ENERGY-EFFICIENT
FLOW-CONTROL FOR
ON-CHIP NETWORKS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL
ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Georgios Michelogiannakis
June 2012

Abstract

With the emergence of on-chip networks, the power consumed by router buffers has become a primary concern. Bufferless flow control has been proposed to address this issue by removing router buffers and handling contention by dropping or deflecting flits. In this thesis, we compare virtual-channel (buffered) and deflection (packet-switched bufferless) flow control. Our study shows that unless process constraints lead to excessively costly buffers, the performance, cost and increased complexity of deflection flow control outweigh its potential gains. To provide buffering in the network but without the cost and timing overhead of router buffers, we propose elastic buffer (EB) flow control which adds simple control logic in the channels to use pipeline flip-flops (FFs) as EBs with two storage locations. This way, channels act as distributed FIFOs and input buffers as well as the complexity for virtual channels (VCs) are no longer required. Therefore, EB networks have a shorter cycle time and offer more throughput per unit power than VC networks. We also propose a hybrid EB-VC router which is used to provide traffic separation for a number of traffic classes large enough for duplicate physical channels to be inefficient. These hybrid routers offer more throughput per unit power than both EB and VC routers. Finally, this thesis proposes packet chaining, which addresses the tradeoff between allocation quality and cycle time traditionally present in routers with VCs. Packet chaining is a simple and effective method to increase allocator matching efficiency to be comparable or superior to more complex and slower allocators without extending cycle time, particularly suited to networks with short packets.

Acknowledgements

First and foremost, I would like to thank my academic advisor Prof. William J. Dally. Bill has shown tremendous commitment to teaching and mentoring. I have always found it remarkable how he is able to fit in individual and group meetings with students in his schedule even though his list of responsibilities keeps growing, especially when he was chairing the Computer Science department. Bill has supported me over the years in more ways than I can possibly remember. He has always given me space to develop my own creativity, has always believed that I could handle everything that came up, and has provided me with opportunities. In addition, Bill is a remarkable individual because he has many impressive hobbies and he excels in all of them. Overall, Bill is an amazing role model.

I would also like to thanks Prof. Christos Kozyrakis for his support and mentoring over the years, as well as serving as a member of my reading committee. Also, I would like to thank Prof. Nick Mckeown for eagerly acceptive to be the third member of my reading committee.

Of course, these five years of my life were made enjoyable with the company of my close friends, with whom I have had some great moments away from stress of graduate school. I would especially like to thank Christine Moore, Daniel Sanchez, Christina Delimitrou, Elliot English, Panagiotis Tsiligkiris, George Karakonstantakis and other members of the Greek community in Stanford. I can only hope to keep in touch in the future.

Furthermore, I thank members of the CVA group, especially those who were admitted in the same year as me and therefore spent most of the time with me: Curt Harting, Nan Jiang and Daniel Becker. Daniel deserves a special mention for being

my officemate which means that we shared many humorous moments together. I would also like to thank David Black-Schaffer for introducing me to the *Selection Natural* soccer team in which I had the opportunity to play soccer with some great friends. Finally, I would like to thank James Chen for introducing me to tennis and table tennis.

Last but not least, I would like to thank my family and friends back in Greece for their support and encouragement during my time at Stanford as well as for my future endeavours. My parents have been especially understanding when it came to the difficulties and duration of a PhD.

The work in this thesis was supported in part by the National Science Foundation under Grant CCF-0702341, in part by the National Security Agency under Contract H98230-08-C-0272-P007 and in part by the Robert Bosch Stanford Graduate Fellowship.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In recent years, computer architecture research has experienced a substantial shift in focus. Technology scaling has slowed down, which has led to scaling difficulties for large uniprocessors [91]. Some of these difficulties are that the clock frequency of today's large uniprocessors can not simply be increased, or the power dissipation will become prohibitive. In addition, uniprocessor performance scaling is further hindered because only a limited amount of parallelism can be extracted from a typical instruction stream using conventional superscalar instruction techniques [96, 67]. Despite these problems, applications such as large datacenters, social networking, computer graphics, gaming, machine learning, as well as scientific applications can benefit from additional processing power and thus motivate research to provide processing power beyond that made possible solely by uniprocessor scaling [47].

This has sparked the rise of parallel computing. Parallel computing divides any given problem into subproblems. Each subproblem is handled by a single processing unit in parallel with the rest. Parallel computing faces numerous challenges from the application level all the way down to the architecture and VLSI level [4]. From the hardware's perspective, parallel computing systems are limited by the area and power costs of adding more processing units, as well as by the need for the processing units to communicate with each other and the cache hierarchy. From a programmer's point of view, there are many crucial problems, such as parallel programming which includes parallelization of workloads, programming languages, task placement and scheduling, as well as communication patterns between processing units and tasks. Applications are often limited by their inherent lack of parallelism, which prevents them from taking advantage of all the processing units available to them.

There has been substantial research on parallel computing focusing on each of the important challenges. Large-scale chip multiprocessor (CMP) architectures have been developed to fill a processor die with multiple processing units. The processing units in a CMP can vary from simple in-order pipelines to moderately complex superscalar processing units. Even though in many systems using many simple processing units is more cost efficient than fewer complex ones [18, 19], design requirements and the inherent parallelism of the anticipated applications dictate the choice of processing units. CMPs can take full advantage of future technology scaling because they can always deploy more processing units, instead of making the existing ones more complex. An added benefit of multi-core systems is redundancy which can be used for error detection and correction, as well as the ability to power gate idle processing units while others perform computations [108].

Apart from processing units, CMPs also require cache blocks and other blocks (such as memory controllers). The majority of CMPs use private L1 caches attached to each processing unit. However, there have been numerous proposals for L2 caches that can be shared or private, and centralized or distributed across the chip [98, 114]. Each such block connects to an on-chip router through a network interface in order to communicate with other processing units, cache blocks, memory controllers or other units in order to facilitate communication for parallel workloads, task migration, and other functions. This is illustrated in Figure 1.1.

There have been multiple parallel programming languages or libraries for existing languages which provide programmers the ability to spawn threads to communicate via explicit message passing [101] or via sharing memory address space [53]. Past research on parallel computing not only addresses numerous shortcomings, but also highlights the importance of communication patterns, data placement and the network fabric, because the constant exchange of information requires energy, as well as is an important factor to performance [98, 102, 112]. These factors play a more important role in large-scale systems.

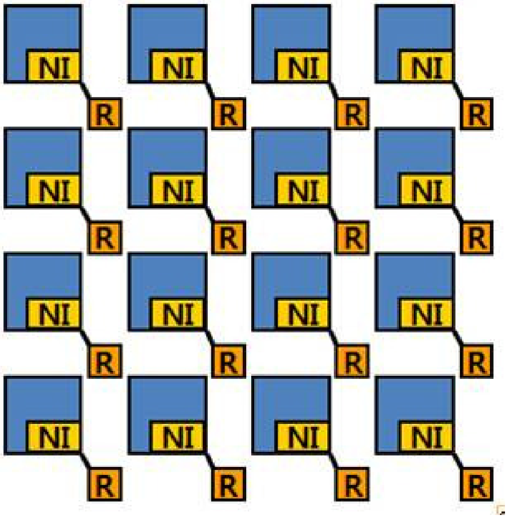


Figure 1.1: Abstract illustration of a multi-core architecture.

Small-scale CMPs have the option of using interconnect schemes such as buses, rings and crossbars [66, 24]. While buses are relatively simple, they suffer from performance scalability issues because all communication is serialized. Moreover, arbitration for the shared medium can take many cycles due to the propagation delay to and from the allocator. On the contrary, crossbars eliminate serialization by providing a separate path from each source to each destination. Unfortunately crossbars also do not scale because the area and power costs of a crossbar increase quadratically with the number of sources and destinations. Finally, rings have a large average hop count because it is proportional to the number of routers, and all traffic shares the same links, making bandwidth a potential bottleneck. Hence, none of these three interconnect options are appropriate for large-scale CMPs.

1.1 Overview of On-Chip Networks

1.1.1 Basic Architecture of On-Chip Networks

Packet-switched on-chip networks have been proposed as a scalable and modular communication medium for large-scale systems, such as CMPs [23, 27, 92]. On-chip

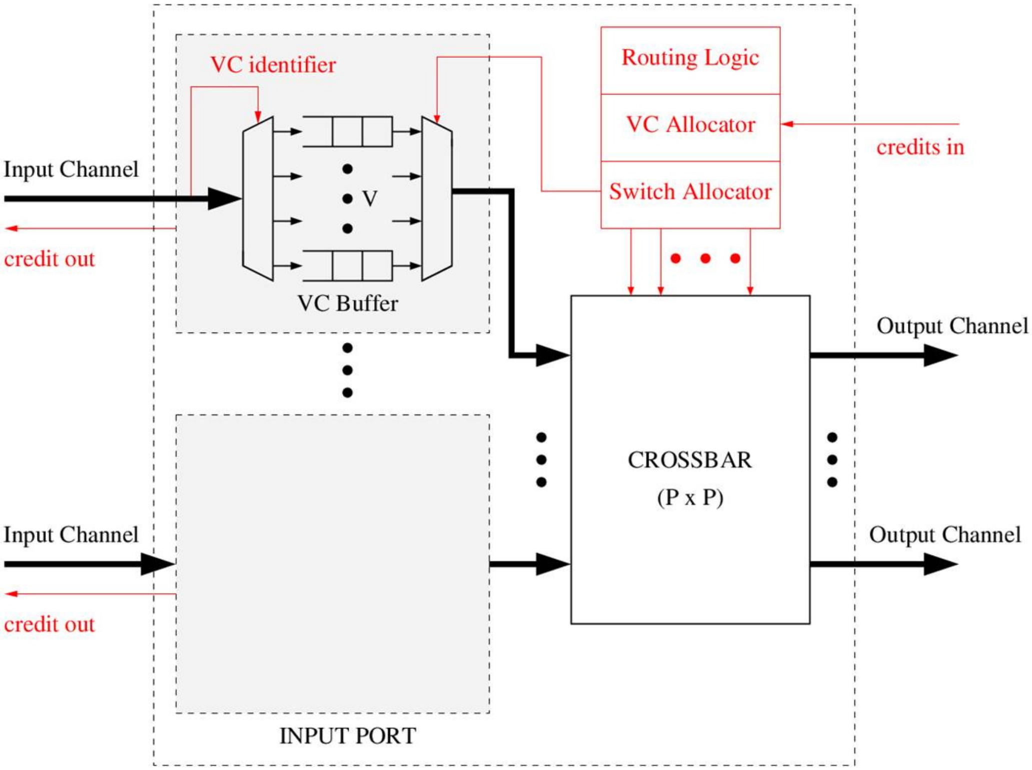


Figure 1.2: Architecture of a typical VC router.

networks are composed of a set of routers connected via point-to-point links in a manner specified by the network topology. Packets are divided into flow control digits (flits). Flits may be composed of multiple–but usually one–physical digits (phits); the size of phits is defined by the network datapath width. Large data flows are typically divided into multiple packets in order to aid resource allocation and routing in the network. With this organization, packets are transferred across the narrower channels over several cycles, incurring a serialization latency which is equal to the number of cycles to transmit the tail (last) flit after submitting the head (first) flit, in the absence of backpressure. The head flit carries the destination address that routers use to determine the proper output port and virtual channel (VC) [22] for the whole packet. Routing can be either deterministic (always follows the same path between any two points), oblivious (includes random choices) or adaptive (network stage such as congestion is taken into account).

