The Runahead Network-On-Chip

by

Zimo Li

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science
Graduate Department of Electrical and Computer Engineering
University of Toronto

Abstract

The Runahead Network-On-Chip

Zimo Li
Master of Applied Science
Graduate Department of Electrical and Computer Engineering
University of Toronto
2015

With more cores per chip multiprocessor and higher memory demands from applications, it is imperative that networks-on-chip (NoC) provides low-latency, power-efficient communication. Conventional NoCs tend to be over-provisioned for worst-case bandwidth demands. This results in ineffective use of network resources and power inefficiency. In terms of performance, low-latency techniques often introduce overheads and incur complexity in the router microarchitecture. We find that both low latency and power efficiency is possible by relaxing the constraint of lossless communication.

We propose the *Runahead NoC*, a lightweight, lossy network that provides single-cycle hops. Allowing for lossy delivery makes it possible to design an extremely simple bufferless router microarchitecture. The *Runahead NoC* operates either as a *power-saver* to improve power-efficiency, or as an *accelerator* to provide ultra-low latency communication for selected packets. The *Runahead NoC* reduces network power consumption by 1.80x as a *power-saver* and improves latency by 1.66x as an *accelerator*.

Acknowledgements

The two years of my M.A.Sc study here at the University of Toronto was full of challenges and rewards. First and foremost, I would like to thank my thesis advisor, Natalie Enright Jerger, for her great support and guidance throughout my graduate study. Natalie not only accepted me when I was clueless due to the sudden loss of my first supervisor, she also guided me back on track. Congratulations to Natalie on her upcoming newest member of her family! I would also like to thank Gregory Steffan, my first supervisor, for his support and guidance in the beginning of my study. His sudden passing was devastating to me, but I think I came out of it as a stronger person.

I would also like to thank the members of my thesis committee: Ravi Adve, Jason Anderson and Andreas Moshovos for their technical insights and suggestions in regard to this thesis.

I sincerely thank the excellent research members in our group: Ajay, Jorge, Mark, Mario, Parisa, Robert, Shehab, and Wenbo for their brilliant minds. Thank you for all the moments we have shared in the office and for the hours of struggle that you have saved me. My Master's experience would not have been the same without you. Special thanks to Joshua who inspired the basis of this thesis during a lunch at a sushi bar.

Finally, the completion of my thesis would not have been possible without the support of my family and friends. Their understanding and unconditional love have given me the confidence to rise against to many challenges and to overcome them. A special thanks to Rain for her support and love throughout my study, and for being there in both the good and the bad days.

Contents

Chapter 1

Introduction

Chip multi-processors (CMPs) have drawn significant research interest nowadays as they show promising potential in terms of performance in comparison to uni-processors, as the performance of uni-processors is now limited by power and heat constraints. However, as number of cores integrated on a single chip increases, conventional communication mechanisms are not efficiently meeting the increased bandwidth demand. Network-on-chip (NoCs) have become an emerging research area as NoCs have been shown to have the potential to be an effective and efficient way of providing communicating fabric in CMP systems.

NoC is an essential part of system design, especially in situations where the number of cores on the chip is expected to grow in the future. With thousand-core systems anticipated in the future, higher communication demand and traffic will put more pressure on NoCs and consume more power. However, energy efficiency is already a huge concern for researcher and designers [10, 11] as NoCs consume a significant amount of power in modern chip multiprocessors [32,55]. Thus, NoCs designers must consider router micro-architecture in order to produce a scalable design as better architecture can improve both performance and energy efficiency.

1.1 NoC Design Challenges

Minimizing power consumption requires more efficient use of network resources. Though buffers consume a significant portion of network power and area [32], traditional NoC designs tend to provision large amounts of buffers to meet worst-case throughput requirements. Yet large buffers are often unnecessary as single-flit packets represent a high fraction of the total network traffic in real applications [41]. Several bufferless NoC designs have been proposed in the past [21, 28, 43, 45]. These designs achieve significant power savings at a cost of lower saturation throughput compared to conventional buffered routers. Previous analysis [5,26,31] of single-threaded and multi-threaded CMP workloads finds that NoC channel utilization tends to be low, with average injection rates of only 5%. Low resource utilizations translate to inefficient use of network resources. To address this, several multi-NoC systems have been proposed in the past [1, 18, 20, 23, 51, 53, 57]. Multi-NoCs allow for more efficient utilization of total bandwidth since they can be designed with heterogeneous physical subnetworks; messages can be categorized and injected into different networks depending on packet type. For example, latency sensitive messages are injected into a low-latency, high-power network, while non-critical messages are injected into a low-power network [1, 53].

To better meet the higher communication demands of future CMPs, significant research has been done to minimize the NoC latency caused by growing network diameters. For example, lookahead routing [24] and speculative switch allocator [50] are common techniques to reduce network latency by reducing router pipeline stages. Non-speculative single-cycle routers can be designed by allocating router switches in advance of packet arrival [38]. NoC latency can also be reduced by employing route predictions [30, 42] or bypassing intermediate routers via express virtual channels [39]. Though these designs improve performance, they come at a cost of increased complexity, power and area.

1.2 Overview of Runahead Network-On-Chip

We propose the *Runahead NoC*, which serves as

- A *power-saver* that exploits heterogeneous traffic for more efficient use of network resources
- An *accelerator* that provides lightweight, low-latency communication on top of a conventional NoC.

The Runahead NoC is designed for simplicity; it is bufferless with a lightweight router architecture (i.e., simplified routing and port arbitration) that is designed to enable single-cycle hops across the network. To accomplish this simplicity, the Runahead NoC is lossy, allowing packets to be dropped in the presence of contention. It is inspired by the “best effort” concept in internetworking, meaning that there is no guarantee a packet will arrive at its destination.[1] Our design is not meant to be a stand-alone network; it is meant as a plug-and-play network that either replaces resources in an existing NoC to save power or is added on top as an accelerator. The Runahead network is bufferless and very lightweight, consuming very little area and power. As a *power-saver*, the Runahead NoC utilizes existing network resources and is meant to transfer latency-sensitive single-flit packets. The remaining resources act as a conventional, lossless network, delivering all packets (including the latency-sensitive ones that are injected into the Runahead network) to provide guaranteed delivery. As an *accelerator*, the Runahead NoC is added as an additional subnetwork to a conventional NoC to allow faster transfer of latency-sensitive single-flit packets. By using a separate physical subnetwork, the entire Runahead network can be simply turned on and off depending on its usefulness to the application, unlike prior low-power techniques where power gating specific router components is complex.

[1] In contrast, ”best effort” in NoC literature usually means that there is no guarantee on bandwidth and latency. Best effort NoCs have been explored in the context of quality of service.

1.3 Research Highlights

This thesis makes the following contributions:

- We propose the novel Runahead NoC that is designed for simplicity, providing single-cycle hops and "best effort" delivery for latency-sensitive packets.
- We evaluate the Runahead NoC as a *power-saver* and show that it achieves 1.80× and 1.73× savings in network power and active silicon area, while still providing 1.33× lower latency.
- We evaluate the Runahead NoC as an *accelerator* and show that it improves packet latency by 1.66× on average, with only 10% and 16% overheads in network power and active silicon area.
- We showed that in full-system simulation, the Runahead network can accelerate multi-threaded applications by 1.04× in power-saver mode, and by 1.10× on average in accelerator mode.

1.4 Thesis Organization

The rest of the thesis is organized as follows:

- Chapter 2 presents relevant background on NoCs, and motivations for the Runahead network. We first introduce the basics of flow control and router micro-architecture. We also analyze the source of network latency and major contributors of NoC power consumption. We then discuss the motivations behind the Runahead network.
- Chapter 3 presents the design of the Runahead router and how it can be integrated into a conventional baseline network. It also addresses details like fairness and starvation.
- Chapter 4 presents the methodology and simulation setups used to evaluate the proposed network. We used 2 different simulation setups: synthetic traffic and full-system. We used Booksim and SynFull for synthetic traffic to evaluate Runahead performance in terms of arrival rate, throughput, and packet latency. We switch to Gem5 and Booksim for full-system simulation to evaluate the Runahead network's performance in real systems. We also compare the Runahead network with other related techniques. Last, we describe methods that we use to measure power and area of the proposed design.
- Chapter 5 presents the evaluation of the Runahead network. We show that the Runahead network can successfully deliver the majority of the packets that are injected. We also show the performance improvement in terms of packet latency and application execution time. We demonstrate that Runahead network is a lightweight accelerator in terms of power and area. The proposed network can also significantly reduce area and power usage when used as a power saver.

- Chapter 6 presents relevant works to the Runahead network. These include topics in multi-plane NoC designs, low-latency NoCs, QoS based NoCs, NoC prioritization schemes and critical word optimizations.
- Chapter 7 summarizes the thesis and discusses directions on how the Runahead network can be further explored in the future.

Chapter 2

Background And Motivation

Performance is always a driving force in the advance of technology. However, higher performance usually is achieved by using more energy and power. As modern CMPs are limited by power constraints, designing an energy efficient system becomes more and more important. NoCs have become a bottleneck both in terms of power and performance of a system. As a result, designers are looking for ways to decrease network latency while staying within the power and energy constraints of a system. To understand how to minimize network latency, we need to first understand compositions and factors that affect network latency. Next, we will discuss the major contributors to power in a modern NoC system and describe some of the ways to increase power efficiency. Last, we discuss the motivations behind the proposed Runahead network.

2.1 Network Latency

Network latency is defined as the cycles a message spends during its traversal in a network. It is proportional to the number of hops a packet needs to travel in order to reach its destination. However, to obtain the exact latency of a packet, we need to know how the message gets transmitted between routers and how the packet interacts with other packets in the network. How a packet interacts with other packets in the network is determined by flow control policies.

2.1.1 Flow Control

Flow control is the policy for allocation and arbitration of buffers and links in the network. In simpler terms, the flow control determines when a packet can traverse the next link on its route and when it has to wait at the router. An effective flow control policy would utilize network resources efficiently to provide low latency and high throughput regardless of system conditions. The network latency in a particular system is lower bounded by zero-load latency [34]. Zero-load latency is determined by multiplying the average distance a packet will travel with the per-hop latency of a router. As the name suggests, a packet experiences zero-load latency when there are no other packets in the network. Although the zero-load latency is set by topology and routing algorithm, flow control determines how close the the actual latency is compared to this theoretical limit in real systems. There are two categories of flow control techniques in network systems: Circuit-switching and packet-switching [16]. Each has their advantages and disadvantages.

Circuit-Switching

Circuit switching was first used in telephone networks where a dedicated connection is setup between the caller and callee. The connection stays active through the duration of the call. In a traditional circuit switched NoC, a probe message is sent from the source to the destination. The probe message sets up the link between the source and destination on a particular route. An acknowledgement is sent to the source once the destination receives this probe message, indicating the link is setup and ready to use. The source then sends the message along the pre-reserved links. A tear down message to free the reserved links is sent once the destination receives the entire message. During the message transmission phase, circuit switching network works like an ideal network where messages incur only wire delay. However, significant delay exists in setup, acknowledgement and tear down phases. Also, links are under-utilized and cannot amortize the large network latency of setup and tear down phases if only a small amount of data is transferred between source and destination. Circuit switched networks are also bufferless, which makes them power and energy efficient compared to a buffered network. In summary, a circuit switched network is energy efficient and acts like an ideal network when large data chunks are transferred, but suffers from poor link utilization and generally is not a good choice for most NoC systems [34].

Packet-Switching

Packet switching allocates resources on a hop-by-hop basis. Unlike a circuit switched network, where a probe message is sent to allocate entire link resources from source to destination, a packet switched network just injects the packet and allocates resources at each router when packets arrive. Packets wait at intermediate routers in case of contention. In a packet switched network, a message is divided into packets, and then subdivided into flits for network transmission. Most NoC systems use packet-switching flow control as it provides efficient resource allocations and arbitrations.

A basic technique in packet switching is store-and-forward flow control [16]. As the name implies, each router stores incoming flits and waits until the entire packet arrives before forwarding it to the next router. Buffer and link resources are thus allocated at packet granularity. This means that a head flit must wait for the tail flit to arrive even when there are network resources available for forwarding. This technique suffers from serialization delay at each hop due to the fact that a packet needs to be serialized in order to be forwarded to the next router.

An improved technique is called virtual cut-through flow control. It allows flits to be forwarded before the entire packet arrives at each router as long as there are enough storage spaces in the next router to hold the entire packet. Although the per-hop serialization delay is removed compared to store-and-forward, it still requires large buffer spaces in each router to keep packets moving forward.

Having large buffers are very expensive in terms of area and power usage, thus, wormhole flow control [15] is proposed to reduce the buffer requirement of virtual cut-through flow control. Unlike virtual cut-through and store-and-forward flow control that allocate resource at packet granularity, wormhole flow control allocates network resources at flit granularity. This finer granularity results in more efficient utilization of storage and bandwidth resources. Wormhole flow control also introduces its own problem which is "Head of line blocking". This is caused by the fact that flits are received and forwarded in a FIFO fashion. If the flit of a packet at the head of a queue is unable to proceed due to insufficient resources at the downstream router, all packets behind it also get blocked even though they may be forwarded to other downstream routers that are lightly loaded.

To solve head of line blocking, Virtual Channel flow control [14] is proposed. In virtual channel flow control, separate queues are available for different flows. There can be several virtual channels for one input/output physical channel. Virtual channels arbitrate for physical channels on a cycle by cycle basis. When the head of one VC becomes blocked, other packets can still utilize the physical channel in other virtual channels. This increases utilization of the link resources and improves overall network throughput. Most of the current NoC systems use VC flow control.

2.1.2 Virtual Channel Router Microarchitecture

Enabling VC flow control comes at a cost of complex router microarchitecture. Figure 2.1 shows the microarchitecture of a typical state-of-the-art credit-based virtual channel router. Typical VC routers have 5 logical pipeline stages. The following is a description of each pipeline stage:

- Route Compute (RC): All head flits need to determine the output port before they can arbitrate for the crossbar. RC is performed to calculate the incoming flit's output port at each router.

- VC Allocation (VA): All head flits need to arbitrate for a VC at a downstream router before they can be forwarded. In VA, each input VC chooses one output port, and places a request for a VC in the downstream router. Then, each VC in the corresponding output port (output VC) grants input VC's requests. Each output VC can only grant a single input VC request. Thus, in case of contention, output VC arbiters would decide which input VC's request to grant.

- Switch Allocation (SA): All flits arbitrate for access to the crossbar's input and output ports upon successful allocation of a VC. Similar to VA, in case of contention, the switch arbiters decide which request has priority over others.

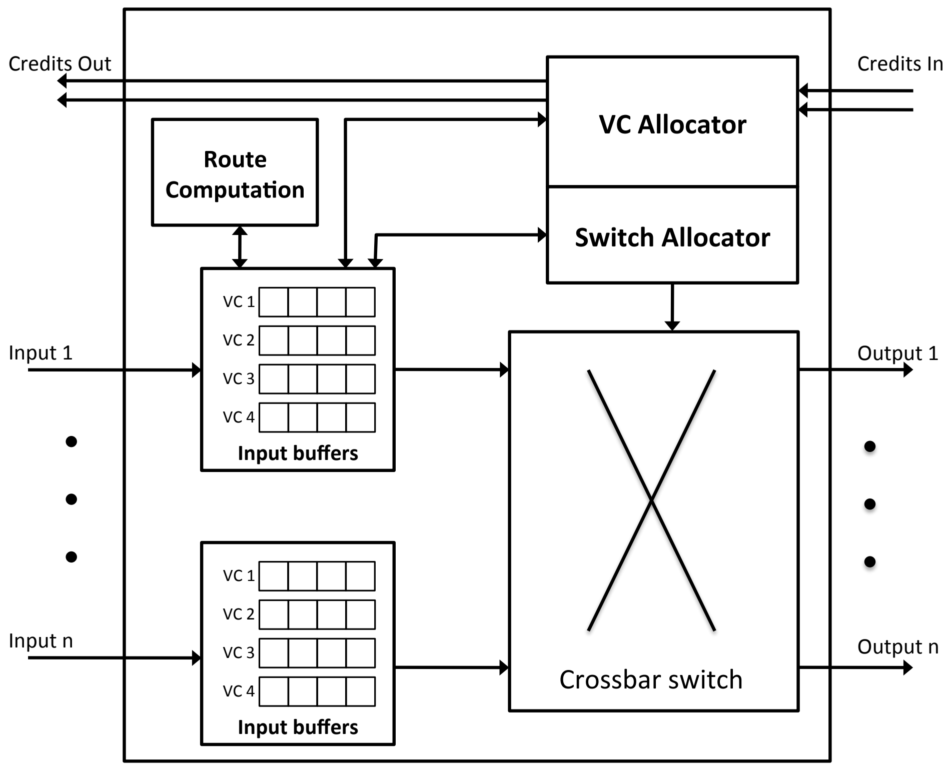


Figure 2.1: A typical VC router microarchitecture [34].

