutlu, "A case for bufferless routing in on-chip networks," in *Proceedings of the 36th Annual International Symposium on Computer Architecture*, ser. ISCA '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 196–207. [Online]. Available: https://doi.org/10.1145/1555754.1555781

[47] H. Kim, C. Kim, M. Kim, K. Won, and J. Kim, "Extending bufferless on-chip networks to high-throughput workloads," in *2014 Eighth IEEE/ACM International Symposium on Networks-on-Chip (NoCS)*, 2014, pp. 9–16.

[48] C. Feng, Z. Liao, Z. Lu, A. Jantsch, and Z. Zhao, "Performance analysis of on-chip bufferless router with multi-ejection ports," in *2015 IEEE 11th International Conference on ASIC (ASICON)*, 2015, pp. 1–4.

[49] G. Michelogiannakis, D. Sanchez, W. J. Dally, and C. Kozyrakis, "Evaluating bufferless flow control for on-chip networks," in *2010 Fourth ACM/IEEE International Symposium on Networks-on-Chip*, 2010, pp. 9–16.

[50] G. Michelogiannakis, D. Pnevmatikatos, and M. Katevenis, "Approaching ideal NoC latency with pre-configured routes," in *Int'l Symp. on NOCS*, May 2007, pp. 153–162.

[51] M. Hayenga and M. Lipasti, "The NoX router," in *MICRO-44*, 2011.

[52] M. Azimi, D. Dai, A. Kumar, A. Mejia, D. Park, R. Saharoy, and A. S. Vaidya, "Flexible and adaptive on-chip interconnect for tera-scale architectures," *Intel Technology Journal*, vol. 13, no. 4, pp. 62–77, 2009.

[53] L. Chen, D. Zhu, M. Pedram, and T. M. Pinkston, "Power punch: Towards non-blocking power-gating of NoC routers," in *HPCA*, 2015.

[54] P. Lotfi-Kamran, M. Modarressi, and H. Sarbazi-Azad, "Near-ideal networks-on-chip for servers," in *HPCA*, 2017.

[55] K. Sewell, R. G. Dreslinski, T. Manville, S. Satpathy, N. R. Pinckney, G. Blake, M. Cieslak, R. Das, T. F. Wenisch, D. Sylvester, D. Blaauw, and T. N. Mudge, "Swizzle-switch networks for many-core systems." *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 2, no. 2, pp. 278–294, 2012.

[56] M. Badr and N. E. Jerger, "SynFull: Synthetic traffic models capturing cache coherent behaviour," in *ACM/IEEE ISCA*, June 2014, pp. 109–120.

[57] C. Bienia, "Benchmarking modern multiprocessors," Ph.D. dissertation, Department of Computer Science, Princeton, NJ, USA, 2011.

[58] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The SPLASH-2 programs: Characterization and methodological considerations," *SIGARCH Comp. Archit. News*, vol. 23, no. 2, pp. 24–36, 1995.

[59] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The Gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011.

[60] N. Agarwal, T. Krishna, L. S. Peh, and N. K. Jha, "GARNET: A detailed on-chip network model inside a full-system simulator," in *IEEE Int. Symp. on Performance Analysis of Systems and Software*, April 2009, pp. 33–42.

[61] A. Bakhoda, J. Kim, and T. M. Aamodt, "Throughput-effective on-chip networks for manycore accelerators," in *MICRO-43*, 2010, pp. 421–432.

[62] I. Akgun, J. Zhan, Y. Wang, and Y. Xie, "Scalable memory fabric for silicon interposer-based multi-core systems," in *2016 IEEE 34th International Conference on Computer Design (ICCD)*, 2016, pp. 33–40.

[63] A. Kannan, N. E. Jerger, and G. H. Loh, "Exploiting interposer technologies to disintegrate and reintegrate multicore processors," *IEEE Micro*, vol. 36, no. 3, pp. 84–93, 2016.

[64] D. Stow, Y. Xie, T. Siddiqua, and G. H. Loh, "Cost-effective design of scalable high-performance systems using active and passive interposers," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2017, pp. 728–735.

[65] D. Stow, I. Akgun, W. Huangfu, Y. Xie, X. Li, and G. H. Loh, "Invited: Efficient system architecture in the era of monolithic 3D: Dynamic inter-tier interconnect and processing-in-memory," in *2019 56th ACM/IEEE Design Automation Conference (DAC)*, 2019, pp. 1–4.

[66] G. H. Loh and Y. Xie, "3D stacked microprocessor: Are we there yet?" *IEEE Micro*, vol. 30, no. 3, pp. 60–64, 2010.

[67] G. H. Loh, "3D-stacked memory architectures for multi-core processors," in *2008 International Symposium on Computer Architecture*, 2008, pp. 453–464.

[68] H. Matsutani, "Research challenges on 2-D and 3-D network-on-chips," in *2013 First International Symposium on Computing and Networking*, 2013, pp. 24–25.