Routers are the basic building blocks of scalable interconnects. Routers using VC flow control use per-VC input port buffers. Head flits at buffer heads go through routing computation and VC allocation. Non-head flits are assigned the same output port and VC as their head flit. Flits then proceed to switch allocation. Allocators try to find the best match considering all requests and output port states. Winning flits traverse the switching fabric, which delivers them to the proper output port. Routers are typically pipelined and several speculation or pre-computation techniques are used to reduce the critical path or the latency under low load [85, 59]. A typical VC router is illustrated in Figure 1.2.

Routers optimized for latency use look-ahead routing which performs the routing computation for the packet's next hop [32]. This way, packets enter the router with their outputs pre-encoded in the head flit. Thus, low-latency routers perform look-ahead routing, VC allocation and switch allocation in the first pipeline stage and switch traversal in the second. To achieve this, speculative VC allocation is used which enables VC and switch allocations to be performed in parallel for all packets, but packets can only advance if they receive both grants [85]. Other routers use more pipeline stages to reduce complexity by removing speculation. This frequently shortens cycle time because it places VC allocation, switch allocation and perhaps routing into separate pipeline stages.

On-chip networks typically use VC flow control to enable deadlock avoidance, optimize channel utilization, improve performance and provide quality of service (QoS) [22, 13]. Disjoint traffic classes use separate VCs and routing algorithms which are designed to avoid cycles within and across VCs [29]. Blocked flits in one VC do not affect flits in other VCs since they are stored in separate FIFOs of each input's buffers. Per-VC credits are used to avoid input buffer overflow. A router can transmit a flit only if it has a credit to consume from the downstream router for that VC. Credits represent free slots in the corresponding next-hop buffer for that VC.

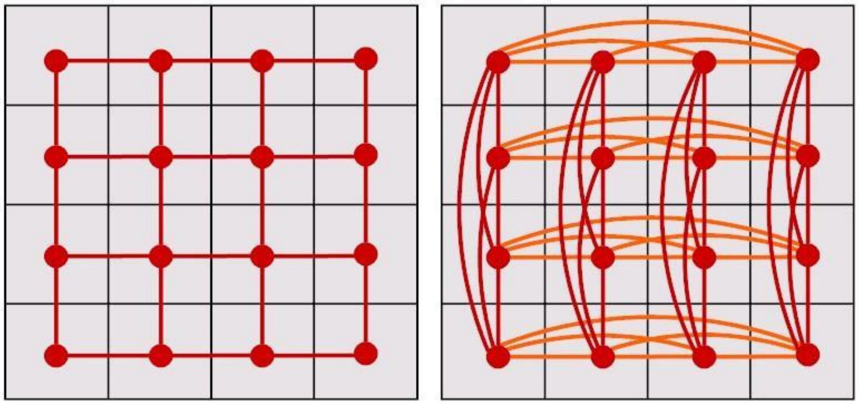


Figure 1.3: Two representative topologies in a 2D layout: 2D mesh (left) and 2D flattened butterfly (right).

1.1.2 Topology

The topology defines how routers are connected to each other and to the network endpoints. For a large-scale system, the topology has a major impact on the performance and cost of the network. A variety of topologies have been used in on-chip networks [27, 13]. In this section, we discuss the 2D mesh and the 2D flattened butterfly (FBFly) [57] because they are used later in this thesis for performance evaluation. These topologies represent interesting design points since they offer distinct advantages and disadvantages and are fundamentally different. Figure 1.3 illustrates these topologies. Table 1.1 provides an asymptotic comparison of the two topologies for key metrics. Similar topologies exist to the ones we discuss in this section. For example, topologies which have direct links between distant routers (express links), such as express cubes [39, 25] or other low-diameter networks [113], can have similar advantages as the FBFly due to their reduced hop count. Finally, the fat tree topology [69] connects routers in a tree manner, and uses an increasing number of point-to-point links for routers closer to the root.

2D mesh . The 2D mesh is a popular topology choice in large-scale CMPs [11, 48]. Each of the $\frac{T}{C}$ routers connects to its four neighboring routers and C source or destination nodes (network endpoints). The degree of concentration C, in nodes per router, is typically applied to reduce the number of routers and hops.

Table 1.1: Comparison of two popular topologies for a CMP system with T sources and destinations (cores, cache banks, and controllers). The point-to-point links have a width of W bits. N is the number of routers in the network and P is the number of input/output ports of each one. C refers to the concentration factor. The hop count is the number of link traversals under dimension-order minimal routing. The average number of hops assumes uniformly distributed traffic.

	Routers (N)	**Router I/Os (P)**	**Bisection BW**
2D mesh	$\frac{T}{C}$	$(C+4)$	$\sqrt{N}W$
2D FBFly	$\frac{T}{C}$	$(C+2(\sqrt{N}-1))$	$\sqrt{N}W\lfloor\frac{N}{2}\rfloor$

	Hops (worst)	**Hops (average)**
2D mesh	$2\sqrt{N}$	$\sim\sqrt{N}+1$
2D FBFly	4	3.5

A mesh topology with a concentration C of more than one is commonly referred to as a cmesh [40, 24].

The major advantage of the mesh is its simplicity. Links are short and balanced and the overall layout is very regular. The routers are low radix with up to $C+4$ input and output ports, which reduces their area footprint, power overhead and critical path. The major disadvantage of the mesh is the large number of hops that flits have to potentially go through to reach their final destination; the number of hops is $2\sqrt{N}$ for N routers (the hop count is the number of link traversals under dimension-order minimal routing). Each router imposes a minimum latency (e.g. 2 cycles for latency-optimized routers) and is a potential point of contention. A large number of hops has direct impact on the energy consumed in the interconnect for buffering, transmission, and control, Hence, meshes could face performance and power scalability issues for large-scale systems. To address these concerns, researchers have proposes meshes with physical [25] or virtual [65] express links.

2D Flattened Butterfly . The 2D FBFly is derived by flattening the routers in each row of a conventional butterfly topology while preserving inter-router connections [57]. This way, routers connect with every other router in each axis. Essentially, this topology provides the connectivity of a mesh with additional express links. Thus, in a 4×4 network, each router connects with three other routers in the X axis, and with three other routers in the Y axis. Similarly to the mesh, a concentration factor is typically applied to reduce the router overhead. If the concentration factor in our example is four nodes per router, each router is 10×10.

The major advantage of the FBFly is the small number of hops for network traversals under minimal routing. For two dimensions, flits can always reach any destination node with four hops (i.e. three routers). Using the longer links minimizes the number of routers visited and their associated latency and energy overheads. The additional links reduce the chance of congestion and provide higher bandwidth as well. The major disadvantage of the FBFly is the need for high-radix routers, which are expensive in terms of area and power due to the large switches they require. The larger number of links also increases area and leakage power.

1.2 On-Chip Network Consumption and Motivation

Current and past on-chip network research shares the common goal of reducing power consumption compared to that of the *ideal interconnect*. The ideal interconnect represents an abstract network where the only power consumed is that for propagation between senders and receivers. This does not include any of the overhead present in realistic networks, such as buffers, allocators and switches, which were added for the purpose of providing a functionally correct network, and perhaps provide additional services such as QoS. Because such overhead is necessary, realistic implementations aim to reduce the gap, illustrated in Figure 1.4 and originally presented in [65], but can never fully close it. Much of the complexity of past proposals to reduce energy stems from the fact that such techniques must also respect other constraints of the network, such as cycle time, area and latency. Thus, optimizing for power tends to

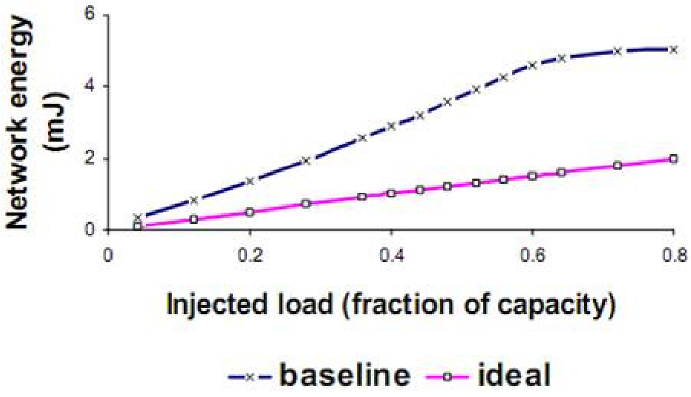


Figure 1.4: The power consumption gap between realistic on-chip networks and the ideal interconnect which only consumes power for data propagation.

focus on increasing throughput per unit of power, while also respecting the other constraints.

Recently, Intel produced the Teraflop processor, with 80 single-precision, floating-point cores [48]. This design delivers performance in excess of teraflops and to a large degree was an attempt to identify shortcomings of current architectures in preparation for future many-core designs. The Intel 80-core Teraflop processor is organized in tiles. Each tile contains instruction and data memory slices, computation blocks and an on-chip router. The floorplan of the design is illustrated in Figure 1.5.