- Switch Traversal (ST): Flits that were granted access to the crossbar in SA traverse the crossbar switch.
- Link Traversal (LT): Flits coming from the crossbar's output traverse links to reach the downstream routers.

All five logical pipeline stages can be fit into a single clock cycle if the frequency is low. However, for aggressive clock frequencies, the router architecture must be pipelined. A typical VC router will have 5 physical pipeline stages just like the logical stages. The number of physical pipeline stages is directly related to the per-hop router delay. There are many exiting works that aim to reduce the pipeline stages of a router in order to decrease the per-hop router cycle delay. For example, by computing the route ahead of time [24, 29, 46] and combining multiple stages [50]. Details can be found in Chapter 6 on related work.

2.2 Power Constraints

NoCs consume a significant amount of power in modern CMP systems [27,32,55]. Thus, energy efficiency has been a primary concern for researchers and designers [10, 11]. For example, the network consumes 39% of the total power budget in Intel's 80-core TeraFLOPS router [32]. The primary contributors to

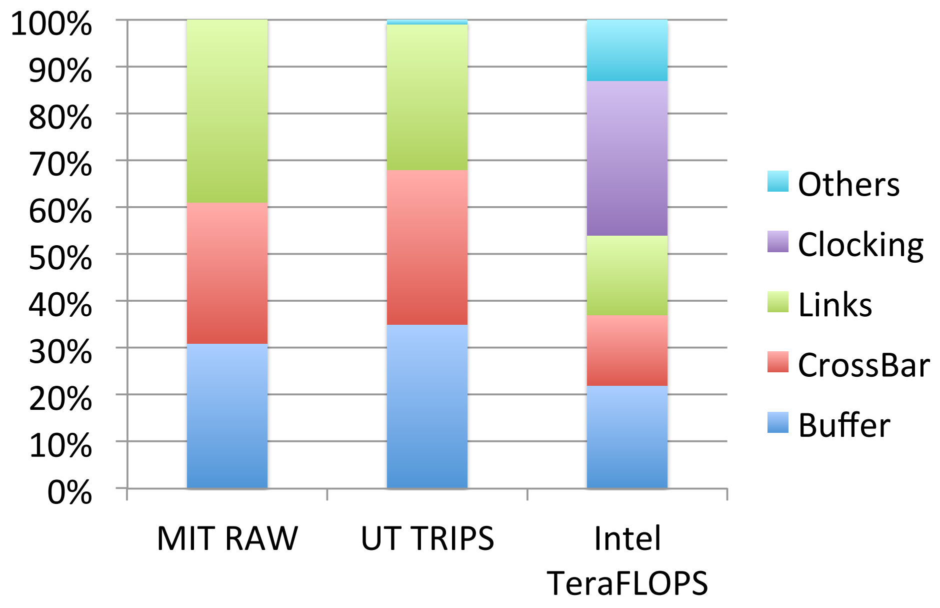


Figure 2.2: Total power contribution of existing NoC systems [27, 32, 55].

NoC power are buffers, crossbar switches and links. Figure 2.2 shows the power composition for each of the 3 components in their systems. Note that only Intel's 80-core TeraFLOPS system reported clocking power consumption. In the TeraFLOPS system, the next largest contributors after clocking are buffers. Buffers, crossbar switches and links contribute 99% of the total power used by the networks in other 2 systems.

To save power, one option is to eliminate NoC router buffers as they are large contributors of NoC power. Buffers are the only router microarchitecture components that can be removed with modifications to other aspects of a network. On the other hand, crossbars and links are essential to make the network functional. The crossbar switches are necessary to allow incoming flows to make turns cycle-by-cycle, and the links between routers and cores are necessary to deliver the packets to their destination. Although it is possible to optimize crossbars and links for energy efficiency, these optimizations are usually complex. Eliminating NoC router buffers is a relatively simple option to save power. Several bufferless NoC designs had been proposed in the past [21, 28, 43, 45]. These designs achieve significant power savings but suffer from lower saturation throughput compared to conventional buffered networks.

2.3 Motivation for the Runahead network

Low Channel utilization in real system workload

To motivate the Runahead network, we first notice the fact that link utilization is usually very low in real system workloads. Previous analysis [5, 26, 31] of different single-threaded and multi-threaded workloads for CMPs discovered that average injection rates are around 5%. With such low injection rates, channel utilization will also be low. As a result, most of the NoC resources are over-provisioned for worst-case bandwidth demands. Also, during our preliminary studies, we found that a large portion of the packets that are sent through the network are single-flit packets [41].

Because the channel utilization is low, the average network latency within a NoC is close to the zero-load latency as defined in Section 2.1.1. This means many of the complex components in the VC router that enforces flow control policies are under utilized. These components are capable of managing heavy loads for worst-case bandwidth demands, but instead, they only provide simple arbitrations and allocations when the network is lightly loaded for the majority of the time.

Buffers are an expensive resource

As discussed in Section 2.2, buffers are an expensive resource. A network would be more energy efficient if buffers are eliminated. A combination of bufferless and buffered NoC in a multi-NoC system may provide improved performance and save power.

The Runahead network takes advantage of these observations and is designed to only handle the common scenarios where a single-flit packet travels through the network without contention while keeping the energy usage low. To achieve this, the proposed network drops packets according to a pre-defined port-arbitration. Because the complex arbitration mechanism is removed, a packet can be forwarded in one cycle at a router. To ensure the correctness of applications, we need to ensure a packet that gets dropped eventually reaches to its destination. The Runahead network relies on a conventional, lossless NoC to provide guaranteed delivery. The main objective for the Runahead network is simplicity. It will have faster delivery time compared to the conventional NoC due to the one cycle per hop delay. The selected packets that travel via the Runahead NoC will arrive earlier with deterministic latencies and speed up the application.

The Runahead network is simple and lightweight. It provides ideal and deterministic network latency if a packet is delivered successfully. The Runahead network targets single-flit packets as they contribute a significant portion of the total packets that are sent through the network in real applications. The proposed network also eliminates the need for buffers as it will drop any packets in case of contention according to a pre-defined port arbitration. Combined with a conventional packet switched network, the combination will have both the benefit of a buffered and a bufferless network.

Chapter 3

The Runahead Network

In this chapter, we present our Runahead NoC architecture. We first give a high-level overview of how the Runahead network can be used to either accelerate performance or save power (Section 3.1). In Section 3.2, we give an overview of our proposed Runahead router architecture. Then we describe how routing computation and port arbitration are performed (Section 3.3), enabling the single-cycle per-hop latency. Finally, we discuss how we apply critical word forwarding for data packets (Section 3.4) and how to integrate our Runahead network with the regular network (Section 3.5).

3.1 Overview

The Runahead network is a lightweight, lossy network, designed to achieve a single-cycle per-hop latency. It is meant to be paired with a regular lossless NoC, which provides guaranteed delivery of all packets. The Runahead network can be 1) added to a regular NoC as an *accelerator*, providing low-latency transmission of latency-sensitive packets, or 2) integrated into a regular NoC as a *power-saver*, providing power and area savings without harming performance.

3.1.1 Use As an Accelerator

When used as an accelerator, the configuration of the existing regular network is left unchanged and its operation is undisturbed. All packets are injected into the regular network, while only latency-sensitive single-flit packets are injected into the Runahead network. Multi-flit packets are excluded to minimize the complexity when one or more flits of a packet are dropped. In a cache-coherent CMP, the Runahead network carries all control packets, which are typically single flit. It also carries single-flit data response packets. These packets are sent in response to a cache miss and only contain the critical word (i.e., the initially requested word) of the data block. This is described in more detail in Section 3.4. Since the regular network is lossless, any packets dropped by the Runahead network will still arrive at their destination. The Runahead network achieves very low packet latency due to its simplified router architecture, which enables only a single-cycle delay per hop. The goal of the accelerator is to provide an opportunity for ultra-fast delivery of latency-sensitive packets while incurring low power and area overheads.

3.1.2 Use As a Power-Saver

When used as a power-saver, the existing regular network is under-provisioned to allow for the integration of the Runahead network. As in the accelerator case, the Runahead network only carries latency-sensitive single-flit packets. The regular network still carries all packets to guarantee delivery of any packets that may be dropped by the Runahead router. In our experiments, we consider two different configurations: 1) a regular multi-NoC system and replace one of the subnetworks with our Runahead network and 2) partition half of the channel width in a single regular NoC for the Runahead Network. The Runahead network consumes very little power and area. This is because it is bufferless and consists of a simplified router architecture with no routing tables nor complex arbiters. Despite the increased congestion in the smaller regular network, overall application performance is unharmed since latency-sensitive packets are transferred very quickly. The goal of the power-saver is to minimize area and power consumption while maintaining comparable or better performance.

3.2 The Runahead Routers

To achieve single-cycle hops and ensure low area and power consumption, the routers in the Runahead network need to be simple. In this work, we specifically target a 2D mesh topology, which is commonplace in modern systems (e.g., Tilera [58] and Intel Terascale [32]). Figure 3.1 illustrates the design of the Runahead router. It consists of five multiplexers: one for each of the output ports of the four cardinal directions and one for the ejection port. The Runahead routers share the same injection port as the routers in the regular network. Runahead routers are bufferless. Only four registers are needed to store up to four single-flit packets that may come in from the input ports of the 4 cardinal directions at any cycle. Injected packets are stored in the input buffer in the regular router. When a packet is injected into the Runahead network, the header information is extracted from incoming packets at the input ports and directed to the routing computation and port arbitration unit. For clarity, data connections are not shown in the figure.

3.2.1 Lossy Delivery

We design our Runahead routers for XY dimension-order routing (DOR), which greatly simplifies routing and arbitration. In XY-DOR routing, a packet travels the network in the X-dimension first before travels in Y-dimension. Port arbitration directs packets from input ports to their corresponding output ports

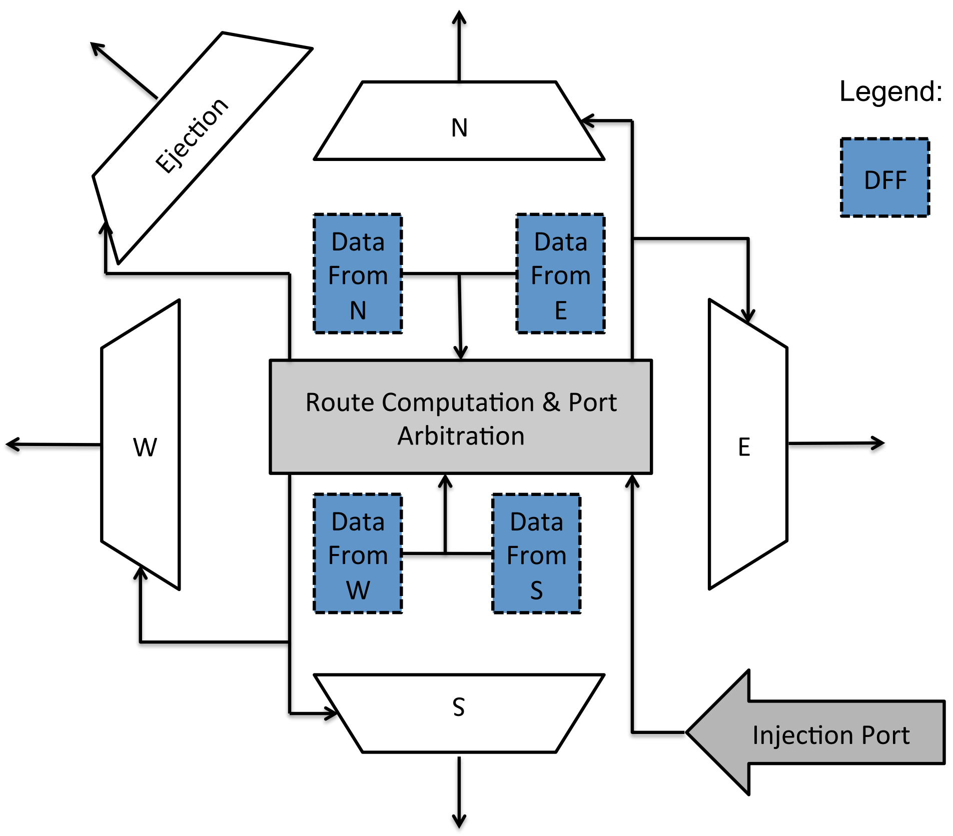


Figure 3.1: Runahead router design

and determines which packets to drop in the event of a conflict. The Runahead router does not collect dropped packets, and the Runahead network does not try to forward them again. This is different from prior work, such as SCARAB [28] and BPS [25], where a NACK is sent to the source for packet re-transmission. In the Runahead network, the dropped packet will always be delivered by the lossless regular network. The Runahead network is inherently deadlock-free since packets are dropped instead of blocked upon conflicts. This eliminates the needs for complex deadlock prevention mechanisms. Section 3.3 describes the details of routing computation and port arbitration.

3.2.2 Single Cycle Hops

Unlike conventional virtual-channel routers with 3 to 5 pipeline stages, the Runahead router delivers packets in a single cycle per hop. This is done by combining the route computation, port arbitration and link traversal all into a single step. The Runahead routers are hardcoded for XY DOR; no routing tables are necessary. The output port of an incoming packet is quickly determined from the destination information in the header. By allowing for dropped packets, the Runahead router greatly simplifies port arbitration. Our design uses a fixed arbitration scheme where the priority of incoming packets is static for each output port. Specifically, our port arbitration scheme always prioritizes packets going straight over packets turning. For example, when contending for the north output port, the south input port always beats out the east and west input ports. Thus the logic that controls the output port multiplexers is very simple, allowing the entire process to fit within one cycle. Since all hops are single cycle, the latency of a packet is fully deterministic from source to destination, assuming it is not dropped along the way. The latency in cycles is equal to the number of hops travelled. Because of this, it is impossible for a packet to arrive on the regular network earlier than on the Runahead network.

3.2.3 Single-Flit Packets

Since there are no buffers, the Runahead router does not have virtual channels, just a single physical channel per output port. Handling multi-flit packets introduces too much complexity in the Runahead network since at any moment, any flit can be dropped. Because of this, it is possible for a multi-flit packet to arrive at a destination with some flits missing. We would need additional complexity at the cache controllers and memory controllers to support incomplete data packets. Thus, our design only supports single-flit packets to minimize overheads and complexity.

3.3 Route Computation and Port Arbitration

Route computation and port arbitration are performed together as one unit in the router to allow packets to traverse each hop in a single cycle. The destination is encoded in the header as signed X and Y values that indicate the relative number of hops remaining until the destination. The sign indicates the direction of travel. Thus route computation is a simple matter of determining the correct output port based on these values.

We employ a fixed arbitration scheme for each output port. In our design, a packet that is going straight has higher priority than a packet that is turning. If packets from two different input ports are turning towards the same output port, one of the input ports is hardcoded to always take precedence. For example, if both the east and west input ports are contending for the north port, the arbitration always selects the west port. Similarly, for the ejection port, arbitration is hardcoded such that specific input ports always win. This minimizes complexity and allows us to combine the route computation and port arbitration steps into a single cycle.

With XY DOR routing, our fixed arbitration scheme yields only three places where a packet can be dropped:

1. At injection,
2. When the packet is making a turn from the X to Y direction, or
3. At ejection when a routing conflict occurs at the ejection port.