The Intel 80-core Teraflop processor features a 2D mesh to interconnect tiles. Routers are clocked at 5GHz, and the network is mesochronous (the clock frequency is the same throughout the design, but the clocks of adjacent clock regions need not be phase-aligned). Because of the long channels, high frequencies and high computational demands, the on-chip network consumes 26% of each tile's power, as illustrated in Figure 1.6. The rest of the power is consumed predominantly by clocking due to the high frequencies, and by the expensive floating point operations. This served as another confirmation to the on-chip network community of the importance of reducing the energy consumption of on-chip networks especially when considering larger-scale systems. Other studies have reached similar conclusions by reporting that instruction and data supply comprise 70% of the power for a RISC processor, compared to 5% for the processor datapath [5].

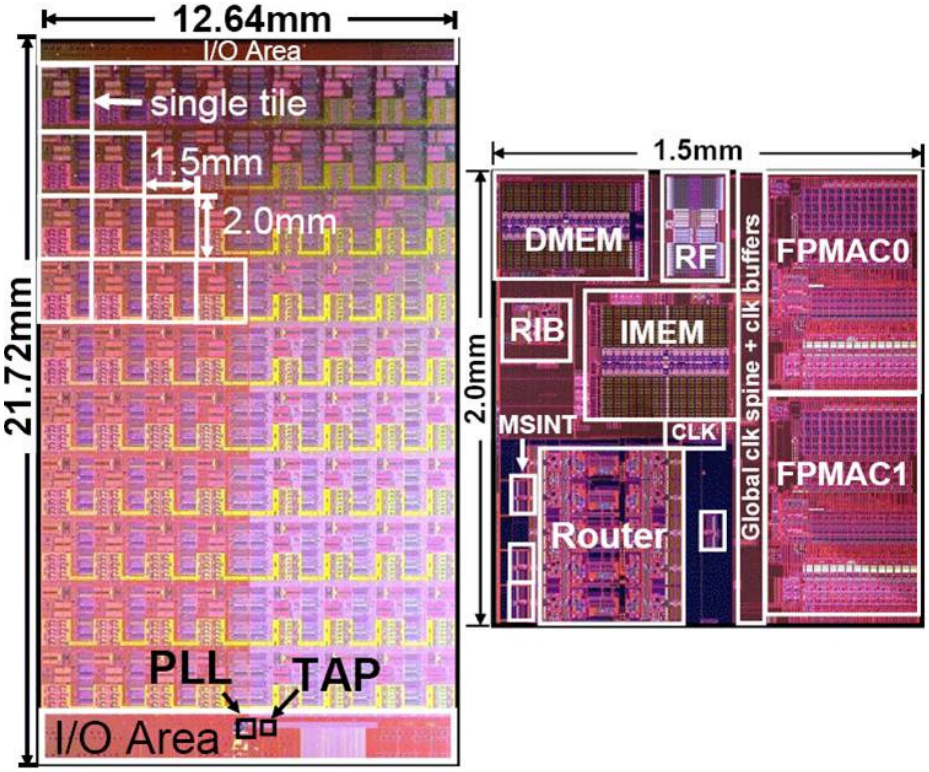


Figure 1.5: Overview of the Intel 80-core Teraflop processor.

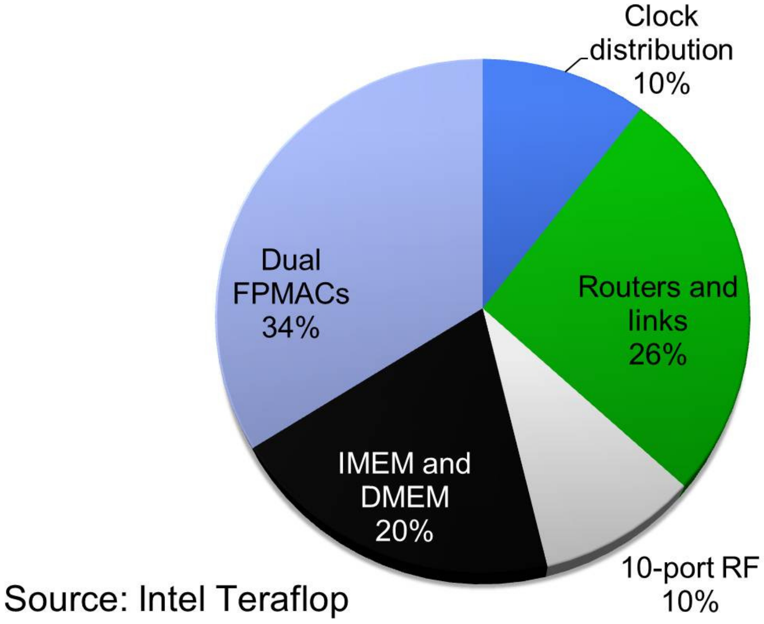
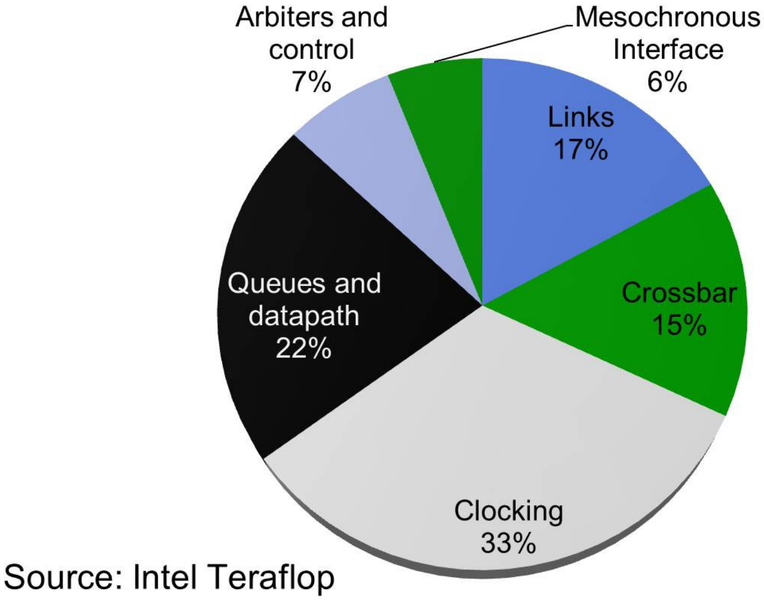


Figure 1.6: Power breakdown of a single tile in the Intel 80-core Teraflop processor.

Figure 1.7 illustrates the power breakdown of the on-chip network in the Intel 80-core Teraflop processor. All the components illustrated offer numerous opportunities for optimizations and have sparked research from custom circuit design to routing and flow control. However, the only component which is not fundamentally required and can thus be removed completely is the router buffers, which represents 22% of the network power. Channels are necessary to data and control information propagation. Crossbars are required to allow packets take turns and thus cannot be removed in a packet-switched network. Finally, allocation and routing logic is necessary for the correct operation of crossbars and packet movement in packet-switched networks. This has motivated research to eliminate router buffers.

Router buffers are used to handle contention. However, contention can be prevented or handled in other ways. Circuit-switched networks prevent contention by pre-allocating bandwidth [24]. However, circuit-switched networks impose large latency and power overheads because circuits need to be established before packet transmission, even for small packets. Bufferless networks handle contention by dropping



Source: Intel Teraflop

Figure 1.7: Network power breakdown in the Intel 80-core Teraflop processor.

packets under contention and relying on the source to retransmit them, or deflecting them to any free output. Since bufferless flow control does not have the cost and timing overhead of router buffers which has been shown to be significant, bufferless flow control has been researched recently as a way to reduce the energy of on-chip networks [83, 30].

1.3 Thesis Overview

In this thesis, we perform a comprehensive study to improve and evaluate bufferless flow control as a viable solution to the rising router buffer costs. To avoid the complications and complexity of bufferless flow control illustrated in our study, this thesis proposes elastic buffer (EB) flow control which provides buffering in the network without the timing and cost overhead of router buffers. Finally, this thesis proposes packet chaining which increases the matching quality of switch allocators to be comparable or superior to more complex and slower allocators.

Our bufferless study is presented in Chapter 2; it focuses on deflection networks and starts by improving routing in deflection networks. For this, we propose multi-dimensional routing (MDR) which defines all outputs which reduce the distance of packets to their destination as *productive*, instead of only the output according to dimension-order routing (DOR). This way, deflection becomes less probable because there are more productive outputs. This reduces latency by an average of 5% compared to BLESS [83], which describes a deflection flow control scheme we use for our comparisons. Furthermore, we synthesize a switch allocator required by deflection flow control, and show that because flits cannot wait in inputs, all inputs must receive a grant to an output, and thus the allocator has a long serial path that traverses every output arbiter. Deflection allocators also need to prioritize older packets to prevent livelocks. Therefore, deflection switch allocators have an 81% longer delay than switch allocators for a VC network with two VCs.

In addition, our bufferless study examines closely the implementation and usage of input buffers. Not only can input buffers be implemented efficiently with custom SRAM blocks [6], but empty buffers can be bypassed in the absence of contention such that dynamic energy is not consumed when not necessary [107]. This way, power savings for the bufferless network are marginal at best. Since one deflection induces two extra hops, deflections consume 6.7× the dynamic energy for buffering contending flits instead. Specifically, the deflection network only consumes less power than the buffered network for injection rates lower than 7%, but never consumes less power than 98.7% of the power for a 2D mesh buffered network. That is because at low loads leakage power provides a marginal advantage to bufferless networks. On the contrary, increasing the injection rate increases the number of deflections, which increases the channel activity factor by 4.6% at an injection rate of 20%, which equals 5.5× the buffer access and leakage power. Moreover, because each deflection causes an even number of hops, the maximum latency for BLESS is 108 cycles with an injection rate of 20%, whereas for the buffered network it is 13. The buffered network also has a 17% lower average latency. Moreover, the VC network provides 21% more throughput per unit power. Finally, bufferless networks need large buffers at the endpoints to prevent protocol deadlocks and buffer overflow due to the lack of backpressure.

To retain buffering in the network without the cost for router buffers and without the complications and complexity of bufferless flow control, we propose EB flow control in Chapter 3. EB flow control adds a simple control logic block to drive the enable inputs of the master and slave latches of master-slave pipeline flip-flops (FFs) separately. This allows FFs to use each latch independently for buffering and makes FFs behave as two-slot FIFOs (EBs). Consecutive EBs make network channels act as distributed FIFOs. Flits advance to the next EB using a ready-valid handshake. With EB flow control, channels are used for buffering in lieu of input buffers.

Removing input buffers removes VCs. This increases head-of-line blocking and therefore reduces performance. However, area and power are also reduced from removing the buffers. Therefore, the datapath can be made wider such that performance or cost is equalized compared to a network with input buffers. Removing VCs also removes the VC allocator. Consequently, because an input may only request a single output, the switch allocator is replaced by output arbiters. Furthermore, credits are not used. Therefore, routers are significantly simplified compared to VC routers [85]. However, due to the lack of credits, EB networks use a congestion sensing mechanism which counts flits bidding for and traversing output channels.

Two EB router designs are presented. The two-stage EB router emphasizes on throughput by reducing cycle time. Compared to a VC router with speculative switch allocation [85], the two-stage EB router offers 8% more throughput per unit power in a 2D mesh with DOR, assuming equal clock frequencies. It also reduces cycle time by 45%. The second EB router design merges the two-stages of the two-stage router to avoid pipelining overhead and prioritize latency. Compared to the two-stage router, this single-stage router requires 29% less energy per transferred bit but also has a 33% longer cycle time. By using the optimal EB router and shortest cycle time for each comparison, a 2D mesh EB network provides 21% more throughput per unit power, 22% more throughput per unit area or has an up to 45% shorter cycle time compared to a similar VC network. Furthermore, because EB networks have a wider datapath when equalizing performance or cost, they reduce zero-load latency.

To provide complete traffic separation, multiple *physical channels are used in the same way as multiple* virtual channels (VCs). An efficient way to provide multiple physical channels is by using duplicate and independent physical networks. However, this is only efficient up to a number of traffic classes. To provide an arbitrarily large number of traffic classes, we propose a hybrid EB-VC router which reinstates input buffers with VCs but only to drain flits at router inputs that are blocked for a predefined number of cycles, as well as other flits belonging to the same class. Transmission of further flits of that class is hindered by a control signal routed upstream.

A properly-chosen predefined threshold makes hybrid EB-VC routers more energy efficient than VC routers because buffers are not used in the common case, as well as more energy efficient than EB routers because hybrid routers reduce head-of-line blocking (in addition to resolving deadlocks). Specifically, in a 2D mesh, networks with EB-VC routers offer 21% more throughput per unit power than VC routers, and 12% than EB routers. However, the presence of buffers as well as VC and switch allocators makes hybrid routers comparable to VC routers in area and cycle time. Even though area is comparable, hybrid routers still allow limited interaction between flits in different VCs. Thus, VC networks offer 41% more throughput per unit area compared to hybrid EB-VC networks, while EB networks offer 49%.

Cycle time can be crucial for network performance. However, VC and hybrid EB-VC networks require allocators to assign VCs to packets and permit them to traverse the switch [22, 85]. Because allocators are often in the router's critical path and the network is sensitive to allocator performance, there is a tradeoff between allocation quality and cycle time. To increase allocation quality without extending cycle time, in Chapter 4 we propose *packet chaining*, a method for improving allocation efficiency for iterative allocators that is particularly suited to networks with short packets and short cycle times.

Packet chaining chains packets together, facilitating reuse of a departing packet's switch connection. Connections are established when head flits are granted by the switch allocator, as in incremental allocation [84]. This way, a collection of short packets destined to the same output keep resources reserved in the switch allocator similar to longer packets. This allows an allocator to build up an efficient matching over a number of cycles like incremental allocation, but not limited by packet length.

Packet chaining includes starvation control because connections could otherwise remain active indefinitely.

Packet chaining is implemented by adding a separate packet chaining (PC) allocator, and thus doubles the area and power for allocation. The allocation timing path is lengthened only marginally, since the PC and switch allocators operate in parallel. Packet chaining increases the allocation efficiency of a single-iteration separable allocator to be comparable to or higher than more expensive and slower allocators. For single-flit packets at maximum injection rate, packet chaining increases network throughput by 15% and reduces average latency by 22.5%, compared to a highly-tuned router with incremental allocation using a single-iteration iSLIP switch allocator [74]. Packet chaining also outperforms multi-iteration iSLIP allocators and wavefront allocators [103] by 10% and 6% respectively, and gives comparable throughput with an augmenting path allocator [31]. Performance gains decrease as packet length increases because incremental allocation creates connections as well. However, packet chaining still provides better or comparable throughput with more expensive and slower allocators for packets of any length. Finally, packet chaining increases IPC up to 46% (16% average) for a CMP executing application benchmarks, because short packets are critical in typical cache-coherent CMPs.

1.4 Thesis Contributions

This thesis makes the following contributions:

- We conduct a comprehensive evaluation of bufferless flow control, focusing on deflection flow control. In this study, we first optimize routing in bufferless networks and then show the complications and complexity of bufferless flow control. Thus, we conclude that bufferless flow control is not a viable technique to remove router buffers and consequently their cost and timing overhead.

- To provide buffering in the network without the cost for router buffers we propose EB flow control. We describe the single-stage EB router which focuses on reducing latency and the two-stage EB router which minimizes cycle time. For also propose a congestion-sensing mechanism to be used instead of credit count. Traffic separation is provided with separate independent networks. However, for a large number of traffic classes, we propose the hybrid EB-VC router which can handle an arbitrary amount of traffic classes and also increases throughput per unit area compared to both VC and EB networks.

- We propose packet chaining which increases the matching quality of iterative allocators to be comparable or superior to more complex and slower allocators. Packet chaining is an elegant solution to the tradeoff between matching quality and cycle time, and focuses on single-flit packets which are critical in typical cache-coherent CMPs.

1.5 Thesis Structure

The rest of this thesis is structured as follows: Chapter 2 presents our study which illustrates the complications of bufferless flow control. Chapter 3 presents elastic buffer flow control which provides buffering to the network without the cost for router buffers. Chapter 4 presents packet chaining which increases allocation quality without increasing cycle time. Chapter 5 discusses related work. Finally, Chapter 6 concludes and provides future research directions.

Chapter 2

Bufferless Flow Control

This chapter examines if bufferless networks are an efficient alternative to buffered networks [81]. This is done by comparing a state-of-the-art packet-switched bufferless network with deflecting flow control, BLESS [83], and the currently-dominant virtual channel (VC) buffered flow control [22]. To perform an equitable comparison, both networks are optimized. In particular, VC networks feature efficient custom SRAM buffers and empty buffer bypassing. This chapter also proposes multi-dimensional routing (MDR), a novel routing scheme for BLESS, where flits bid for all outputs that would reduce their distance to their destinations regardless of dimension order constraints. As shown in this chapter, bufferless networks offer a minimal energy advantage at best, while offering lower throughput and facing complications such as allocation time and latency distribution. Therefore, bufferless flow control is not a viable solution to the rising router buffer costs.

Bufferless flow control aims to reduce on-chip network cost and timing overhead by removing the router buffers, which are often reported to consume significant amounts of area and power. Specifically, bufferless flow control proposals report up to 60% area and up to 39% energy savings in a conventional CMP network [83]). Other cost models and buffer implementations quote numbers for the buffers as high as 75% [37] of the router area and 22% [54] of the router energy. Finally, recent designs from Intel consume 28% of the chip energy in the on-chip network, and 22% of the router power in the buffers [48].

The cost of router buffers has motivated research to decrease it. This research focuses both on buffer implementation and usage. Network designers have to choose between custom SRAMs, compiler-generated SRAMs, and flip-flop (FF) or latch arrays. While buffer implementation is a tradeoff of engineer time and cost, custom SRAMs can significantly reduce implementation costs. Research on buffer usage has proposed schemes to use buffer space more efficiently, thus reducing buffer size or dynamic power. These optimizations may reduce buffering overhead up to a point where the extra complexity and performance issues of bufferless flow control outweigh potential cost savings. Related work is analyzed in Section 5.

2.1 Methodology

To perform the evaluations, we use a modified version of Booksim [24] for cycle-accurate microarchitecture-level network simulation. To estimate area and power we use ITRS predictions for a 32nm high-performance process [50], operating at 70°C. Modeling buffer costs accurately is fundamental in our study. Orion [106] is the standard modeling tool in network-on-chip (NoC) studies, but recent studies show that it can lead to large errors [54, 55], and the update fixing these issues was not available at the time of this work. Instead, the models from Balfour and Dally [6] are used, which are derived from basic principles, and validate SRAM models using HSPICE.

The network is assumed to be clocked at 2GHz with 512-bit packets. Channel wires are routed above other logic and include only repeater and flip-flop area in channel area. The number and size of repeaters per wire segment are chosen to minimize energy. The conservative low-swing model of this chapter has 30% of the full-swing repeated wire traversal power and twice the channel area [45]. Router area is estimated using detailed floorplans. VC buffers use efficient custom SRAM-based buffers. We do not use area and power models for the allocators, but perform a detailed comparison by synthesizing them. Synthesis is performed using Synopsys Design Compiler and a low-power commercial 45nm library under worst-case conditions. Place and route is done using Cadence Silicon Encounter. Local clock gating is enabled.

Evaluation comparisons use FLIT-BLESS. FLIT-BLESS performs better than WORM-BLESS [83], but incurs extra overhead because all flits contain routing information. However, this overhead is not modelled in this chapter, giving BLESS a small advantage over buffered flow control. Furthermore, buffered BLESS has not been evaluated in [83], and therefore is not considered in our study.

Two topologies are used for a single physical network with 64 terminals. The first is an 8×8 2D mesh with single-cycle channels. Routers are 5×5 and have one terminal connected to them. The second is a 2D flattened butterfly (FBFly) [57] with four terminals connected to each router. Therefore, there are 16 10×10 routers laid out on a 4×4 grid. Short, medium and long channels are two, four and six clock cycles long, respectively. Injection and ejection channels are a single cycle long. For both topologies, one clock cycle corresponds to a physical length of 2 mm. These channel lengths are chosen so that both networks cover an area of about 200 mm^2.