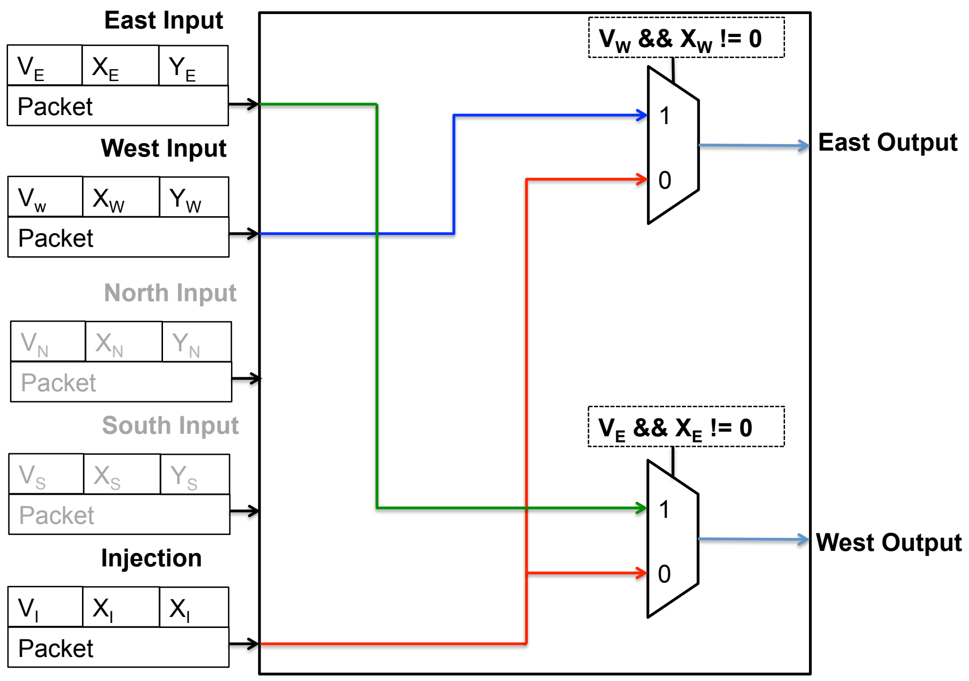


Figure 3.2: Route Computation and port arbitration for East and West output port

This applies to all packets no matter how far they are traveling. As a result, the number of places where a packet can be dropped remains constant and does not scale with the network size.

The route computation and port arbitration unit is shown in Figures 3.2 3.3 and 3.4. The figures are separated to show the logic for different output ports. The inputs are obtained from the header information of the incoming packets at each input port. The required signals from each input packet is denoted by $X_{direction}$, $Y_{direction}$, $V_{direction}$, which correspond to the destination X and Y values and a valid bit; the valid bit simply indicates that there exists a valid packet waiting at the input port. In parallel to route computation and port arbitration, the corresponding X or Y value in the packet header is updated for the next hop.

East and West Output

The advantage of using XY DOR is that it simplifies east and west routing and arbitration, as shown in Figure 3.2. The east and west direction output ports only need to consider the latches of their opposing input ports, as well as the injection port. Anytime a packet arrives at either the east or west input port with a non-zero X value, it is guaranteed to be forwarded straight since it has the highest priority in our fixed arbitration scheme. It is impossible for a packet to turn on to either the E or W directions. It is also impossible for a packet to be forwarded back to the direction from which it arrived.

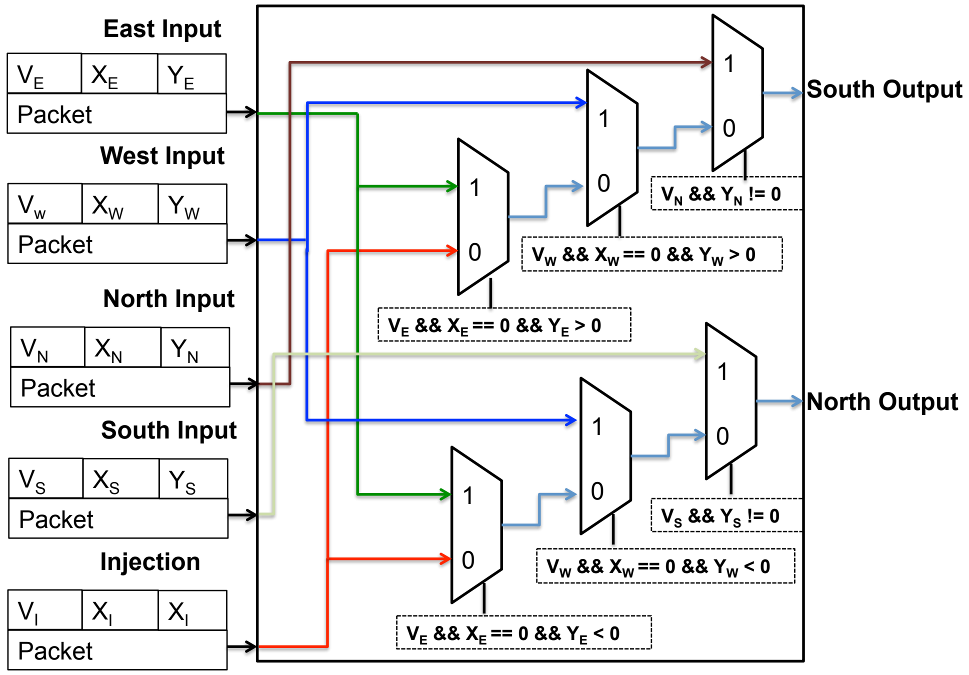


Figure 3.3: Route Computation and port arbitration for North and South output port

North and South Output

Routing and arbitration are more complicated for the north and south output ports since they need to consider packets that are turning. The structure is shown in Figure 3.3. Arbitration is hardcoded such that the outermost multiplexer always chooses the opposing input port if there is a valid incoming packet; this enforces our straight-first arbitration. The logic for this only needs to look at the Y value in the header to check that the packet has not reached its destination yet. If there is no valid packet at the opposing input port, the fixed arbitration scheme first checks to see if the west input port is turning, followed by the east input port, and finally the injection port. This implies that with our implementation, a packet at the west input port always takes precedence over a packet at the east port when both of them are trying to turn to the same output port. A packet at the east or west input port is determined to be turning if it contains a zero X value with a non-zero Y value. Note that a packet traverses at most three 2-to-1 multiplexers from its input port to its output port, keeping the critical path delay low.

Ejection Output:

The ejection port, shown in Figure 3.4, is similar to that of the north and south output ports. In our implementation, incoming packets are ranked based on the following order of input ports: N, S, W, E. As with the north and south output ports, an ejecting packet traverses at most three multiplexers. To determine if a packet is destined for the ejection port, both the X and Y values need to be zero. We

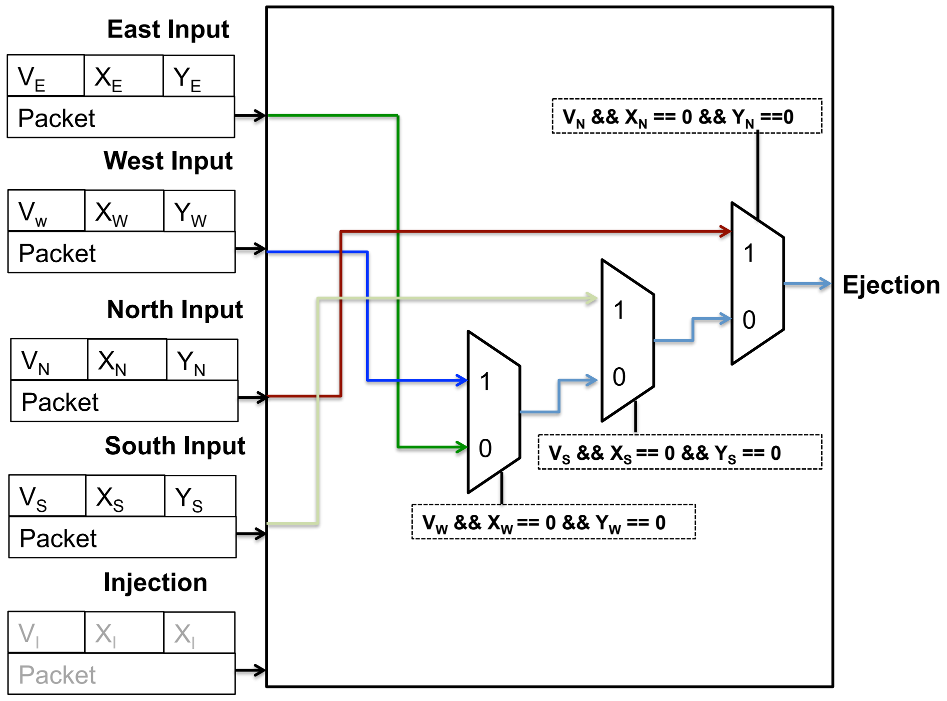


Figure 3.4: Route Computation and port arbitration for Ejection port

assume that a packet will never be injected with the same source and destination nodes, thus eliminating the need to connect the ejection port to the injection port.

3.3.1 Starvation and Fairness

Note that the port arbitration scheme is stateless. There is no mechanism to detect and prevent starvation. For example, when arbitrating for the north output port, packets at the south input port will always take precedence over those at the east and west input ports even if it means the packets at these two ports are always dropped. This keeps complexity at a minimum in the Runahead router architecture, allowing for low overhead and ensuring that the design fits within a single clock period.

Hardcoding arbitration leads to some potential unfairness or starvation. However, since all data injected into the Runahead network is also injected into the regular NoC, forward progress for the application is guaranteed. Similar with fairness, the goal of the Runahead network is not to provide fair communication because the regular network that the Runahead network relays on already provides such platform. In Chapter 5, we evaluate the potential unfairness that arises in our network; in practice, arrival rates are relatively uniform across all source nodes. Due to low contention in these networks, packets are often delivered successfully mitigating any concerns about fairness. Without mechanisms to prevent starvation and ensure fairness, the Runahead network has less overhead once it is combined with a regular NoC.

3.4 Critical Word Forwarding

The Runahead network is meant to carry only latency-sensitive packets. In a cache-coherent CMP, all control packets (i.e., requests, invalidations, acknowledgements) and data response packets are latency-sensitive. However, the Runahead network is designed for single-flit packets to avoid the complexity of dropped flits in multi-flit packets. As a result, data response packets cannot be carried on the Runahead network. To compensate for this, we apply critical word forwarding optimization. A word is critical if an instruction that requires it is issued in the processor pipeline before the word arrives at the L1 cache. This word causes processor to stall and hurts performance. The critical word is the most latency-sensitive word in the data block. Fortunately, 67% of the critical words on average are the first word in a cache block in real applications [12]. Also, many modern CMPs can support critical word forwarding. When composing the data response packet, the initially requested (critical) word is sent in the first flit. This way, when the first flit arrives at the L1 cache, the critical word is forwarded directly to the processor so that it can continue executing before the rest of the packet has arrived. Thus in our implementation, we assume critical word forwarding and inject the first flit of all data response packets (bound for the L1 cache) into the Runahead network.

3.5 Integration to the Regular Network

The Runahead network can be easily integrated with an existing network. It is a stand-alone NoC that simply connects its routers' injection and ejection ports to the injection and ejection queues of the regular network. The injection port of each Runahead router connects directly to the head of the regular input buffer, so that single-flit packets waiting to be injected into the regular network are also injected into the Runahead network. In our experiments, we find that a large portion of packets are dropped at injection, accounting for up to 50% of all dropped packets in the Runahead network. This is because in port arbitration, packets from the injection ports have lowest priority, as explained in Section 3.3. To improve this, we design the Runahead router to try to inject a packet multiple times for as long as it is at the head of the injection queue. If the packet at the head of the queue does not succeed in arbitrating for its output port, we try again in the next cycle if the packet is still at the head (i.e., if the packet has not yet been injected into the regular network either). If the packet is injected into the Runahead network successfully, a flag is set at the input port so that we do not try to inject it again in subsequent cycles.

3.5.1 Preventing Duplicated Ejection

The ejection ports are connected to the regular output buffers. When a packet is ejected from the Runahead network, it is immediately forwarded to the corresponding cache controller or memory controller. To keep a simple interface between the network and the processors, ejected packets need to be filtered to prevent duplication seen by the processor. The baseline router will have a small buffer that records ejected packets and drops the same packet that arrives at a later time. This ensures successfully delivered packets via the Runahead network are not sent to the controllers twice. Note that a packet will never be ejected from the regular network before the Runahead network, as discussed in Section 3.2. This further simplifies the design and operation of the small buffer at the ejection port. An entry that records the packet ID is created in the buffer when a packet arrives via Runahead network. When a packet arrives from the regular network, a lookup in the buffer would determine if this packet should be discarded. The packet is dropped if the same ID exists in the buffer, indicating this packet has already been ejected. The entry is then removed to free up space for subsequent entries. For the applications we have studied (PARSEC), the maximum number of entries that a buffer needs to hold is 15. Assuming the packet ID is 8 bytes (the length of a control packet), this buffer would be 120 bytes, which is very small in terms of size.

In the unlikely event that the buffer is full, the network interface will discard packets that arrive on the Runahead network. This is safe since any packet that arrives in the Runahead network will also arrive on the regular network.

3.6 Chapter Summary

We presented the details of the Runahead network in this chapter. In order to keep the design simple, the Runahead network uses a pre-defined port-arbitration scheme and drops packets if there is contention. The crossbar switches are very simple and are controlled by information readily available in the flit headers. To further increase performance, the Runahead network forwards critical words to the processor to decrease stall time. In the next chapter, we will present the methodology of our evaluation.

Chapter 4

Methodology

In this chapter, we discuss the methodology and simulation environment of our experiments that evaluate the performance, power and area of the Runahead network. We first evaluate the performance using synthetic traffic to isolate specific attributes such as arrival rate and throughput of the Runahead network. Next, we present how we used SynFull to evaluate the packet latency of the Runahead network in comparison with other network setups. We then talk about our experimental setups in full-system simulation. Finally, we present how we measured area and power using DSENT [54] and RTL.

4.1 Synthetic and SynFull Traffic

4.1.1 Synthetic Traffic

To evaluate the performance of our Runahead network, we used a modified version of Booksim, a cycle-level network simulator [35]. Several synthetic traffic patterns are used to cover a wide range of network utilization scenarios. Descriptions for each traffic pattern are listed in Table 4.1. All configurations use an 8×8 2D mesh network. The configuration parameters for our regular network are listed in Table 4.2.

Traffic Pattern	Description
Uniform	Each source sends an equal amount of traffic to each destination
Bitcomp	Bit complement. Bits in destination node ID are the complement of bits in source node ID
Bitrev	Bit reverse. Bits are reversed in destination node ID compared to source node ID
Right neighbour	Source sends to its right neighbor. For right edge nodes, they send to the left edge nodes 1 row below
Transpose	Destinations are the transpose of the sources in a mesh topology
Asymmetric	Sources in upper half send to lower half nodes in the same column at same distance from the middle and vice versa

Table 4.1: Synthetic traffic pattern descriptions

Topology	8×8 mesh
Channel width	8 byte
Virtual channels	6 per port (4 flit each)
Router pipeline stages	3
Routing algorithm	X-Y dimension-order
Flit size	8-byte
Control/Data packet size	1 Flit/9 Flits

Table 4.2: Regular network simulation parameters for synthetic and SynFull traffic simulations

Topology	8×8 mesh
Channel width	10 byte (8B for flit, 2B for other metadata)
Virtual channels	None
Routing algorithm	X-Y dimension-order
Flit size	8-byte

Table 4.3: The Runahead network simulation parameters for synthetic and SynFull traffic simulations

The Runahead network simulation parameters are listed in Table 4.3. To evaluate the effectiveness of the Runahead network in conjunction with the regular network, we combine the Runahead network with the regular network. We run the same set of synthetic traffic patterns listed in Table 4.1 with all single-flit packets except for right neighbour traffic. We cannot saturate the combined or regular network with right neighbour traffic due to the amount of network resource we have in our configuration. Even at 100% injection rate, the network can still forward packets with normal latency. For other traffic patterns, we measure the latency and throughput of the combined network compared to the regular network.

4.1.2 SynFull Traffic

To further evaluate our Runahead network design, we use multi-programmed SynFull traffic workloads [3]. SynFull workloads are designed to reproduce the cache coherent behavior of multi-threaded applications from SPLASH-2 [59] and PARSEC [7]. These workloads consist of single-flit control packets and multi-flit data packets. For each 64-core workload, we run 4 identical instances of a 16-way multi-threaded application. Each instance is assigned a 4×4 quadrant of cores. For SynFull, memory controllers are located at the left and right edge nodes of the 8×8 mesh. All four workloads send memory traffic throughout the chip.

SynFull simulation network setups

We use SynFull to compare our proposed Runahead network with two other multi-NoC systems and the regular network. The SynFull setups for each configuration are as follows:

- **Regular-NoC:** This setup has a single lossless NoC with the same configuration as specified in Table 4.2. Total channel width is 8 bytes.

- **Multi-NoC_Random:** In this setup, the NoC is composed of two independent Regular networks with the same configuration. Total channel width is 16 bytes (8 bytes + 8 bytes). The workload is shared evenly between the two NoCs (i.e., 50% of traffic is randomly injected into each network).