Routers use a two-stage pipeline. The VC network features input-first separable round-robin allocators, speculative switch allocation [95] and input buffer bypassing [107]. No communication protocol is assumed. The VC network uses dimension-order routing (DOR) for the mesh and FBFly. The deflection network uses multi-dimensional routing, explained in Section 2.2. We do not assume adaptive routing for the VC network since such a comparison would require adaptive routing for the bufferless network as well. We choose the number of VCs and buffer slots to maximize throughput per unit power. While this penalizes the VC network in area efficiency, power is usually the primary constraint.

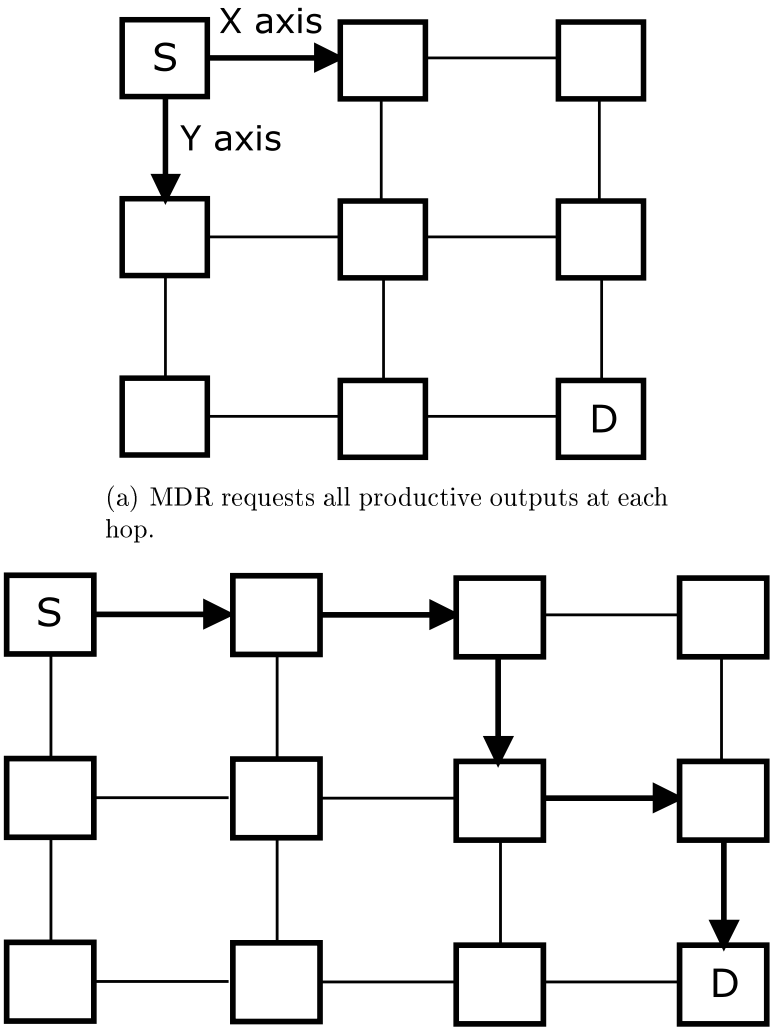
We generate results for either uniform random traffic or we average over a set of traffic patterns: uniform random, random permutation, shuffle, bit complement, tornado and neighbor [24]. This set is extended for the FBFly to include transpose and a traffic pattern that illustrates the effects of adversarial traffic for networks with a concentration factor. Averaging among traffic patterns makes our results less sensitive to effects caused by specific traffic patterns.

2.2 Routing in Bufferless Networks

BLESS networks use DOR [83]. In VC networks, DOR prevents cyclic network dependencies without extra VCs. Since flits are forced to complete their traversals in one dimension first, there can never be a flit wanting to turn from the last dimension back to the first one. However, in bufferless networks flits never block waiting for router buffers, so there can be no network deadlocks due to cyclic dependencies, making DOR unnecessary. As discussed in Section 2.4.4, extra care must be taken to prevent deadlocks arising from ejection buffers at network destinations, such as protocol deadlocks [42].

Two oblivious routing algorithms that decrease deflection probability are proposed here, based on the observation that a flit often has several *productive* outputs (i.e. outputs that would get the flit closer to its destination). For example, in a 2D mesh, those are the two outputs shown in Figure 2.1(a), unless the flit is already at one of the axes of its final destination. Our first routing algorithm, MDR, *exploits choice* by having flits request all of their productive outputs. If both outputs are available, the switch allocator assigns one pseudorandomly. However, this may make flits favor one dimension until there are no more hops in that dimension. Thus, flits will be forced to switch dimensions. During that time, they will have only one productive output, increasing their deflection probability.

With MDR, there is one productive output in each dimension with remaining hops. If a flit exhausts all hops in a dimension, it will have one less productive output, increasing its deflection probability. We can improve MDR by prioritizing the dimension that has the most remaining hops, which increases the number of productive outputs at subsequent hops. We call this scheme prioritized multi-dimensional routing (PMDR). Figure 2.1(b) shows an example path with PMDR in a 2D mesh. Due to PMDR, all the hops except the last one have two productive outputs. In an FBFly with minimal routing, flits only take one hop in each dimension, so PMDR is equivalent to MDR. However, PMDR increases allocator complexity: since a BLESS allocator already needs to prioritize flits by age, PMDR requires either prioritizing output ports or two allocation iterations.



(a) MDR requests all productive outputs at each hop.

(b) PMDR prioritizes the output with the most hops remaining in that dimension.

Figure 2.1: MDR and PMDR routing algorithms for deflection bufferless networks.

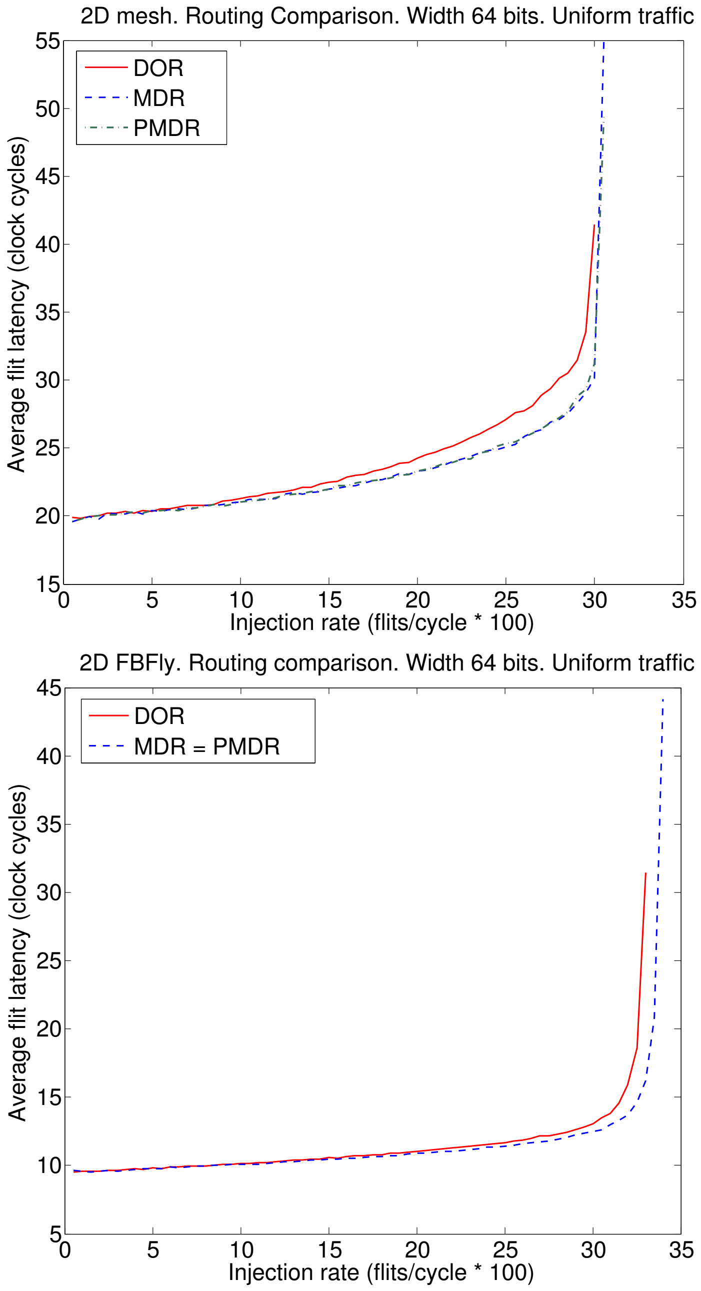


Figure 2.2: Comparison of DOR, MDR and PMDR routing algorithms.

Figure 2.2 compares DOR, MDR and PMDR in mesh and FBFly bufferless networks. In the mesh, MDR offers 5% lower average latency than DOR and equal maximum throughput. In the FBFly, MDR achieves 2% lower average latency and 3% higher maximum throughput under uniform random traffic as well as the average over the set of traffic patterns. On the other hand, latency is decreased by 5% for the mesh and 2% for the FBFly, by average over all injection rates, under uniform random traffic. Under a sample 20% flit injection rate, 13% more flits were only able to choose a single output with DOR compared to MDR. Also, PMDR achieves only marginal improvements over MDR (0.5% lower average latency in the mesh). Given its higher allocator complexity, we use MDR for the rest of the evaluation.

2.3 Router Microarchitecture

This subsection explores router microarchitecture issues pertinent to bufferless networks.

2.3.1 Allocator Complexity

To conduct the analysis, this section presents an implementation of an age-based BLESS allocator, shown in Figure 2.3. Requests are partially ordered according to their age by the sorting blocks shown on the left, each of which sorts two requests. Partial ordering is guaranteed to detect the oldest flit, which is sufficient to prevent livelocks. In order to reduce the complexity and shorten the timing path of the sorting logic, our implementation does not generate a complete ordering of flits by age. Requests are assigned priority based on the result of the partial sort, with the oldest flit having the highest priority. The output arbiters satisfy the highest-priority request from those remaining, and then forward remaining requests to the next arbiter. Because the oldest flit has the highest priority, it is guaranteed to not be deflected, thus preventing livelocks.

Although the logic of each arbiter is simple, all requests need to be granted [83] because all flits need to be routed due to the lack of buffers. This creates a long critical

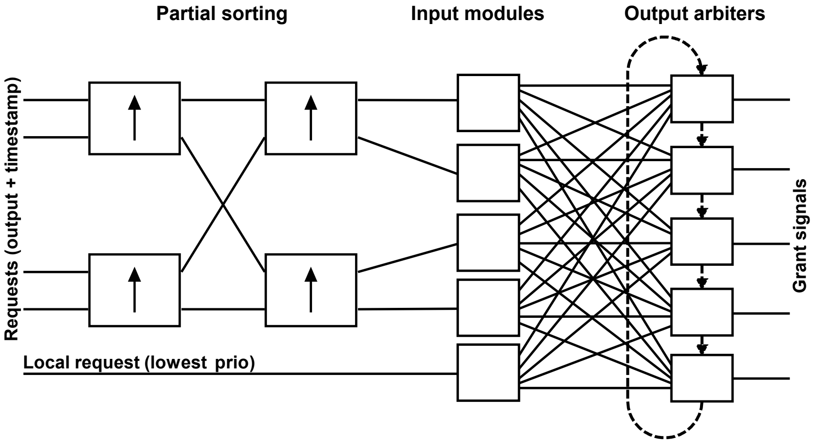


Figure 2.3: BLESS allocator implementation.

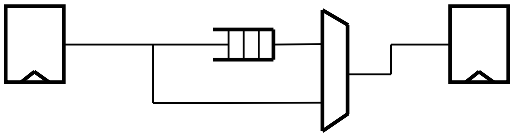


Figure 2.4: Empty buffer bypassing.

path that passes through each output. This critical path is necessary for deflection flow control, because the decision of a particular output arbiter may depend on the decision of other arbiters, because a flit that was not granted by other arbiters needs to be granted by that particular output. This critical path scales linearly with router radix. Note that this design would perform slightly worse than the idealized BLESS allocator that we use in our simulation-based evaluation. While the idealized BLESS allocator performs optimal matchings, the implementation described in this section trades off accuracy for cycle time. Therefore, BLESS is given an advantage by using an ideal allocator without considering timing and cost overheads.

Table 2.1 compares our BLESS allocator and an input-first separable speculative switch allocator with round-robin arbiters for a VC network with 2 VCs [10]. Dynamic power is estimated by applying Synopsys Design Compiler's default activity factor to all inputs. The VC network switch allocator represents a reasonable design and has comparable cycle time to other separable and wavefront allocators [10, 24]. We compare against the switch allocator because a similar VC allocator has a shorter

Table 2.1: Allocator synthesis comparison.

Aspect	Separable Input-First	Age-based	Delta
Number of nets	1165	1129	0%
Number of cells	1100	1050	0%
Area (μm^2)	2379	2001	-16%
Cycle time (ns)	1.6	2.9	+81%
Dynamic power (mW)	0.48	0.27	-44%

critical path and speculation parallelizes VC and switch allocation [10].

As Table 2.1 shows, the age-based allocator has an 81% larger delay. If the allocator is on the router's critical path, this will either degrade network frequency or increase zero-load latency if the allocator is pipelined. The separable allocator need to maintain state in FFs for the round-robin arbitration. Therefore, it requires more area and power for the FFs and the associated clock tree. However, the area and power differences are a small fraction of the overall router area and power, which are dominated by the buffers and the switch [54, 6]. Therefore, it is unrealistic to assume bufferless routers with one pipeline stage, assuming a tight timing budget. Speculation is not applicable to reduce router latency in bufferless networks because there is only one type of allocation (switch) and flits failing speculation need to be deflected.

2.3.2 Buffer Cost

The cost models of this study assume efficient custom SRAM blocks for buffers [6], which impose a smaller overhead compared to other implementation options. However, designers might be unable to use such designs, e.g. due to process library constraints. Using either standard-size SRAM blocks or FF arrays will likely make buffers more costly, increasing the motivation for eliminating them or reducing their size.

Additionally, we use empty buffer bypassing [107], shown in Figure 2.4. This allows flits to bypass empty buffers in the absence of contention. Empty buffer bypassing requires the addition of a FF to store flits which bypass the buffers, as well as a multiplexer to select among that FF and the buffer to feed the switch. Empty buffer bypassing has a negligible effect in timing and cost, but is able to drastically reduce the buffer dynamic power under light to medium load. Flits that bypass the buffer traverse that additional FF instead of the buffer, and therefore consume the energy to traverse one FF which is similar to traversing the pipeline FF of a two-stage bufferless router.

While these schemes suffice to implement efficient buffers, additional techniques could be applied. For instance, leakage-aware buffers [88] reduce leakage by directing incoming flits to the least leaky buffer slots and supply-gating unused slots. Also, to increase buffer utilization and reduce buffer size, researchers have proposed dynamic buffer allocation which makes more efficient use of buffer space, thus enabling a reduction in buffer size while maintaining performance constant [87].

To check the accuracy of our models, we synthesized and then placed and routed a 5×5 mesh VC router with 2 VCs and 8 buffer slots per VC. Due to process library constraints, we were only able to use FF arrays for buffers. With this implementation, the FF arrays occupied 62% of the overall router area and consumed 18% of the router power at a 20% flit injection rate with uniform random traffic. A compiler-generated SRAM block from the same library would occupy 13% of the router area. The SRAM area results are in line with our models.

2.4 Evaluation

This subsection presents a quantitative comparison of BLESS and VC flow control and discusses design tradeoffs.

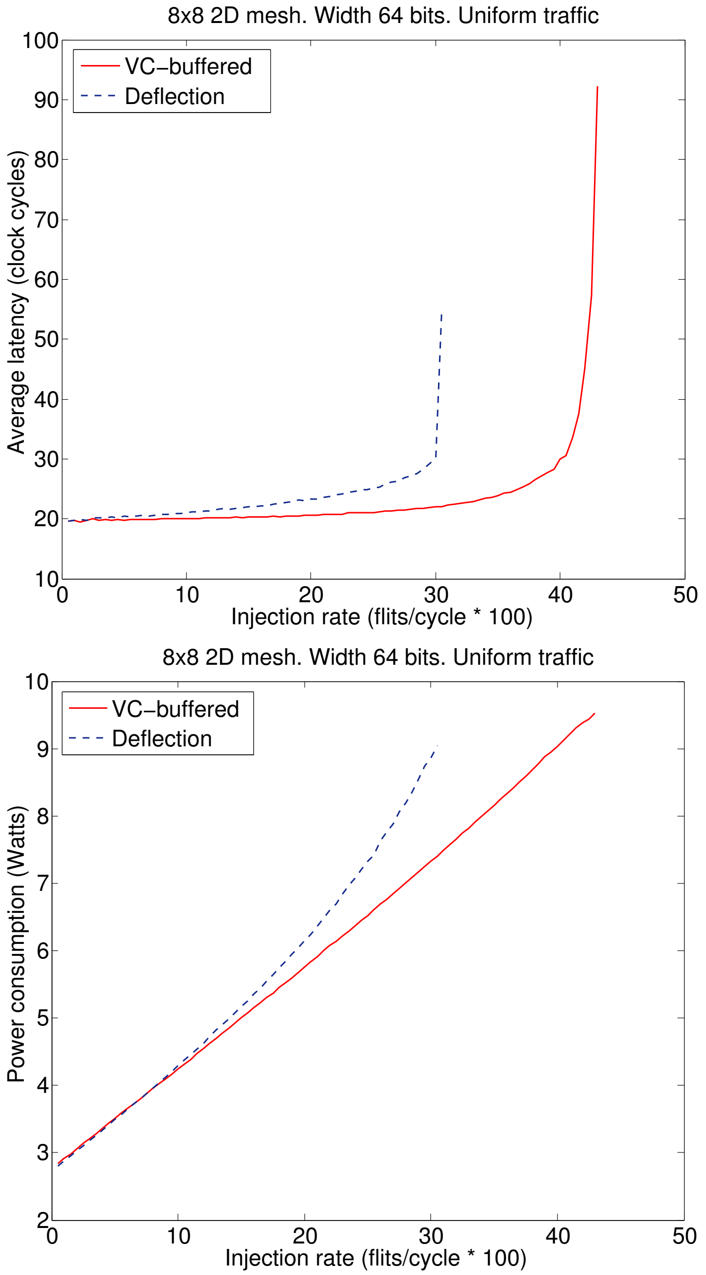


Figure 2.5: Mesh latency and power comparison.

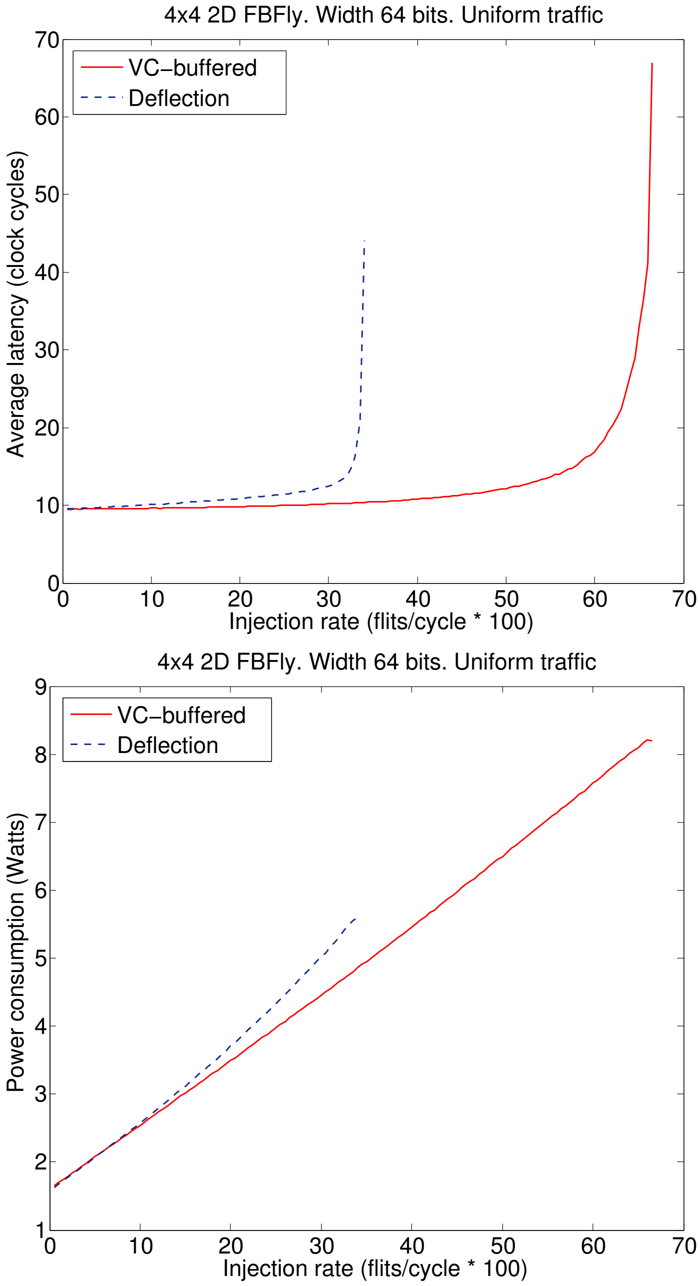


Figure 2.6: FBFly latency and power comparison.