- **Multi-NoC_Select:** In this setup, the NoC is configured identically to that of Multi-NoC_Random. However, instead of sharing the traffic evenly, Network 1 is responsible for latency-sensitive traffic (i.e., packets that we would inject into the Runahead network). This includes single-flit packets and critical words. Network 2 handles all other traffic. Since delivery is guaranteed, single-flit packets are only injected into Network 1 instead of both networks.

- **Combined_NoC:** In this setup, we have a single Regular network, which carries 100% of the injected packets, along with the proposed Runahead network that carries latency-sensitive packets (i.e., single-flit packets and critical words). As delivery is not guaranteed in the Runahead network, duplicate injection of latency-sensitive packets into both networks is required. The total channel width in this case is 18 bytes (8 bytes for the Regular network and 10 bytes for the Runahead network). To enable critical word forwarding, we may need additional metadata that is not included in the flit headers. Every cache miss creates an entry in an MSHR (Miss Status Handling Register) in most modern processors. The metadata should contain the index into the MSHR so the proper misses are handled. Assuming each MSHR has 32 entries, we only need 5 bits (less than one byte) to index every entry. We allocate two extra bytes to the Runahead network channel width merely to be conservative to account for any additional metadata for supporting critical word forwarding. This does not give our Runahead network a performance advantage since all packets are single-flit; in fact, it incurs a power and area disadvantage.

To keep measurements consistent across all four setups, we only measure the latency of unique packets that are seen by the application. This means that if a packet is injected into both the Runahead and regular networks, we only measure the latency of the packet that arrives first; subsequent arrival of the same packet is discarded by the NoC. For data packets, latency is taken for the entire packet to arrive, not just the critical word. To measure the potential benefit of accelerating critical words, we report the difference in arrival times between the critical word and the rest of the data block.

To investigate the fairness of the port injection scheme, we use the source node arrival rate to show the percentage of packets arrived that originated from a particular source node. This measure would indicate if a particular direction or section of the network is experiencing unfairness. We do this for both synthetic traffic patterns and SynFull simulations.

# of Cores/Threads	16/16, 1GHz
Private L1 Cache(D & I)	16KB 4-way LRU 64Byte blocks
Shared L2 Cache	fully distributed, 8-way LRU, 4 MB total
# of directories	4 directories located at each corner of the topology
Cache Coherence	MOESI distributed directory

Table 4.4: Full-system simulation system parameters

Topology	4×4 mesh
Channel width	16 byte in Baseline NoC and accelerator mode, 8 byte in power saver mode
Virtual channels	6 per port (4 flit each)
Router pipeline stages	3
Routing algorithm	X-Y dimension-order
Flit size	Same as channel width
Control/Data packet size	1 Flit/5 Flits in Regular NoC and accelerator mode, 1 Flit /9 Flits in power saver mode

Table 4.5: Full-system baseline network simulation parameters

4.2 Full-System Simulation

To evaluate the real system performance of the Runahead network, we used Gem5 [8] and Booksim [35]. The system parameters are listed in Table 4.4. All configurations use a 4x4 mesh topology with multi-threaded workloads from SPLASH-2 [59] and PARSEC [7]. These workloads consist of single-flit control packets and multi-flit data packets. For each multi-threaded workload, we run 16 threads with the simmedium input set until completion. We measure the execution time by recording the number of ticks in the application's region of interest reported by Gem5. For full-system simulations, the memory controllers are located at the corner of the mesh network. We keep the cache sizes low in an effort to provide greater stress on the network.

For a baseline comparison, we use a regular network with simulation parameters listed in Table 4.5. We changed the regular network's channel width from 8 to 16 bytes. Control flits are still 8 bytes, but data packets only need 5 flits instead of 9 flits due to the increased channel width. The wider channels allow us to compare different NoC setups fairly as they have the same channel bandwidth for data transmission. We also want to explore the performance of the Runahead network when it utilizes part of regular network's channel resources instead of replacing an entire network in a multi-NoC design. This enables us to evaluate the effect of a lossless NoC with narrower channels in the Runahead Network compared the regular NoC with wider channels. The parameters used for Runahead network is listed in Table 4.6. Note that the channel width for the Runahead network has 2 extra bytes, this is to be conservative to account for any additional metadata. This does not give our Runahead network a performance advantage since all packets are single-flit; in fact, it incurs a power and area disadvantage as discussed before.

To evaluate the performance benefits of the Runahead network in accelerator mode, we combined

Topology	4×4 mesh
Channel width	10 byte (8B for flit, 2B for other metadata)
Virtual channels	None
Routing algorithm	X-Y dimension-order
Flit size	8-byte

Table 4.6: The Runahead network simulation parameters for full-system simulations

the proposed network with the regular network. Next, we half the channel width of the regular network in the proposed network to allow for a equal channel width comparison between the regular network and the Runahead network. In this case, the Runahead network operates in power saver mode. We also compared the Runahead network in power saver mode with two other existing works: Aergia [52] a prioritization scheme, and DejaVu switching [1], a multi-plane NoC design.

Aergia is a prioritization scheme that uses the notion of slack to determine the priorities of packets. Like any other prioritization schemes, arbiters and allocators will first grant requests from packets with higher priorities before requests from lower priority packets. Aergia calculates packet priority based on local slack which is defined to be the number of cycles a packet can be delayed without delaying any subsequent instructions. A packet with lower slack will have higher priority. Aergia uses three indirect metrics to estimate the local slack of a packet. First is the number of L2 miss predecessors which is the number of older requests sent by the same core with L2 misses. Packets with zero miss predecessors will likely have low slacks as their latencies are unlikely to be overlapped by their predecessors. Second metric is L2 hit/miss status. This indicates if a packet will be a L2 hit or a miss. If the packet is a L2 miss, it is likely to be a critical packet with low slack due to the high latency associated with memory access. This high latency is less likely to be overlapped by other packets. The last metric is the hop count that a packet will travel compared to its predecessors. Packets with smaller hop counts will arrive at their destination faster, thus, will have higher slacks compared to packets with higher hop counts. Aergia combines these three metrics to obtain the priority of a packet before injection.

DejaVu switching is a multi-plane NoC design where single-flit control packets and multi-flit data packets are separated in to different planes. To enable fast delivery of data packets, reservation packets are sent out first on the control plane to reserve network resources on the data plane routers. The reservations enable the data packets to be forwarded without suffering delays associated with making routing decisions at every router. With reduced data packet latency, the frequency and voltage of the data plane can be reduced to save power while maintaining performance.

Full-system simulation network setups

The following are our full-system simulation setups and simulation parameters for the five different NoCs.

- **Regular NoC** uses parameters listed in Table 4.5. Its total channel width is 16 byte.
- **Accelerator Mode** In this setup, we use the Runahead network as an accelerator. The Runahead

network is added directly to the regular NoC with parameters listed in Table 4.5. The total channel width for this setup is 26 byte (16B base + 8B Runahead data + 2B Runahead metadata)

- **Power Saver Mode** In this setup, we use the Runahead network as a power saver. We partition half of regular NoC's channel width for Runahead network, making the regular network to have 8 byte channels. The total channel width in this setup is 18 byte. (8B base + 8B Runahead data + 2B Runahead metadata)

- **Aergia** We use the same setup as the regular network. Total channel width in this setup is 16 bytes. However, we change the prioritization scheme for allocations and arbitrations to Aergia. We calculated the slack similar to Aergia [52]. However, we assumed a perfect L2 miss predictor so we can calculate the slack more accurately. Packets that have lower slacks will have higher priority during arbitration and allocation of network resources.

- **DejaVu** In this setup, we use the regular network in power saver mode (8 byte channel) for the control plane. The control plane uses same parameters as the regular network in power saver mode listed in Table 4.5. The reservation packets are sent 3 cycles ahead of data packets and they travel in the control plane. We assumed the reservation queues in the data plane have infinite size. To model the simplified router design in the data plane of DejaVu switching, we use routers with one cycle router delay with only 1 VC in our simulation. Other simulation parameters are the same as a regular network in power saver mode. Also, we forward the critical words to the processor as soon as the head of the data packets arrive. We kept the frequency of both control and data plane the same as the system clock so it can deliver its full performance benefits.

We first show the performance benefits of the Runahead network with/without the critical word forwarding optimization compared to the regular network. In subsequent evaluations, we only use the Runahead network with the critical word forwarding optimization as it delivers better performance.

4.3 Power and Area

4.3.1 DSENT setup

Power and area of the 8x8 network are modelled using DSENT [54]. Results are collected using a 22nm bulk/SOI, low-V process node. The network configurations and parameters used are the same as SynFull and synthetic traffic pattern simulations. All parameters are listed in Table 4.2 for the regular network and in Table 4.3 for Runahead network. Dynamic power is obtained by modeling the system using the average injection rates collected from the SynFull workloads.

4.3.2 RTL setup

To ensure the feasibility of the Runahead router, we use an open source RTL router design [6] as a conventional router with the same setup listed in Table 4.5. The Runahead router is constructed on top of the existing RTL design. We use Synopsys design compiler with TSMC 65nm technology to evaluate the power and area for a single Runahead router. RTL only models a single router and links between routers are not modeled. We used the default toggle rate of 0.1 which gives an average wire activity factor of 0.025 toggles per cycle for all nets. This would model a system with low injection rate. We study the power and area usage in regular network, accelerator mode and power saver mode described in Section 4.2 under WCCOM (worst case commercial) operating conditions with Zeroload wire load model. Since we only have access to the older TSMC 65nm technology library, the power and area results reported by RTL modeling are higher compared to DSENT's 22nm technology models.

4.4 Chapter Summary

In this chapter, we presented our methodology and simulation environments to evaluate the Runahead network. We first presented the network setup for synthetic traffic and SynFull simulations, then we presented different setups and parameters used in full-system simulations. Finally, we presented how we measured power and area using DSENT and RTL. In the next chapter, we will present the evaluation results.

Chapter 5

Evaluation

This chapter provides performance, power and area evaluations of our proposed Runahead network. We first evaluate the arrival rate in the Runahead network under synthetic traffic, followed by a packet latency evaluation of the Runahead network using real application models from SynFull. Next, we show the results from full-system simulation of the Runahead network using Gem5 and Booksim. Finally, we report area and power usage of the Runahead network.

5.1 Synthetic Traffic Patterns

In this section, we evaluate the Runahead network using synthetic traffic patterns. This enables us to isolate the performance of the Runahead network. Since the proposed network can only support single-flit packets, we only use single-flit packets in our synthetic traffic.

5.1.1 Arrival Rate in the Runahead Network

The Runahead network is only effective at improving performance if a large fraction of packets are successfully delivered. To measure the arrival rate of the Runahead network, we omit the regular network and subject the Runahead network alone to synthetic traffic. The arrival rate is calculated by taking the ratio of injected packets and arrived packets. All packets used in this simulation are single-flit.

Figure 5.1 shows the arrival rate with different synthetic workloads. All nodes need to send 10k packets to a destination depending on the traffic pattern specified. On average, the Runahead network delivers over 90% of injected packets when the injection rate is less than 7%, and delivers half of all injected packets when injection rate is less than 67%. As expected, the average arrival rate decreases exponentially as injection rate increases linearly. With more packets in the network, the greater the likelihood of a routing conflict. However, there are a few exceptions where the arrival rate decreases linearly with injection rate. Right neighbour and asymmetric traffic patterns' arrival rates have a linear

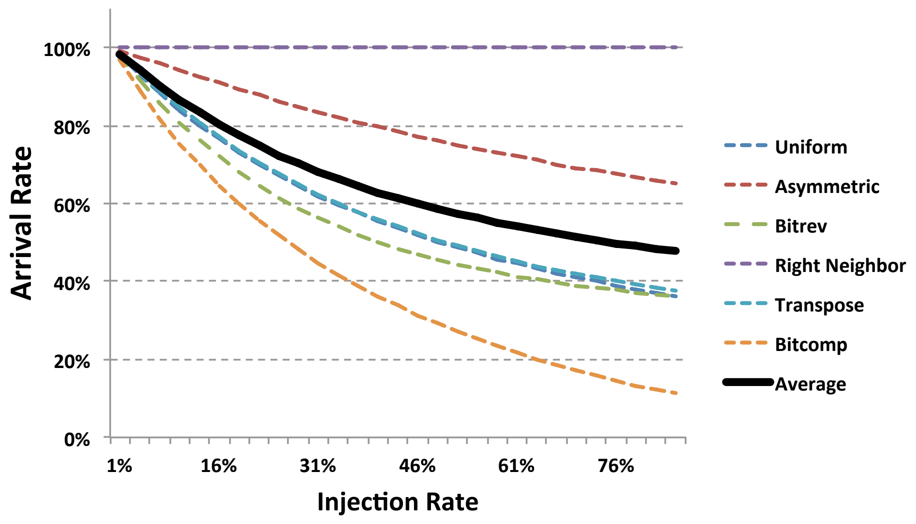


Figure 5.1: Packet arrival rate for synthetic traffic patterns

relationship with the injection rate. For these two traffic patterns, most packets travel in one direction. For neighbour traffic, packets usually go to their right neighbour. For the right edge nodes, packets are sent to the a left edge node one row below. Packets do not encounter routing conflicts due to the direction they are travelling in combination with XY routing. There is only a single source-destination pair at a given router. Packets are always travelling in the opposite direction if they meet at the same router. For example, most of the packets are travelling to the right, some packets that are sent by the right edge node would travel left and down, and a few packets that need to travel diagonally from the lower right to upper left. It is impossible for a packet to get dropped in this traffic pattern because the use of XY routing. Figure 5.1 confirms that there are no dropped packets for neighbour traffic patterns. This gives us the linear relationship between the injection rate and the arrival rate.

For asymmetric traffic, a node in the lower half of the network sends to a single destination in the same column in the top half of the network or vice versa. The only two directions that a packet can travel is up or down. Packets are always travelling in the opposite direction if they meet at the same router. Although similar to neighbour, asymmetric traffic's packets travel more hops on average. These packets pass routers that may want to inject a packet in the same direction in the same cycle. As injection rate increases, the chances of this also increases, but in a linear fashion. Figure 5.2 confirms that as injection rate increases, the percentage of packets dropped at injection increases linearly. Bit Complement (BitComp) traffic poses the worst case scenario for the Runahead network, where almost all packets need to travel diagonally in all directions. This creates more routing conflicts and causes the arrival rate to drop.

As injection rate grows, we experience higher drop rates. Packets can be dropped in one of three places: injection, turning and ejection. Figure 5.2 shows the percentage of packets dropped at injection

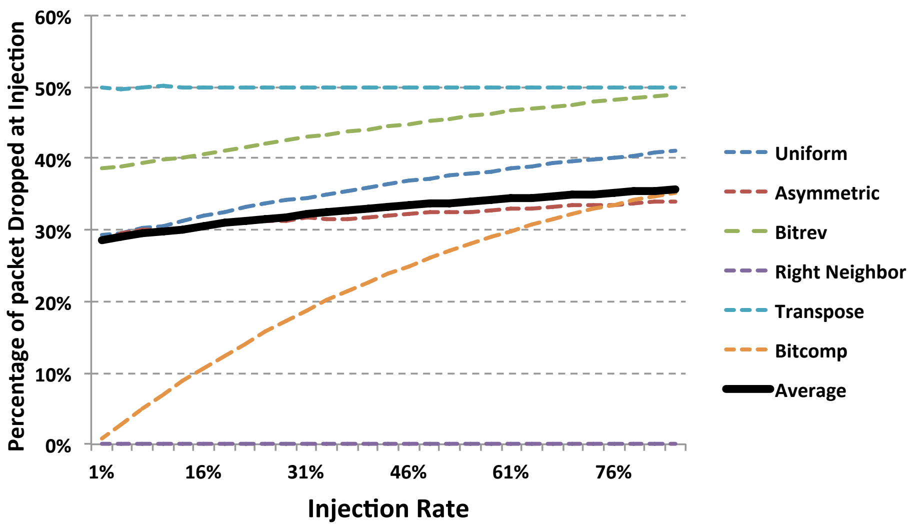


Figure 5.2: Percentage of packets dropped at injection

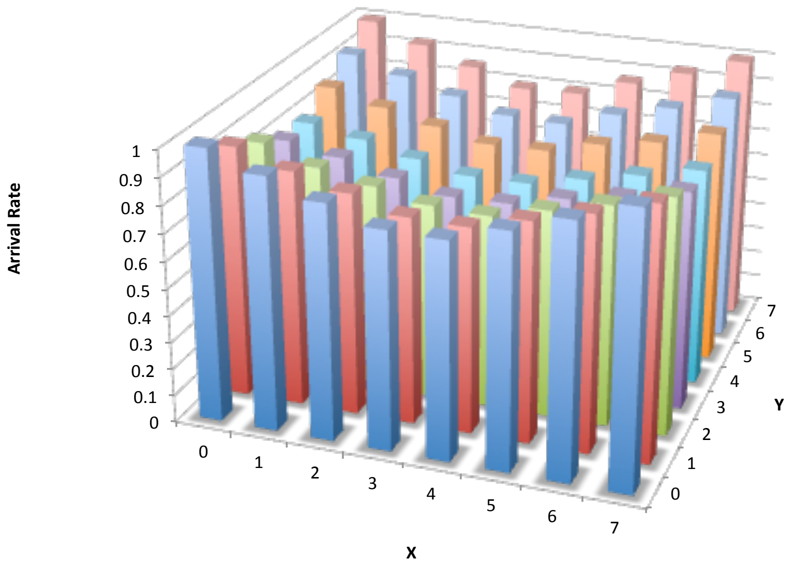


Figure 5.3: Arrival rate by source node for BitComp

with respect to the total number of dropped packets. On average, one in 3 packets is dropped at injection which motivates us to try to inject a packet multiple times as described in Section 3.5.

Although there is potential for unfairness in our port arbitration scheme, we do not see significant evidence of this unfairness in practice. Figure 5.3 shows the source node arrival rate in the worst case scenario (BitComp) for the Runahead network at 7% injection rate. X and Y denote the position of a source node. The graph displays the percentage of delivered packets that originate from a particular source node. Edge nodes experience higher arrival rates than center nodes. This is normal for any 2D

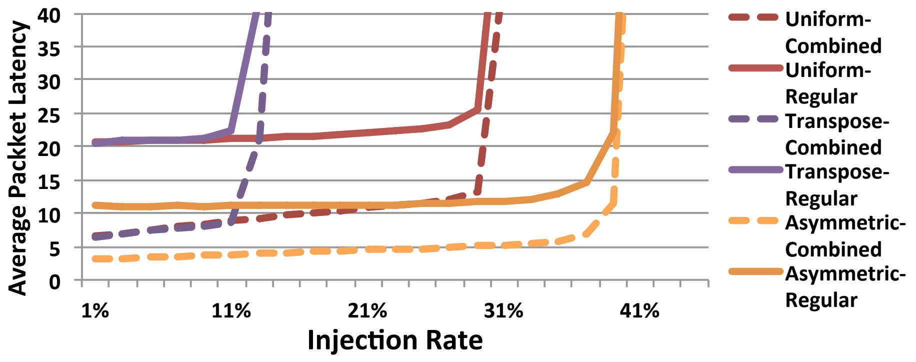


Figure 5.4: Average packet latency for Uniform Random, Transpose and Asymmetric

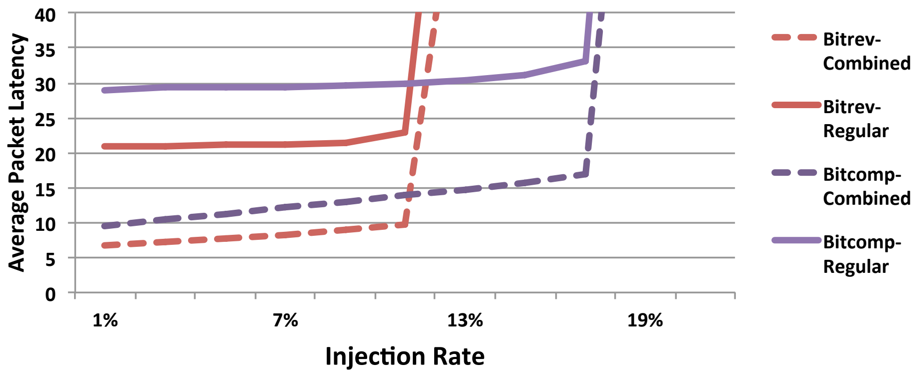


Figure 5.5: Average packet latency for BitRev and BitComp

mesh NoC as the center nodes inherently carry more traffic. There is no significant difference in terms of arrival rate between the east and west side of the network despite the west bias in our implementation (Section 3.3). Note that BitComp represents a worst case traffic pattern for our Runahead network. Although we see some variation across nodes, we do not see particular nodes experiencing starvation or marked unfairness when using the Runahead network.

Given that most applications exhibit low injection rates (∼5% on average) combined with the fact that the Runahead network does not need to handle all traffic lead us to believe that it will be effective in delivering upwards of 90% of packets that are injected into the Runahead network and lead to significant performance improvements.

5.1.2 Latency and Throughput of Combined Network

Figures 5.4 and 5.5 show the average packet latency for the Regular NoC and Combined-NoC on different synthetic traffic patterns. All simulations are done using single-flit packets and all packets are injected into the Runahead network. The Combined-NoC shows a significant decrease in average packet latency compared to the Regular NoC. Note that the packet latency increases faster prior to saturation in the Combined-NoC for several traffic patterns. This is because as the injection rate increases, the arrival

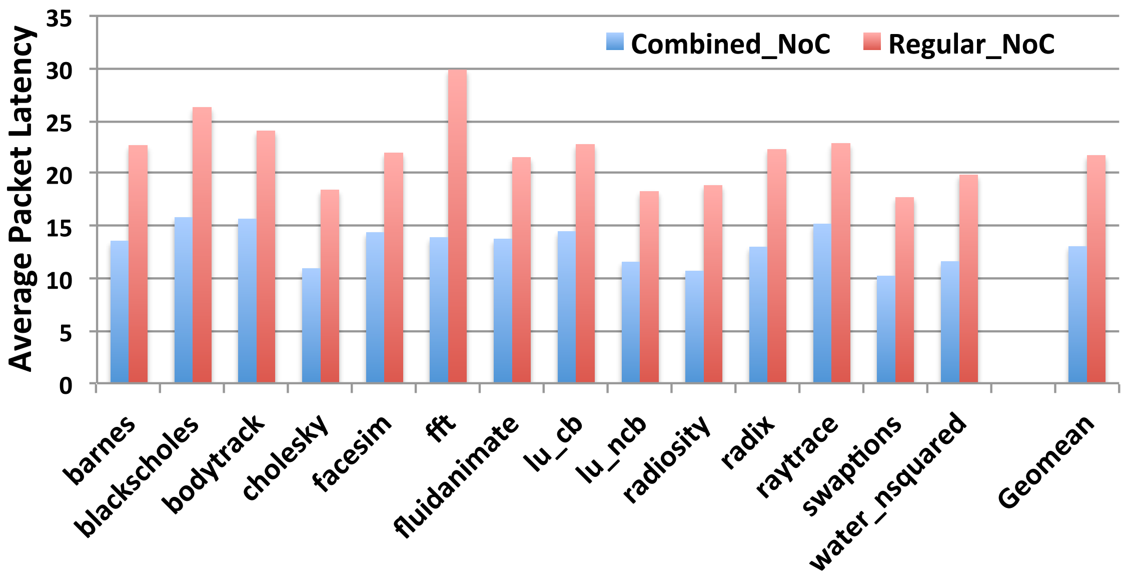


Figure 5.6: Average packet latency in case of accelerator

rate decreases leading to lower effectiveness of the Runahead network; more packets rely on the Regular network for delivery.

Both NoC setups saturate around the same injection rate for most traffic patterns. This is because our Runahead network's injection ports are connected to the same injection queue as the Regular router. If congestion occurs at injection in the Regular network, the Runahead network does not provide any benefit. However, in other traffic patterns such as Bit Reverse where congestion occurs within the network rather than at the injection ports, the Combined-NoC saturates at a higher injection rate.

5.2 SynFull Results

In this section, we compare the four different NoC setups (Section 4) for two different use cases: the Runahead network as an accelerator and as a power-saver. We simulate 14 different applications in SynFull with modified versions of Booksim.

As an Accelerator. In the case of a NoC accelerator, the Runahead network is added on top of a lossless NoC. Figure 5.6 compares the average packet latency between the combined network, and a single Regular network. On average, the combined network achieves 1.66× faster packet delivery in terms of average packet latency.

As a Power-Saver. Figure 5.7 compares the average packet latency between the two different multi-NoC setups and the Combined NoC. The packet latency is normalized to Multi-NoC_Random. Multi-NoC_Random where traffic is shared evenly between the two networks generally out performs Multi-NoC_Select where data and control packets are separated. This is due to better load balancing

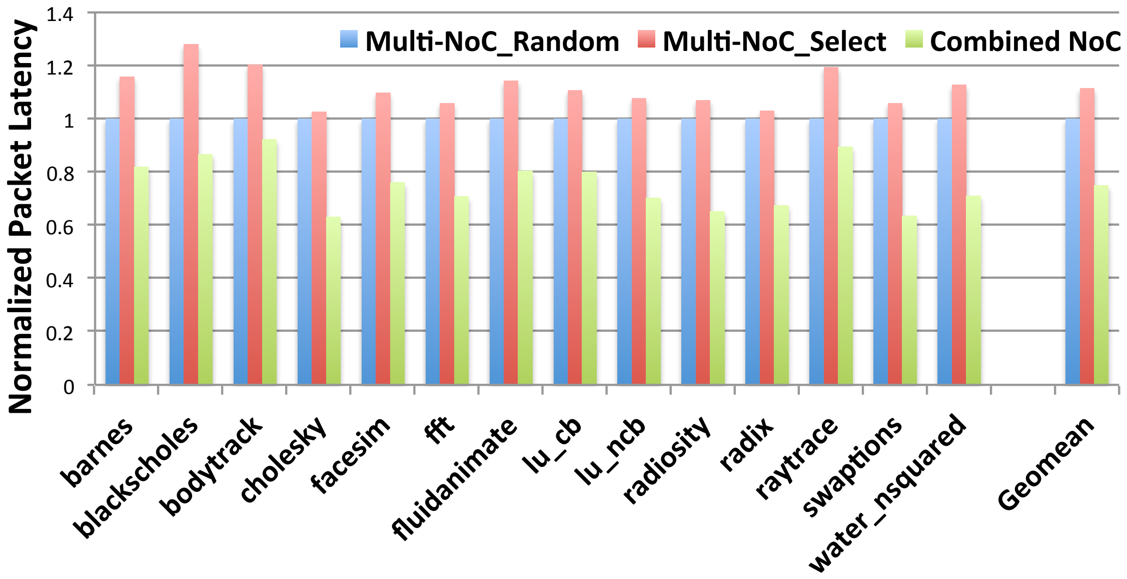


Figure 5.7: Normalized average packet latency in case of power saver

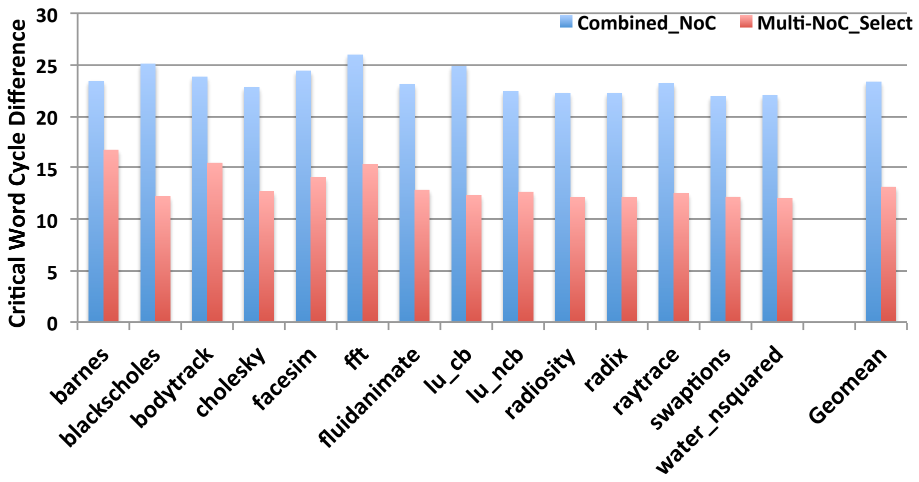


Figure 5.8: Average time difference between arrival of critical word and corresponding cache block

between the two networks. However, the Combined NoC performs the best across all benchmarks. On average, the Combined NoC delivers packets 1.33× faster compared to Multi-NoC_Random, and 1.49× faster compared to Multi-NoC_Select.

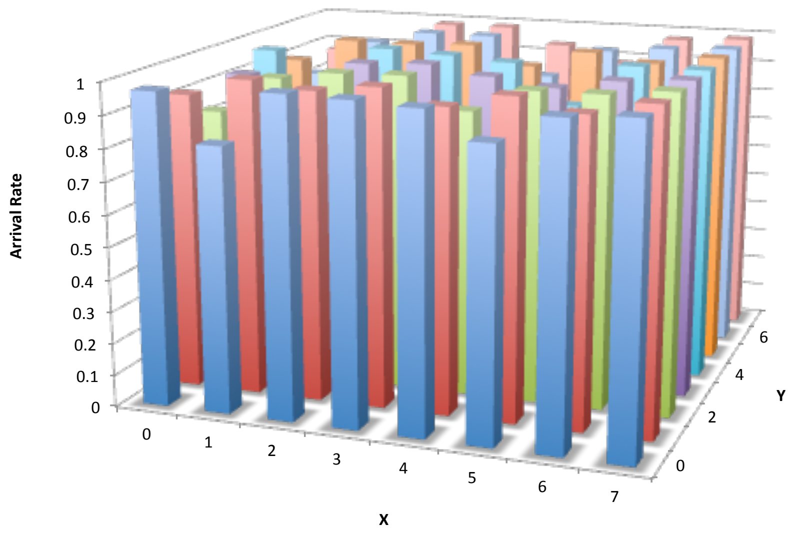
To further investigate the benefits of the Runahead network, we look at the cycle count between the time a critical word arrives and the time when the rest of the data block arrives in Multi-NoC_Select and Combined-NoC in Figure 5.8. The Runahead network on average delivers critical words over 23 cycles faster than the rest of the cache block.

The arrival rate of packets in the Runahead network for each application is listed in Table 5.1. The average arrival rate across all applications is over 97% and the average hop count is 3.7. This means that the Runahead network delivers almost all of the single-flit packets that travel through it, with an average latency of 3.7 cycles. In comparison, the lowest latency that can be achieved in the lossless regular network, which has a 3-stage pipeline router plus link traversal, is 14.8 cycles ($3.7 \times (3+1)$). This zero-load latency is rarely seen in real applications due to congestion with other packets. As the majority of packets in the network are single-flit, the use of the Runahead network enables average packet latency to drop significantly.

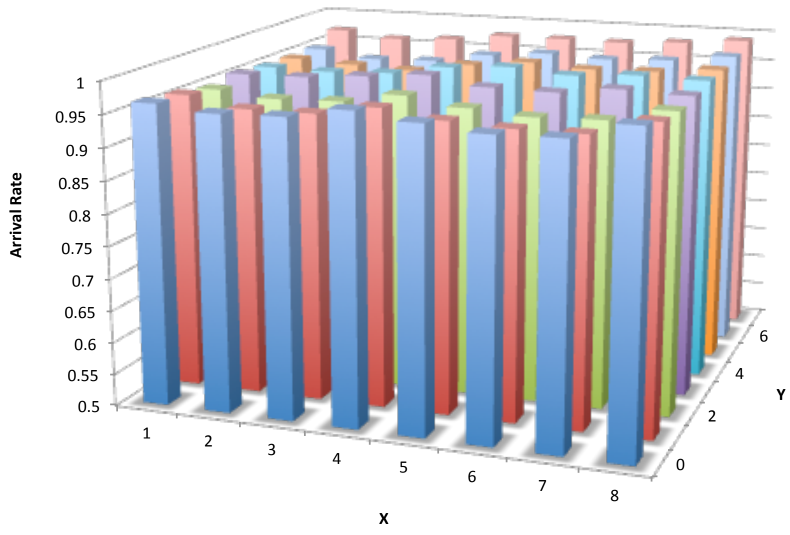
Application	Packet Arrival Rate	Hop Count
barnes	96.29%	3.44905
blackscholes	95.63%	3.61983
bodytrack	95.78%	3.41479
cholesky	98.43%	3.77890
facesim	97.66%	4.10944
fft	97.12%	4.43595
fluidanimate	96.94%	3.57521
lu_cb	97.93%	3.98956
lu_ncb	98.17%	3.56591
radiosity	97.54%	3.52332
radix	97.33%	4.41145
raytrace	96.51%	3.48362
swaptions	98.69%	3.51006
water_nsquared	97.18%	3.45227
Average	97.23%	3.7371

Table 5.1: Packet arrival rate and hop count

To investigate the fairness of our port arbitration scheme, we show the arrival rate per source node for blackscholes (Figure 5.9a) and fft (Figure 5.9b). Blackscholes exhibits the lowest arrival rate while fft exhibits the highest hop count, as shown in Table 5.1. Lower average arrival rate maybe an indication that some nodes experience starvation especially when applications have light traffic. On the other hand, as the distances traveled by packets increase, packets stay in the network longer and have higher chances to cause contention in the Runahead network. For fft, the source node arrival rate is very even across all nodes. We do not see any particular node suffers from low arrival rate. On the other hand, blackscholes exhibits low arrival rate for some nodes due to small variations in packet destinations. Packets are destined to only a few nodes, creating congestion around these nodes which causes the Runahead network to drop more packets. However, the difference in arrival rates is modest leading us to believe that the unfairness of our arbitration scheme does not have a negative impact on application performance.