2.4.1 Latency and Throughput

Figure 2.5 and Figure 2.6 present latency and power as a function of flit injection rate. VC routers are optimized for throughput per unit power. They have 6 VCs with 9 buffer slots per VC in the mesh, and 10 slots per VC in the FBFly. The VC network has a 12% lower latency for the mesh on average across injection rates that do not saturate either network, and 8% lower for the FBFly. At a sample 20% flit injection rate, the VC network has a 17% lower latency for the mesh, and 10% lower for the FBFly. Also, the VC network provides a 41% higher throughput for the mesh, and 96% higher for the FBFly. If we average over the set of traffic patterns, the VC network provides a 24% and 15% higher throughput for the mesh and FBFly, respectively.

Deflecting flits increases the average hop count and therefore the average channel activity factor for a given injection rate. The impact of this effect depends on the injection rate and is shown by the increase in dynamic power in the deflection network due to the larger activity factor. Specifically, the deflection network consumes more power than the buffered network for flit injection rates higher than 7% for the mesh and 5% for the FBFly. For lower injection rates, buffer power is higher than the power consumed by deflections. However, even then the deflection network never consumes less power than 98.7% of the VC network power for the mesh, and 98.9% for the FBFly. The higher power in VC networks is mainly due to buffer leakage.

These small power gains are outweighed by the allocator complexity and the other issues discussed in this paper. Furthermore, if a network always operates at such low injection rates, it is likely overdesigned because the datapath width need not be this wide, making the network more expensive than necessary. In other words, a buffered network that is made cheaper by narrowing its datapath is a more cost-efficient choice over a bufferless network with wide channels.

Without empty buffer bypassing, the dynamic buffer power increases significantly, as detailed in Section 2.4.2. In this case, the deflection network consumes less power for flit injection rates lower than 17% for the mesh, and 12.5% for the FBFly. A power–injection rate curve for the mesh is shown in Figure 2.11. This emphasizes the importance of buffer bypassing.

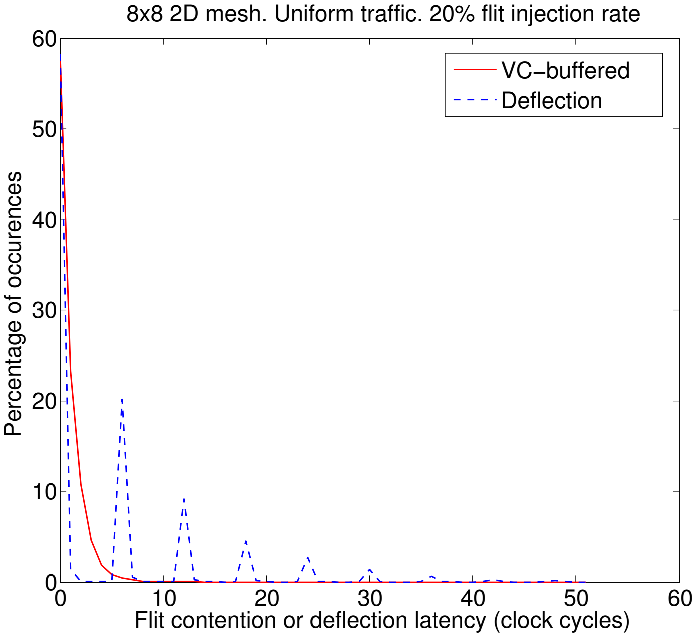


Figure 2.7: Blocking and deflection latency distribution.

Figure 2.7 shows the distribution of cycles that flits spend blocked in buffers or being deflected in the 2D mesh for a 20% injection rate. It was generated by subtracting the corresponding zero-load latency from each flit's measured latency. Since the latency imposed by an extra hop is 3 cycles (2 cycles to traverse a router and 1 to traverse a link), and each deflection adds an even number of hops, the deflection network histogram has spikes every 6 cycles. Thus, this graph also shows the distribution of the number of deflections per flit. In contrast, the VC network has a smooth latency distribution. The average blocking latency for the VC network is 0.75 cycles with a standard deviation of 1.18, while the maximum is 13 cycles. For the deflection network, the average is 4.87 cycles with a standard deviation of 8.09, while the maximum is 108 cycles. Since the average zero-load latency is 19.5 cycles, the VC network has 17% lower latency. These higher latency variations may be crucial to performance: in timeout-based protocols, high latencies will cause spurious retransmissions, and many workloads use synchronization primitives that are constrained by worst-case latency (e.g. barriers).

Figure 2.8 and Figure 2.9 show throughput versus area and power Pareto-optimal

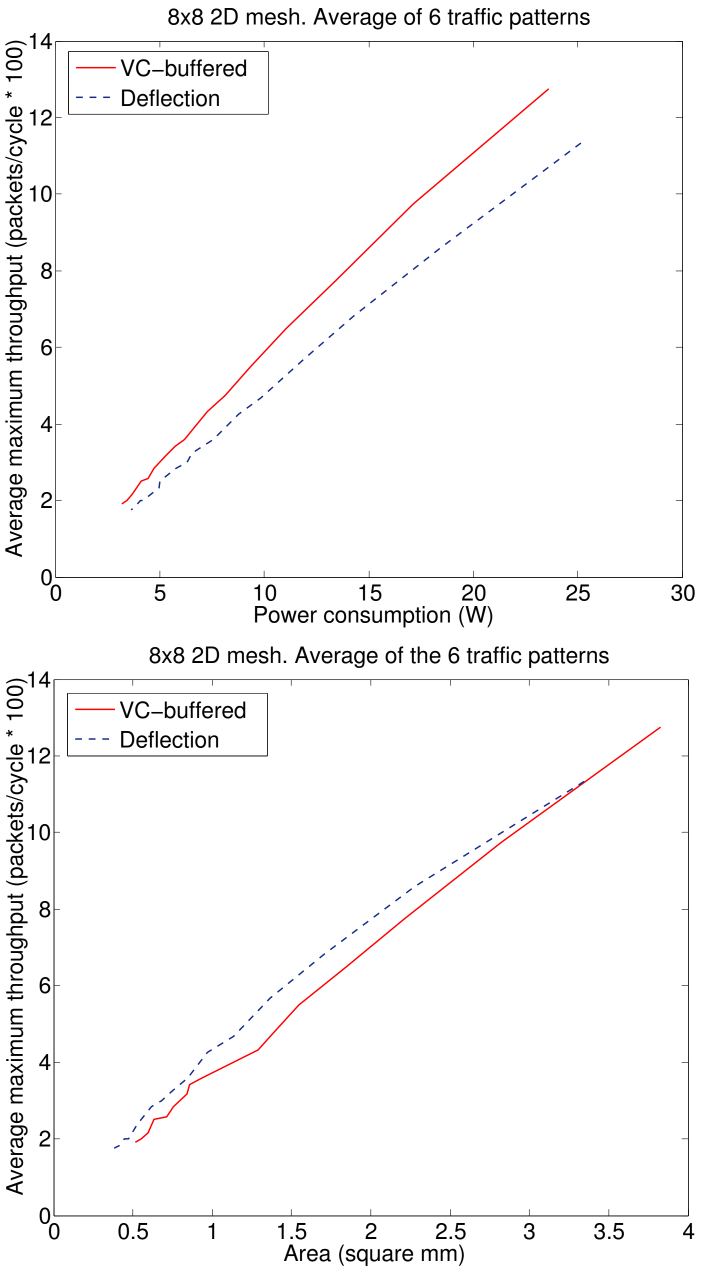


Figure 2.8: Mesh throughput versus power and area Pareto-optimal curves.