(a) Blackscholes source node arrival rate



(b) FFT source node arrival rate

Figure 5.9: Source node arrival rate for SynFull simulations

5.3 Full-System Simulation Results

In this section, we evaluate our Runahead network in real systems. We also make some comparisons to other existing works which also aim at reducing network latency and increasing performance. We first present our evaluation for the Runahead network in comparison to a regular network. Next, we present our results in comparison with Aergia, a prioritization scheme, and DejaVu switching, a multi-plane NoC design.

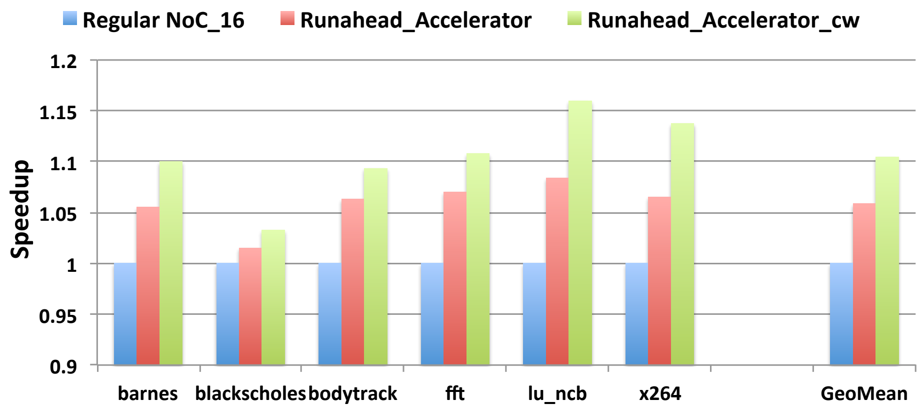


Figure 5.10: Runtime speedup normalized to baseline network

5.3.1 Baseline Comparison

As an Accelerator

We evaluated the performance benefits of the Runahead network when it is used as an accelerator. This means that the channel bandwidth would increase by 10 bytes which is required by the addition of the Runahead network. In Figure 5.10 we compared the system performance for selected workloads in PARSEC and SPLASH-2 using different network setups. As shown in Figure 5.10, the Runahead network accelerated the performance by 5.6% on average without critical word forwarding (Runahead_Accelerator). The critical word forwarding optimization (Runahead_Accelerator_cw) further accelerated the performance by 5% to achieve 10.4% speedup compared the regular network. We have more channel bandwidth in the Runahead network compared to the regular network, so it is expected there would be performance increases.

As a Power-saver

We evaluated the performance benefits of the Runahead network when it is used as a power saver. The execution time speedup is shown in Figure 5.11. Although the channel width of the lossless network in the Runahead network is reduced, we still see speedups compared to the regular network. On average, the Runahead network delivered 0.5% speedup when it only forwards single-flit packets (Runahead_PowerSaver) and 4.4% with critical word forwarding optimization (Runahead_PowerSaver_cw). Applications like fft and lu_ncb which tend to inject more data suffer the most and experienced slowdowns when critical word forwarding is disabled. The processors need to stall and wait until the data packets arrive via the lossless NoC in the Runahead network which has much higher latency compared to the regular network with wider channels. The critical word forwarding optimization enabled the Runahead network to decrease processor stall time and maintain or accelerate performance of these applications while using less power. As the channel width becomes narrower, the routers in the regular network

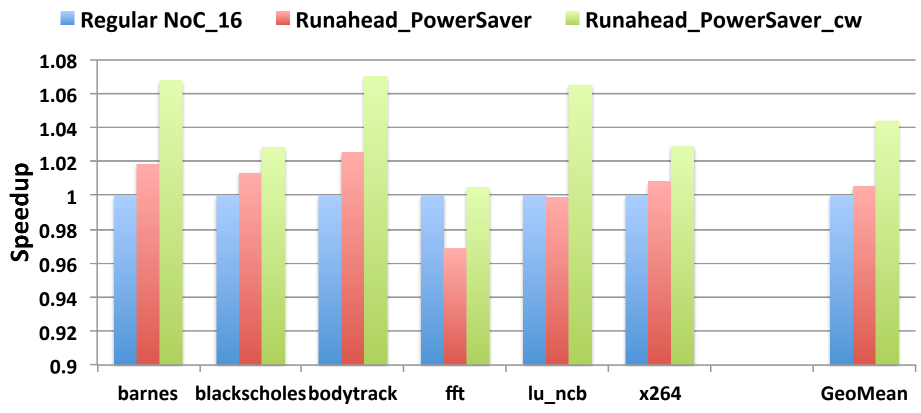


Figure 5.11: Runtime speedup normalized to baseline network

need more time to forward data packets which may cause congestion at injection ports. The Runahead network's benefits are limited by how fast the routers in the regular network can process the injection queue as discussed previously in Section 5.1.2. This causes the performance of the Runahead network in power saver mode to be lower than the performance seen when it is used as a accelerator.

Discussion

The benefits of the Runahead network are not uniformly equal for all benchmarks. Benchmarks with very light network traffic (blackscholes) only achieved small performance increase. Benchmarks that have more network activity benefited more from the Runahead network. To illustrate the network activity in the Runahead network, we show the percentage of flits that are injected into the Runahead network in Figure 5.12. These results were obtained when the Runahead network is running in power saver mode and with the critical word forwarding optimization disabled. Critical word packets are single-flit duplicated packets that travel in the Runahead network. In order to keep the percentage of single-flit packets accurate, we do not include these packets when collecting the results. On average, only 22.9% of all flits that travelled through the network were injected into the Runahead network. Note that we are calculating the percentage with respect to the total number of flits that went through the network. This may skew the results as the data packets are 9 flits long in power saver mode. In our experiments, we confirmed that on average, over 72% of the packets in the network are single-flit packets. The percentages for individual applications are listed in Table 5.2. We noticed that applications with high percentage of single-flit packets benefit more from the Runahead Network in terms of performance. This indicates that the Runahead network has the opportunity to speed up the majority of the network messages despite the fact that it only carries 22.9% of the flit traffic.

The arrival rate and hop count for packets that traveled through Runahead network for individual benchmarks are listed in Table 5.3. All applications have over 95% arrival rate. This shows the Runahead network is capable of delivering almost all of the packets that traveled through it in real systems. Note

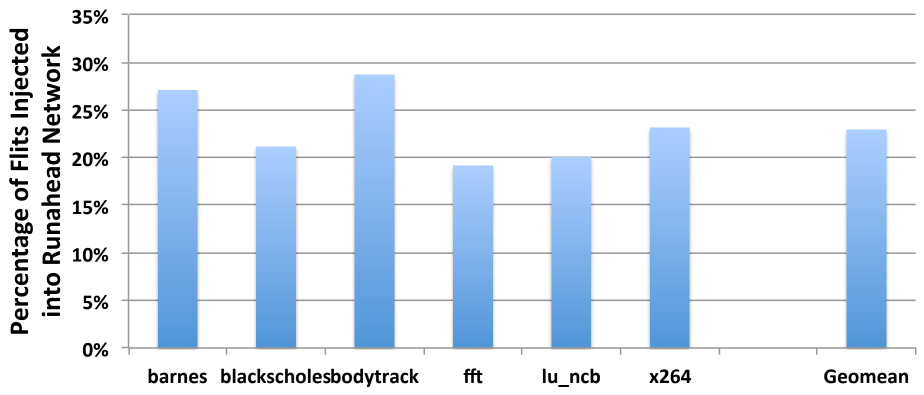


Figure 5.12: Percentage of flit injected into Runahead

Application	Percentage of Single-Flit-Packet
barnes	76.98%
blackscholes	70.70%
bodytrack	78.37%
fft	68.06%
lu_ncb	69.27%
x264	73.01%
Average	72.64%

Table 5.2: Percentage of Single-Flit-Packets

that fft and *lu_ncb* have lower arrival rates compare to other benchmarks. This suggests that there is more network contention in these two benchmarks. Since the Runahead network has 1 cycle per hop delay, the average latency for packets that traveled through the Runahead network is the same as the average hop count which is 3.42 cycles.

5.3.2 Comparison with Aergia prioritization scheme

The Runahead network speeds up the delivery of single-flit packets. To keep the design simple, the Runahead network only distinguishes packets based on their size. However, there are other works such as Aergia [19] that prioritizes packets according to slack. In our implementation of Aergia, we assumed a

Application	Packet Arrival Rate	Hop Count
barnes	97.74%	3.54025
blackscholes	99.32%	2.92945
bodytrack	97.02%	3.51733
fft	95.60%	3.68849
lu_ncb	96.11%	3.30842
x264	98.00%	3.60125
Average	97.29%	3.4210

Table 5.3: Packet arrival rate and hop count

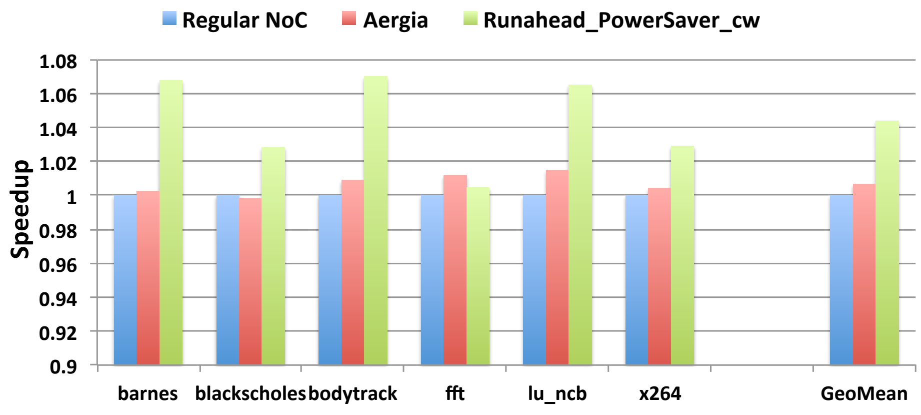


Figure 5.13: Performance comparison between Aergia and Runahead

perfect L2 miss predictor. This means we know exactly which packet will cause a L2 miss and calculate the slack more accurately.

Figure 5.13, shows the performance of the regular network with round-robin scheduling, Aergia and Runahead network in power saver mode. All three setups have the same channel width for data transmission to keep the comparison fair. Aergia has very little impact with PARSEC and SPLASH-2 benchmarks even though we used an unrealistic perfect L2-miss predictor. The reasons are twofold. First, most PARSEC and SPLASH-2 benchmarks do not have high L2 miss rates which is one of the key factors in computing priority in Aergia (Aergia was originally proposed and evaluated using multiprogrammed SPEC workloads). Second, the benchmarks have very little congestion and the network is lightly loaded. This is one of our key motivations that average channel utilization is low. Because of this, prioritization schemes are unlikely to find opportunities to accelerate packets in the absence of congestion. Applications with relatively high network contention like fft and lu_ncb benefited more from Aergia prioritization scheme relative to other applications. On the other hand, Runahead network is able to speedup the performance by 4.4%. The proposed network works well even with lack of congestion. Also, the proposed network does not require complex predictors to calculate priorities of packets before they are injected into the network. This also saves power and energy.

5.3.3 Comparison with DejaVu multi-plane NoC

DejaVu switching is proposed as a multi-plane NoC where control and data packet are separated into different planes. When a data packet needs to be injected, a reservation packet is sent out first on the control plane to make reservations for the data packet. The reservations enable the data packets to be forwarded without suffering delays of making routing decisions at every router.

Figure 5.14 shows the performance of the regular network, DejaVu and Runahead network in power saver mode. All three setups have the same channel width for data transmission to keep the comparison

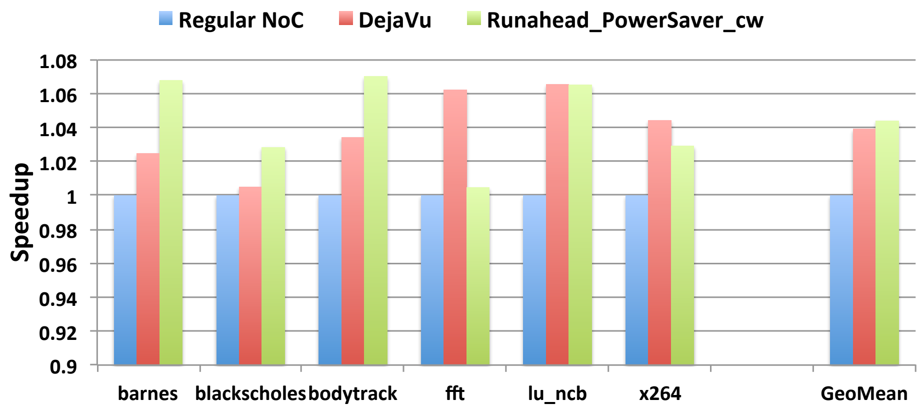


Figure 5.14: Performance comparison between DejaVu and Runahead

fair. DejaVu has a speedup of 3.9% compared to the regular network. This is because data packets travel faster due to the advance reservations at data plane routers. Ideally, data packets would have 1 cycle per hop delay because of the advanced reservations. Due to the separation of data and control packets in DejaVu, both networks are less congested compared to the single lossless NoC used in the Runahead network. This enables DejaVu to deliver data packets faster compared to the Runahead network. Although we forward the critical word in both cases, DejaVu will populate the cache line faster than Runahead network after unblocking the stalled processor with the critical word. This gives DejaVu an advantage in situations where applications access words other than the critical word in the cache line. An instruction that is stalled due to a cache miss for an already requested cache line will stall for less time in the case of DejaVu. This explains the higher performance speedups for fft, *lu_ncb* and $x264$ compared to the Runahead network. These applications tend to have more spatial locality in the same cache line. On the other hand, the Runahead network still delivered 4.4% speedup compared to the regular NoC.

5.4 Power and Area

In this section, we show the power and area consumption of the Runahead network. We first evaluate the 8x8 network power and area usage using DSENT. Next, we evaluate the power and area of a single router using RTL modeling.

5.4.1 Critical Path

DSENT reports a minimum clock period for the Runahead network of 481.073 ps when optimized for a frequency of 2GHz. As a result, we are confident that a Runahead network router can be traversed in a single cycle in a 2GHz system.

Since we only have access to the older TSMC 65nm technology library, the critical path reported by

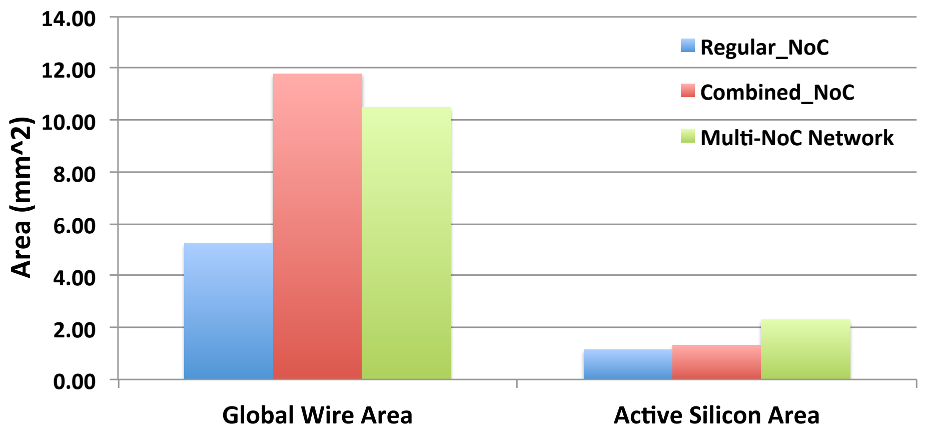


Figure 5.15: Area comparison

Synopsys for a single regular router is 1.5 *ns*. However, the RTL results showed the critical path of the Runahead router did not increase compared to the regular router. This ensures that the combined design can still operate in the original clock speed. To ensure that the Runahead network can forward packets in one cycle, we measure the critical path of the Runahead network alone. Synopsys design compiler reported the critical path for the Runahead network logic alone is 800 *ps* after timing optimizations. The Runahead network can operate almost twice as fast as the regular router. As a result, we are confident that the Runahead network can function correctly in a single router cycle.

5.4.2 Area

DSENT

The Combined NoC incurs low active area overhead compared to the Regular network (i.e., the Runahead network as an accelerator) and achieves significant savings compared to the multi-NoC setups (i.e., the Runahead network as a power-saver), as shown in Figure 5.15. However, the Combined NoC has the highest usage of global wire area. This is due to the wider channels used in the Runahead network to carry additional metadata, as mentioned in Chapter 4.

Table 5.5 shows area that the Runahead network consumes in comparison to the total consumption of the combined network. The Runahead network only accounts for 13.76% of the active silicon area of the Combined NoC. However, because the Runahead network has wider physical channels due to the additional header information, it uses more than half of the global wire area.

RTL Modeling

The details of the RTL area comparison of a single router can be found in Table 5.4. In accelerator mode, there is only 3.2% increase in area usage. The increase in area usage consists of the additional multiplexers and registers that composed the Runahead router. In power saver mode, the Runahead

Use case	Area(um2)	Difference in Area
Regular Network	411332	1
Accelerator	424838	1.032
Power Saver	207901	0.505

Table 5.4: RTL router area comparison between 3 different setups

	Total Usage	Runahead Usage	Percentage
Dynamic Power	0.0237 W	0.00398 W	16.81%
Leakage Power	1.11 W	0.0995 W	8.97%
Total Power	1.13 W	0.103 W	9.13%
Global Wire area	11.8 mm^2	6.55 mm^2	55.56%
Active Silicon Area	1.34 mm^2	0.183 mm^2	13.67%

Table 5.5: Power composition of Combined 8x8 Network

router decreased the area usage by almost 50%. This is due to the narrower channels in the regular router resulting in fewer network resources. These resources includes buffer spaces and crossbar switches. In power saver mode, the sizes for of these components are halved.

5.4.3 Power

DSENT

Leakage power tends to dominate total NoC power consumption. Figure 5.16 shows dynamic and leakage power normalized to the Regular NoC. As shown, the Combined NoC significantly reduces leakage power compared to the multi-NoC designs (i.e., the Runahead network as a power-saver), and incurs only a 10% overhead in leakage power compared to the Regular NoC (i.e., the Runahead network as an accelerator). When measuring dynamic power, we use the average injection rate obtained in our SynFull simulations across all benchmarks. In general, the Combined NoC consumes more dynamic power than the other network setups due to packet duplication in the Runahead network. Unlike in a multi-NoC design where the injection rates of each subnetwork is lower, the Combined NoC has the same injection rate as the Regular NoC, while also incurring additional switching in the Runahead network to carry duplicate latency-sensitive packets. However, in our measurements, dynamic power accounts for only a small portion of total power, ranging from 1.1% for the multi-NoC setups to 2.09% for the Regular NoC and combined networks. As shown in Table 5.5, the Runahead network only accounts for 9.13% of the total network power usage in the combined network. Some of the power consumption for the Runahead network in the combined network comes from dynamic power for reasons discussed previously. Leakage power for the Runahead network is also high due to the wider channels and larger multiplexers to accommodate the additional metedata. Furthermore, the Runahead network accelerates application performance, which naturally leads to additional overall energy savings due to shorter runtimes.

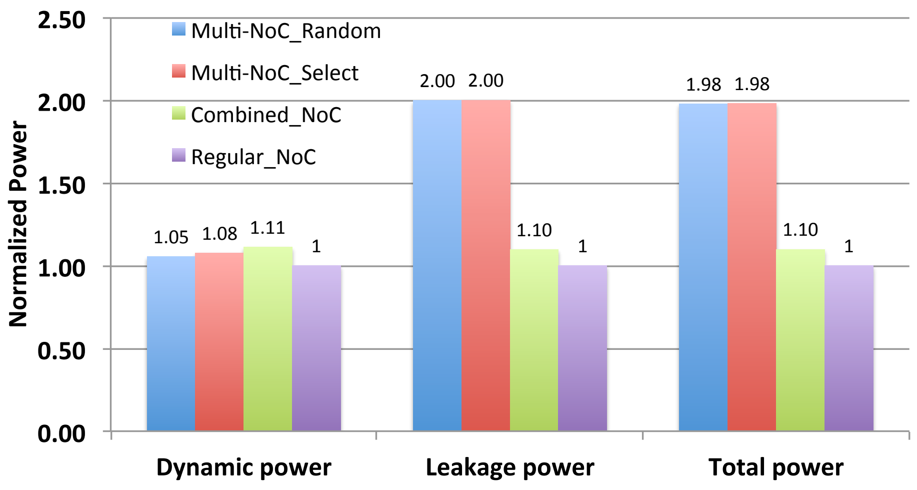


Figure 5.16: Power comparison for 8x8 network

Use case	Power(mW)	Difference in Power
Regular Network	94.7395	1
Accelerator	98.4981	1.040
Power Saver	47.4965	0.513

Table 5.6: RTL router power comparison between 3 different setups

RTL Modeling

Power savings for an individual router in TSMC 65nm are also promising and the results are listed in Table 5.6. The links that connect the routers are not modeled. Compared to the regular router, the proposed router only uses 4% additional power in accelerator mode. However, when in power saver mode, the total power consumption decreased by almost 50% compared to the regular router. This is expected as the sizes for buffers and crossbar switches, which are two major power contributors in the baseline router, are halved in power saver mode.

5.5 Chapter Summary

In this chapter, we evaluated the Runahead network design from various aspects. We first measured arrival rate and source arrival rate using synthetic traffic. These results are obtained by using Runahead network alone without a regular network. Next, we used synthetic traffic to measure the throughput of the network. We saw that the throughput is limited by the baseline router as the Runahead router takes input from the input queue of the baseline router. However, in some cases, the proposed network still managed to improve the throughput for some traffic patterns when the congestion does not occur at injection port. Next, we used SynFull to measure the packet latency in various setups. We found that the Runahead network perform the best even with fewer network resources. We then evaluated the performance of the RunAhead network in real system conditions. With the applications we have studied, prioritization schemes are not well suited as the network is lightly loaded. However, the Runahead network still performed well under these conditions. Finally, we reported the power and area of the Runahead network. The Runahead network can reduce power and area usage significantly in power saver mode, and has very little overhead when used as an accelerator.

Chapter 6

Related Work

In this chapter, we briefly introduce and explore research that is relevant to the Runahead network. As the Runahead network aims at reducing power and increasing system performance, there are many relevant research works. We touch on related topics in areas of multi-NoC designs, prioritization schemes on NoCs, power efficient NoCs, low-latency NoCs, best-effort NoCs and critical word optimizations.

6.1 Multi-NoC Designs

Employing multiple NoCs can improve performance while simultaneously improving area and power efficiency [4]. Flit-reservation flow control [51] uses a separate network for reservation messages; these messages reserve buffers and channels for the exact time a data packet will use them. Doing so speeds up message handling and improves performance.

DejaVu switching [1] proposes a two-network design: one network for control and coherence packets and one for data packets. We compared our design with a design similar to DejaVu and found that the Runahead network out performs DejaVu with fewer network resources.

Flores et al. [23] propose two networks for critical and non-critical traffic. They use heterogeneous networks composed of low-latency wires for critical messages and low-energy wires for non-critical ones. Mishra et al. [44] proposed an heterogeneous multi-NoC system where one network has low latency routers, and the other has high bandwidth channels. In their approach, they also added a prioritization scheme where applications that benefited more from the customized network have higher priority in the network. NoCNoC [53] proposes a two network design that separates critical words from non-critical ones in a cache line. NoCNoC is able to save power by dynamically scaling voltage and frequency of the non-critical network with minimal performance impacts.

Multiple networks also provide opportunities for traffic partitioning [57] and load balancing [4]. Catnap [18] is an energy-proportional multi-NoC design; rather than separating the types of traffic sent to each NoC, networks are turned on and off to respond to changes in network load. Enright Jerger et al. [20] propose a hybrid NoC design where a separate NoC exists on a silicon interposer. In their design, the interposer NoC carries memory traffic, while the NoC on the chip carries the rest of the traffic. Exploiting the otherwise wasted routing resources of the silicon interposer leads to better performance. Most of these multi-NoC designs require VC routers with buffers in all of the networks to function. The Runahead network on the other hand, only requires the addition of simple logics with minimal overheads.

6.2 Prioritization Schemes on Network on Chip

Traditional Network-on-Chips employ simple arbitration strategies for packets. These strategies include round-robin or age-based arbitration. However, not all packets that travel through the network are the same. Some of them are critical packets that cause processor to stall, and some are non-critical such as write-back requests. Bolotin et al. [9] proposed prioritizing control packets over data packets in the NoC. They saw substantial performance improvement when small control packets are prioritized over data packets. Globally Synchronized Frames (GSF) [40] is proposed as a local arbitration, QoS-oriented prioritization scheme. GSF provides prioritization mechanisms within the network to ensure each application receives an equal amount of network resources. The objective of GSF is aimed to provide network-level QoS. This means that GSF will penalize applications that use more network resources.

Application-Aware Prioritization Mechanism (STC) [17] is proposed as a prioritization scheme to accelerate network-sensitive applications. STC ranks applications at regular intervals based on their network intensity. Packets from applications with lower intensity would have higher priority over packets from high intensity applications.

Aergia [52] uses the notion of slack to prioritize packets. Aergia may increase network throughput if network is congested. However, from our previous evaluations, we see little performance impact. This is because the network is lightly loaded and the L2 miss rates are low for the applications studied.

Prioritization schemes can only show their full potential when the network carries heavy traffic, but fall short when the network is lightly loaded. On the other hand, the Runahead network performs well with lack of contention.

6.3 Power-Efficient NoC Designs

Decreasing buffering space has always been one of the top priorities when designing a power-efficient NoC. As discussed in Section 2.1.1, wormhole [15] was proposed to reduce the buffer requirement for the virtual-cut-through flow control [34]. Also, bufferless networks have received significant research attention [21, 28, 43, 45] because of power and energy concerns. Buffers consume 1/3 of the overall power of a NoC, thus, eliminating buffers from NoC is a reasonable action to save power. Bufferless NoCs are great at saving energy and power but at a performance cost. In the case of BLESS [45] and CHIPPER [21], packets are deflected until they reach their destination. In SCARAB [28] and Millberg et al. [43], packets are dropped upon contention and a retransmission message is issued to the source. Our Runahead network does not react to dropped packets and does not deflect packets in the case of contention. This keeps the design of the Runahead network routers simple, and gives packets deterministic network latencies.

MinBD is proposed [22] to increase the performance of CHIPPER by adding intermediate buffers to minimize deflection. MinBD operates similar to CHIPPER, it will deflect packets in case of contention. However, MinBD buffers packets that meet certain conditions to minimize deflection in the network. This increases the performance compared to the bufferless CHIPPER design. However, adding intermediate buffers degrades the power benefits of the MinDB.

A prediction based flow control [47] is proposed to predict buffer usage and throttles injection when buffer spaces are not available. This ensures better utilization of buffer resources and delivers better performance compared to traditional flow control techniques. Clumsy Flow Control [36] also throttles injection source to control the total number of packets in the network for deflection based bufferless networks. CFC reduces deflection of packets in the bufferless network which resulted in significant power savings.

Adaptive Flow Control (AFC) [?] is proposed to adaptively switch between buffered and bufferless network. During low traffic loads, the router will deflect packets similar to BLESS without utilizing any buffer resources. During high traffic load, the router acts like a conventional NoC router. This adaptivity allows AFC to have both the power savings of bufferless network, and the performance of a buffered network. Unlike the Runahead network, AFC requires additional hardware to support the switching between buffered and bufferless mode.

There are also some existing works that save power by scaling the operating voltage and/or clock frequency. Voltage-Frequency Island Partitioning (VFI) [48] is proposed to dynamically scale voltage and frequency for individual routers depending on their utilizations. Zhan et al. [61] proposed to dynamically scale voltage and frequency depending on the packet's slack similar to Aergia. These approaches showed some promising power saving abilities, but the feasibility of these approaches is unclear as the per-node voltage and frequency scaling have non-negligible area overhead and cause additional timing issues. DimNoC [60] is a more practical design that uses specialized ”dim silicon” memory technologies with new buffer architectures to better cope with voltage and frequency scaling. DimNoC also showed significant power and area savings.

These power efficient NoCs require additional hardware or specialized technology to achieve the power savings they are promising. The Runahead network can save power with conventional hardware and uses existing channel resources of the regular network.

6.4 Low-Latency NoC Designs

One of the goals of our network design is to accelerate packet transmission. Similarly, there has been significant research on low-latency NoCs to improve performance. Lookahead routing is a common technique to reduce the number of pipeline stages in the router [24]. In the lookahead routing, the routes are computed one router in advance. This routing information is stored at the head flit. When the head flit arrives at the next router, it can initiate allocation and arbitration requests to the precomputed output port while simultaneously determining the output port in the next router.

A speculative switch allocator is proposed to remove the dependency between switch and VC allocation [50]. A head flit can send both switch and VC allocation requests speculatively at the same time. If both requests are granted, the flit can pass through 2 pipeline stages in one cycle. However, if VC allocation is denied, the switch allocation request is ignored. Ignoring requests may result in under utilization of the switch resources, thus, the speculative requests have lower priority then non-speculative requests that are sent by flits with granted VC allocation requests. This speculative allocation technique performs well for light traffic, but the performance benefits degrade when network is heavily loaded as the number of failed speculative requests increases.

Express virtual channels (EVC) [39] reduce latency by allowing packets to bypass intermediate routers. The express virtual channels share the same physical link with other virtual channels within the network. However, EVCs virtually link routers that are not immediately adjacent to each other. Packets in EVCs can bypass pipeline stages in a non-speculative fashion at intermediate routers. In EVC, packets that are travelling long distances have higher priority than local packets that only need few hops to reach destination. EVCs result in reduction of the overall network latency at a cost of additional area and have no effect on packets that travel short distances.

Dynamic Priority-Based Fast Path NoC router [49] is proposed to bypass switch allocation stage for flits (Fast Flit) that frequently travels the same path. The Fast Path NoC dynamically detects frequent communication patterns in the network using a Path Frequency Analyzer (PFA) and adjusts prioritizations during SA. These fast flits bypass the SA pipeline stage by sending SA requests one cycle in advance. The fast flits can enter crossbar directly if the SA requests are successful. Fast flits will act like normal flits and cannot bypass any stage if the advance SA requests are denied. Our proposed network does not require additional logic that distinguishes packets and does not dynamically adapt to communication patterns.

Recently, SMART [?] is proposed to reduce overall communication latency by allowing packets to travel multiple hops in a single cycle. They observed that the wire delay is much shorter then a typical router cycle. SMART uses asynchronous low-swing repeater circuits and voltage lock repeaters (VLR) in the links to allow packets to travel multiple hops within a single router cycle. The links in SMART require specialized repeaters to enable multi-hop traversal, but the Runahead network can use conventional links that already exist in the regular network.

A low-cost router design [37] reduces latency using a simple ring-stop inspired router architecture for fast traversal of packets travelling in one direction; packets changing direction pay additional latency when they are buffered. A non-speculative single-cycle router pipeline improves performance by allocating the switch in advance of the message arrival [38]. There are also designs that use packet route predictions to speed up the network [30, 42]. Often these low-latency designs increase hardware complexity, energy and area in order to achieve better performance. Our simple Runahead network achieves performance improvements with minimal power and area overhead.

6.5 Best-effort NoCs and QoS

Our Runahead router makes a best effort to deliver packets; however, in the face of contention, packets may be dropped. Other work explores best effort packet delivery in terms of QoS. The proposed works aim to mitigate the impact of congestion and bound message latency for certain traffic. Brand et al. [56] separate traffic into guaranteed traffic and best effort traffic. When congestion is detected in the NoC, best effort traffic will get throttled to provide faster delivery of the guaranteed traffic. On the other hand, when the Runahead network experiences congestion, packets are dropped as their arrival is not essential. Avasare et al. [2] present a communication management scheme that guarantees delivery of control packets and delays data packets if there is congestion. They show that it is possible for a best-effort NoC to provide sufficient communication guarantees with respect to the application requirements.

6.6 Critical Word Optimizations

Delivering the critical word as soon as possible can improve application performance. Separating critical data in either main memory [12] or the caches [33] can efficiently deliver critical data faster. In these 2 approaches, critical words are saved in a separate region in memory or cache with fast response time. NoC designs to prioritize packets based on latency-sensitivity [9,13] have been proposed. These networks differ from our Runahead network as they target specific packets and generally require more complex hardware. Our Runahead network is a general-purpose NoC that can be added to any existing NoC design. It also does not rely on any additional information from the processors.

Chapter 7

Conclusion

7.1 Thesis Summary

Increasing network performance and decreasing energy are the two most important considerations when designing new NoC systems. To cope with future CMP's communication demands, we propose the Runahead network, a lightweight, bufferless, lossy NoC that co-exists with a conventional network. It can serve as either a *power-saver* for more efficient use of the network resources, or as an *accelerator* that provides lightweight, low latency communication on top of a conventional NoC. The Runahead NoC is designed to provide single-cycle hops across the network. To accomplish this, the network is lossy in nature, dropping packets when contention occurs. We present the design of the Runahead NoC router architecture that combines route computation and port arbitration with link traversal.

From experiments with SynFull workloads, we find that, on average, the Runahead network can maintain over 95% packet arrival rate. As an accelerator, the Runahead network reduces average packet latency by 1.66× with only 10% overhead in terms of network power. As a power-saver, the Runahead network achieves 1.73× saving in network active area and 1.80× savings in network power.

Full-system simulation shows that the Runahead can provide 1.10× and 1.04× performance improvement on average in accelerator and power saver mode respectively. The benefit of forwarding critical words is also significant as it shortens processors' stall time. The Runahead network preforms well in PARSEC and SPLASH-2 benchmarks when the network is lightly loaded. We also show that in these system conditions, prioritization schemes may not show their full potential due to lack of contention. Compared to DejaVu switching, a multi-plane NoC approach, Runahead network still delivers better performance because the Runahead network has the ability of both accelerate single-flit control packets and forward critical words faster.

7.2 Future Directions

The design of our current Runahead network focused on mesh topology with simple X-Y dimension ordering routing algorithm. One direction is to extend the Runahead network to other topologies or routing algorithms. This will give Runahead network a broader usage and can be applied to more diverse CMP systems.

Runahead network currently uses hard-coded port-arbitration scheme, if area and power constraints allow, an adaptive port-arbitration scheme can be used to better suit the network communication pattern. Although we did not find significant unfairness with our current applications, starvation may still occur when the network is heavily loaded with more packets getting dropped in the Runahead network. An adaptive port-arbitration scheme may improve the fairness of the current Runahead network design. However, this requires additional hardware to detect communication patterns and to change arbitration dynamically.

A conventional network is needed to ensure the correctness of the network as the Runahead network does not react to any dropped packets. It just assumes the regular network will deliver these dropped packets to the destination. One interesting direction is to let Runahead network handle all network traffic. If a packet gets dropped, the Runahead network would then inject it into the regular network. This prevents the energy overhead associated with the duplicated injection. However, this may introduce more area and power overheads for the routers in the network to enable such flow control technique.

The best effort lossy delivery concept in network-on-chip is relatively new. In the past, only a few designs considered dropping packets as an option. However, in this thesis, we have shown that it is feasible to have lossy network on chips. This can open a new area of research where systems or flow control techniques can operate with a lossy communication fabric.

Bibliography

[1] A. Abousamra, R. Melhem, and A. Jones, “Deja Vu switching for multiplane NoCs,” in *International Symposium on Networks on Chip*, May 2012, pp. 11–18.

[2] P. Avasare, V. Nollet, J.-Y. Mignolet, D. Verkest, and H. Corporaal, “Centralized end-to-end flow control in a best-effort network-on-chip,” in *Proceedings of the 5th ACM International Conference on Embedded Software*, ser. EMSOFT ’05, 2005, pp. 17–20.

[3] M. Badr and N. Enright Jerger, “SynFull: Synthetic traffic models capturing cache coherent behaviour,” in *Proceedings of the International Symposium on Computer Architecture*, 2014.

[4] J. Balfour and W. J. Dally, “Design tradeoffs for tiled CMP on-chip networks,” in *Proceedings of the 20th Annual International Conference on Supercomputing*, ser. ICS ’06, 2006, pp. 187–198.

[5] N. Barrow-Williams, C. Fensch, and S. Moore, “A communication characterisation of SPLASH-2 and PARSEC,” in *IEEE International Symposium on Workload Characterization*, Oct 2009, pp. 86–97.

[6] D. U. Becker, “Efficient microarchitecture for network-on-chip routers,” Ph.D. dissertation, Stanford University, 2012.

[7] C. Bienia, S. Kumar, J. P. Singh, and K. Li, “The PARSEC benchmark suite: Characterization and architectural implications,” in *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques*, October 2008.

[8] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti *et al.*, “The gem5 simulator,” *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, 2011.

[9] E. Bolotin, Z. Guz, I. Cidon, R. Ginosar, and A. Kolodny, “The power of priority: NoC based distributed cache coherency,” in *Networks-on-Chip, 2007. NOCS 2007. First International Symposium on*, May 2007, pp. 117–126.

[10] S. Y. Borkar, "Future of interconnect fabric: a contrarian view," in *Proc. Int. Workshop on System Level Interconnect Prediction*, 2010.

[11] S. Borkar, "Thousand core chips: A technology perspective," in *Proceedings of the 44th Annual Design Automation Conference*, ser. DAC '07, 2007, pp. 746–749.

[12] N. Chatterjee, M. Shevgoor, R. Balasubramonian, A. Davis, Z. Fang, R. Illikkal, and R. Iyer, "Leveraging heterogeneity in dram main memories to accelerate critical word access," in *International Symposium on Microarchitecture*, Dec 2012, pp. 13–24.

[13] W. Dai, H. An, Q. Li, G. Li, B. Deng, S. Wu, X. Li, and Y. Liu, "A priority-aware NoC to reduce squashes in thread level speculation for chip multiprocessors," in *International Symposium on Parallel and Distributed Processing with Applications.* IEEE, 2011, pp. 87–92.

[14] W. J. Dally, "Virtual-channel flow control," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 3, no. 2, pp. 194–205, 1992.

[15] W. J. Dally and C. L. Seitz, "The torus routing chip," *Distributed computing*, vol. 1, no. 4, pp. 187–196, 1986.

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

[17] R. Das, O. Mutlu, T. Moscibroda, and C. R. Das, "Application-aware prioritization mechanisms for on-chip networks," in *International Symposium on Microarchitecture.* IEEE, 2009, pp. 280–291.

[18] R. Das, S. Narayanasamy, S. K. Satpathy, and R. G. Dreslinski, "Catnap: Energy proportional multiple network-on-chip," in *Proceedings of the 40th Annual International Symposium on Computer Architecture*, ser. ISCA '13, 2013, pp. 320–331.

[19] Das, Reetuparna and Mutlu, Onur and Moscibroda, Thomas and Das, Chita R, "Aergia: A Network-on-Chip Exploiting Packet Latency Slack," *Micro, IEEE*, vol. 31, no. 1, pp. 29–41, 2011.

[20] N. Enright Jerger, A. Kannan, Z. Li, and G. Loh, "NoC architectures for silicon interposer systems," in *Proceedings of the International Symposium on Microarchitecture*, 2014.

[21] C. Fallin, C. Craik, and O. Mutlu, "CHIPPER: A low-complexity bufferless deflection router," in *Proceedings of the 2011 IEEE 17th International Symposium on High Performance Computer Architecture*, ser. HPCA '11, 2011, pp. 144–155.

[22] C. Fallin, G. Nazario, X. Yu, K. Chang, R. Ausavarungnirun, and O. Mutlu, "MinBD: A minimally-buffered deflection router approaching conventional buffered-router performance," 2011.

[23] A. Flores, J. Aragon, and M. Acacio, "Heterogeneous interconnects for energy-efficient message management in CMPs," *Computers, IEEE Transactions on*, vol. 59, no. 1, pp. 16–28, Jan 2010.

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

[25] C. Gómez, M. E. Gomez, P. López, and J. Duato, “An efficient switching technique for NoCs with reduced buffer requirements,” in *Parallel and Distributed Systems, 2008. ICPADS’08. 14th IEEE International Conference on.* IEEE, 2008, pp. 713–720.

[26] P. Gratz and S. W. Keckler, “Realistic workload characterization and analysis for networks-on-chip design,” in *The 4th Workshop on Chip Multiprocessor Memory Systems and Interconnects (CMP-MSI)*, 2010, pp. 1–10.

[27] P. Gratz, C. Kim, K. Sankaralingam, H. Hanson, P. Shivakumar, S. W. Keckler, and D. Burger, “On-chip interconnection networks of the TRIPS chip,” *Micro, IEEE*, vol. 27, no. 5, pp. 41–50, 2007.

[28] M. Hayenga, N. Enright Jerger, and M. Lipasti, “SCARAB: A single cycle adaptive routing and bufferless network,” in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture.* ACM, 2009, pp. 244–254.

[29] M. Hayenga and M. Lipasti, “The NoX router,” in *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-44, 2011, pp. 36–46.

[30] Y. He, H. Sasaki, S. Miwa, and H. Nakamura, “Predict-more router: A low latency NoC router with more route predictions,” in *Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2013 IEEE 27th International*, May 2013, pp. 842–850.

[31] R. Hesse, J. Nicholls, and N. Enright Jerger, “Fine-grained bandwidth adaptivity in networks-on-chip using bidirectional channels,” in *International Symposium on Networks on Chip Symposium*, 2012, pp. 132–141.

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

[33] C.-C. Huang and V. Nagarajan, “Increasing cache capacity via critical-words-only cache,” in *International Conference on Computer Design.* IEEE, 2014, pp. 125–132.

[34] N. E. Jerger and L.-S. Peh, “On-chip networks,” *Synthesis Lectures on Computer Architecture*, vol. 4, no. 1, pp. 1–141, 2009.

[35] N. Jiang, D. Becker, G. Michelogiannakis, J. Balfour, B. Towles, D. Shaw, J. Kim, and W. Dally, “A detailed and flexible cycle-accurate network-on-chip simulator,” in *International Symposium on Performance Analysis of Systems and Software*, April 2013, pp. 86–96.

[36] H. Kim, Y. Kim, and J. Kim, "Clumsy flow control for high-throughput bufferless on-chip networks," *Computer Architecture Letters*, vol. 12, no. 2, pp. 47–50, July 2013.

[37] J. Kim, "Low-cost router microarchitecture for on-chip networks," in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 2009, pp. 255–266.

[38] A. Kumar, P. Kundu, A. Singhx, L.-S. Peh, and N. Jha, "A 4.6Tbits/s 3.6GHz single-cycle NoC router with a novel switch allocator in 65nm CMOS," in *International Conference on Computer Design*, Oct 2007, pp. 63–70.

[39] A. Kumar, L.-S. Peh, P. Kundu, and N. K. Jha, "Express Virtual Channels: Towards the ideal interconnection fabric," in *Proceedings of the 34th Annual International Symposium on Computer Architecture*, ser. ISCA '07, 2007, pp. 150–161.

[40] J. W. Lee, M. C. Ng, and K. Asanovic, "Globally-synchronized frames for guaranteed quality-of-service in on-chip networks," in *Computer Architecture, 2008. ISCA'08. 35th International Symposium on*. IEEE, 2008, pp. 89–100.

[41] S. Ma, N. Enright Jerger, and Z. Wang, "Whole packet forwarding: Efficient design of fully adaptive routing algorithms for networks-on-chip," in *Proceedings of the 2012 IEEE 18th International Symposium on High-Performance Computer Architecture*, ser. HPCA '12, 2012, pp. 1–12.

[42] H. Matsutani, M. Koibuchi, H. Amano, and T. Yoshinaga, "Prediction router: Yet another low latency on-chip router architecture," in *International Symposium on High Performance Computer Architecture*, Feb 2009, pp. 367–378.

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

[44] A. K. Mishra, O. Mutlu, and C. R. Das, "A heterogeneous multiple network-on-chip design: an application-aware approach," in *Proc. Int. Design Automation Conference*, 2013.

[45] 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, 2009, pp. 196–207.

[46] R. Mullins, A. West, and S. Moore, "The design and implementation of a low-latency on-chip network," in *Proceedings of the 2006 Asia and South Pacific Design Automation Conference*, ser. ASP-DAC '06, 2006, pp. 164–169.

[47] U. Y. Ogras and R. Marculescu, "Prediction-based flow control for network-on-chip traffic," in *Proceedings of the 43rd annual design automation conference*. ACM, 2006, pp. 839–844.

[48] U. Ogras, R. Marculescu, P. Choudhary, and D. Marculescu, "Voltage-frequency Island Partitioning for GALS-based networks-on-chip," in *Design Automation Conference, 2007. DAC '07. 44th ACM/IEEE*, June 2007, pp. 110–115.

[49] D. Park, R. Das, C. Nicopoulos, J. Kim, N. Vijaykrishnan, R. Iyer, and C. Das, "Design of a dynamic priority-based fast path architecture for on-chip interconnects," in *High-Performance Interconnects, 2007. HOTI 2007. 15th Annual IEEE Symposium on*, Aug 2007, pp. 15–20.

[50] L.-S. Peh and W. J. Dally, "A delay model and speculative architecture for pipelined routers," in *High-Performance Computer Architecture, 2001. HPCA. The Seventh International Symposium on.* IEEE, 2001, pp. 255–266.

[51] L.-S. Peh and W. Dally, "Flit-reservation flow control," in *International Symposium on High-Performance Computer Architecture*, 2000, pp. 73–84.

[52] Reetuparna Das and Onur Mutlu and Thomas Moscibroda and Chita R. Das, "Aergia: exploting packet latency slack in on-chip networks," in *Proc. Int. Symp. Computer Architecture*, 2010.

[53] J. San Miguel and N. Enright Jerger, "Data criticality in network-on-chip design," in *Proceedings of the International Symposium on Networks on Chip*, 2015.

[54] C. Sun, C.-H. Chen, G. Kurian, L. Wei, J. Miller, A. Agarwal, L.-S. Peh, and V. Stojanovic, "DSENT - a tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling," in *International Symposium on Networks on Chip*, May 2012, pp. 201–210.

[55] M. Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffman, P. Johnson, J.-W. Lee, W. Lee, A. Ma, A. Saraf, M. Seneski, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, and A. Agarwal, "The RAW microprocessor: a computational fabric for software circuits and general-purpose programs," *Micro, IEEE*, vol. 22, no. 2, pp. 25–35, Mar 2002.

[56] J. van den Brand, C. Ciordas, K. Goossens, and T. Basten, "Congestion-controlled best-effort communication for networks-on-chip," in *Design, Automation Test in Europe Conference Exhibition, 2007. DATE '07*, April 2007, pp. 1–6.

[57] S. Volos, C. Seiculescu, B. Grot, N. Pour, B. Falsafi, and G. De Micheli, "CCNoC: Specializing on-chip interconnects for energy efficiency in cache-coherent servers," in *International Symposium on Networks on Chip*, May 2012, pp. 67–74.

[58] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C.-C. Miao, J. F. Brown III, and A. Agarwal, "On-chip interconnection architecture of the tile processor," *IEEE Micro*, vol. 27, no. 5, pp. 15–31, Sep. 2007.

[59] S. Woo, M. Ohara, E. Torrie, J. Singh, and A. Gupta, “The SPLASH-2 programs: characterization and methodological considerations,” in *International Symposium on Computer Architecture*, June 1995, pp. 24–36.

[60] J. Zhan, J. Ouyang, F. Ge, J. Zhao, and Y. Xie, “DimNoC: A dim silicon approach towards power-efficient on-chip network,” in *Proceedings of the 52nd Annual Design Automation Conference*, ser. DAC ’15, 2015, pp. 10:1–10:6.

[61] J. Zhan, N. Stoimenov, J. Ouyang, L. Thiele, V. Narayanan, and Y. Xie, “Optimizing the NoC slack through voltage and frequency scaling in hard real-time embedded systems,” *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 33, no. 11, pp. 1632–1643, Nov 2014.