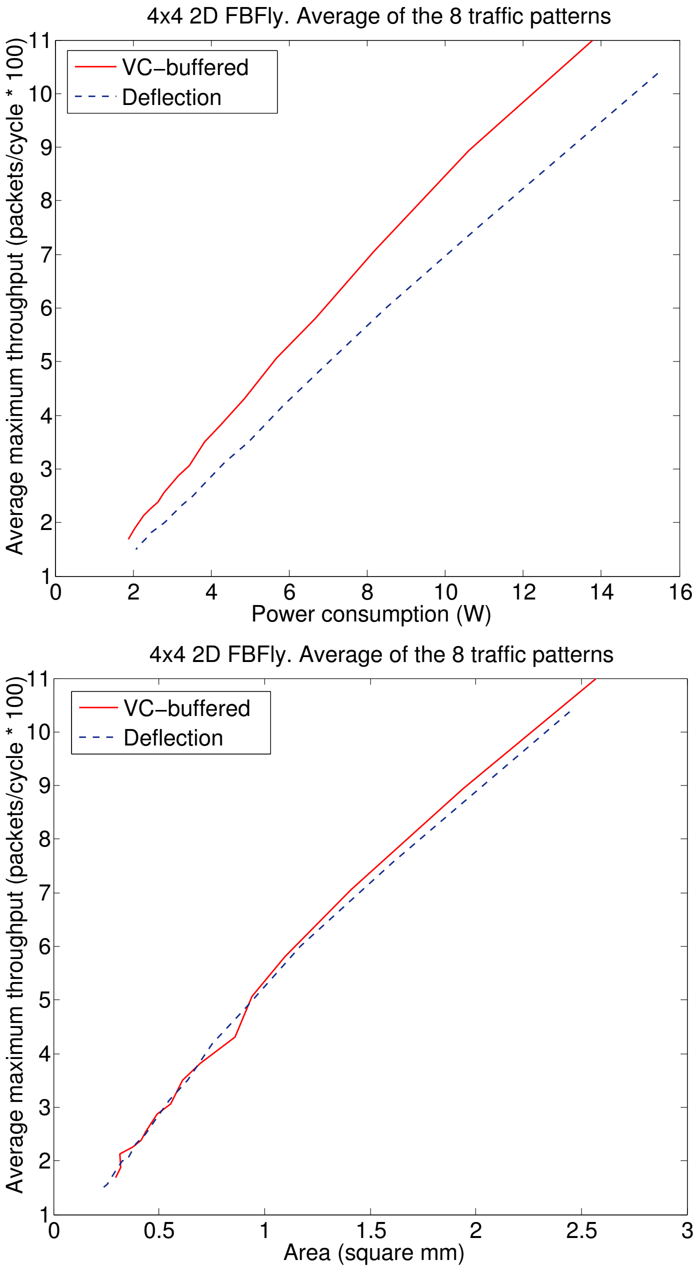


Figure 2.9: FBFly throughput versus power and area Pareto-optimal curves.

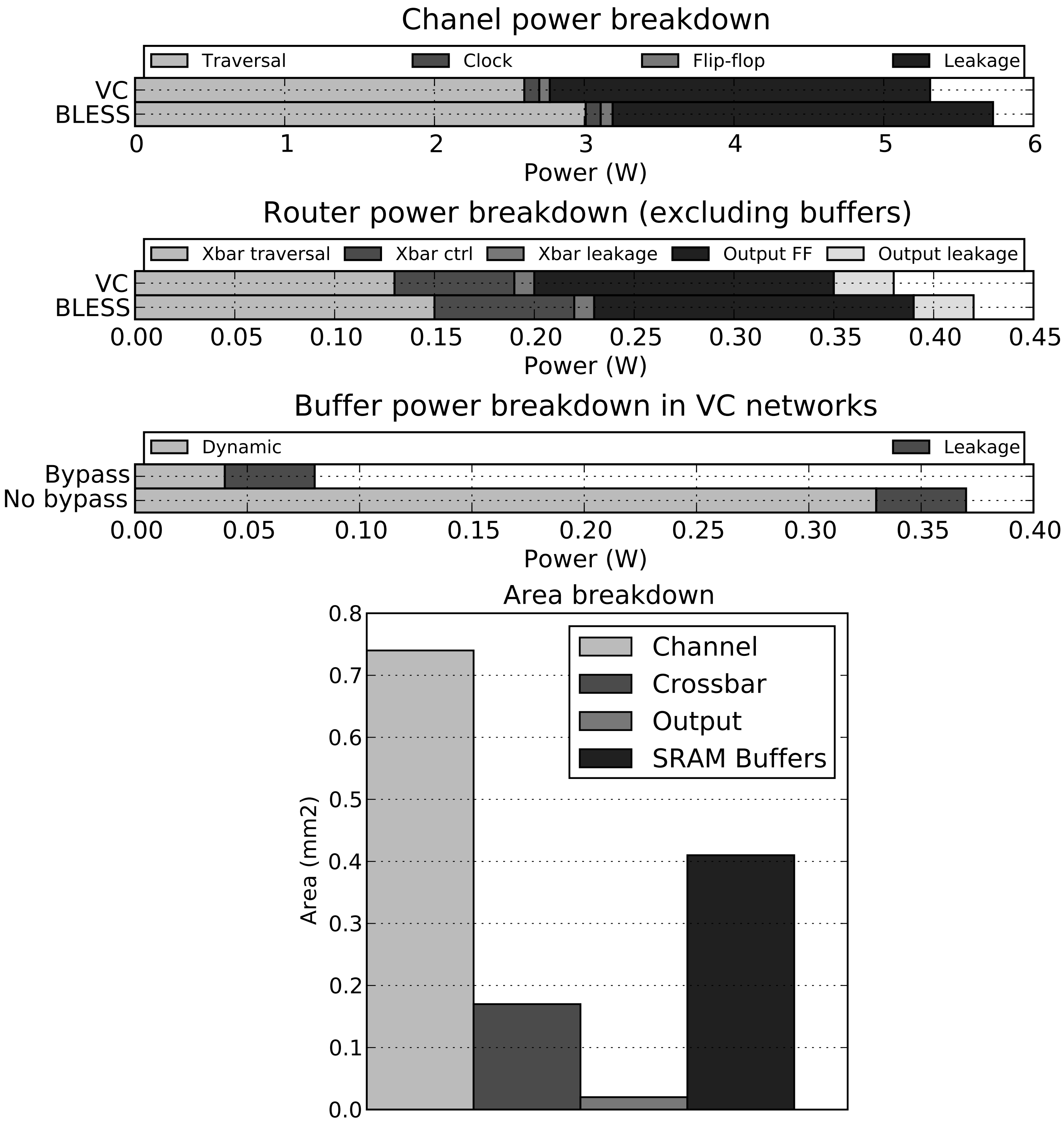


Figure 2.10: Power and area breakdowns for the VC network in a 2D mesh under a 20% flit injection rate with full-swing channels.

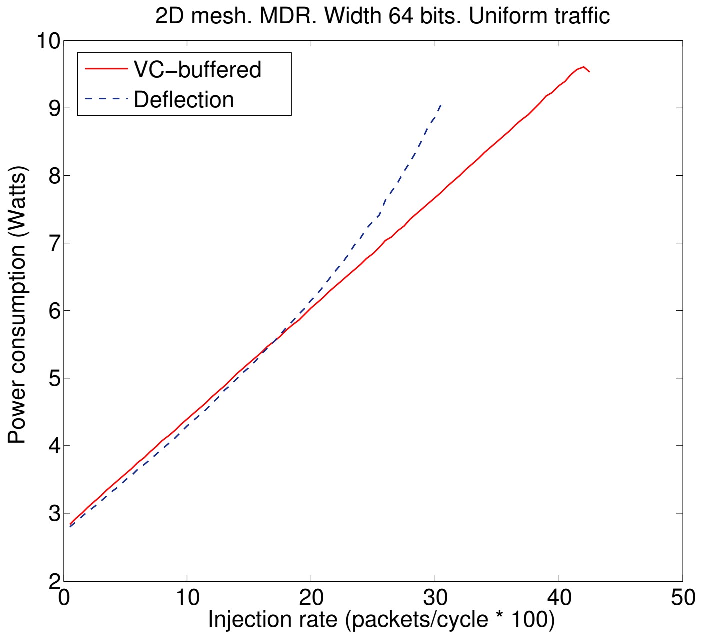


Figure 2.11: Power assuming no empty buffer bypassing.

curves for both networks. Each point of each curve represents the maximum packet throughput achievable by a design of a given area or power. Results are averaged over the set of traffic patterns of each topology. These curves were generated by sweeping the datapath width so that a packet consists of 3 to 18 flits. They illustrate that power or area savings of a network can be traded for a wider datapath, which increases maximum throughput. Thus, points of equal area or power do not indicate an equal datapath width.

As illustrated, the mesh VC network provides 21% more throughput per unit power on average, and requires 19% less power to achieve equal throughput compared to BLESS. The deflection network provides 5% more throughput per unit area due to the buffers occupying 30% of the area, as explained in Section 2.4.2. Consequently, the deflection network requires 6% less area to achieve equal throughput. If the VC network was optimized for area, the buffers would be significantly smaller. The FBFly VC network provides 21% and 3% more throughput per unit power and area respectively. Achieving equal throughput requires 19% less power and 3% less area.

Widening the datapath favors the buffered network. While buffer and channel costs scale linearly with datapath width, crossbar cost scales quadratically. Therefore, taking extra hops becomes more costly. Allocation cost becomes less significant because it is amortized over the datapath width. However, that cost is relatively small, as shown in Section 2.4.2. The quadratic crossbar cost is also the reason the VC FBFly is more area efficient than the deflection network, since widening the datapath to equalize throughput has a larger impact in the area of high-radix routers.

2.4.2 Power and Area Breakdown

Figure 2.10 shows the power and area breakdowns for the VC network in a 2D mesh with a 20% flit injection rate. Each router has 6 VCs, with 9 buffer slots per VC. The buffer cost without bypassing is included. Output clock and FF refer to the pipeline FFs at output ports that drive the long channel wires. Crossbar control is the power for the control wires routed to crossbar crosspoints. Channel traversal refers to the power to traverse the repeated channel segments. Channel clock is the clock wire power to the channel pipeline FFs. Leakage power is included for buffers and channels. For the FF buffers, the given power subsumes clocking, read and write power.

For a 20% flit injection rate, the average channel activity factor is 24.7% on the VC network and 29.3% on the deflection network. The extra 4.6% is due to deflections. This extra power equals 5.5× the buffer access and leakage power. Buffer leakage power is only 0.6% of the overall network power. Removing the buffers saves 30% of the overall network area. The same SRAM buffers without bypassing consume 8.5× the dynamic power with bypassing.

In general, there is no fixed relationship between the number of buffering events in a VC network and the number of deflections in a BLESS network, but intuitively, both increase at roughly the same rate with network utilization, as they depend on contention events. Thus, it is insightful to compare the energy of a buffer read and write with the energy consumed in a deflection. Writing and then reading a single 64-bit flit from and to an input buffer consumes 6.2pJ, while a channel and router traversal takes 20.9pJ (80% of this energy is consumed in the channel). A deflection induces at least 2 extra traversals, causing 42pJ of energy consumption, 6.7× the dynamic energy for buffering the contending flit instead. Therefore, increasing router and channel traversals with deflections is not energy-efficient.

2.4.3 Low-swing Channels

Low-swing channels favor the deflection network because they reduce deflection energy and increase the ratio of the overall power made up by the buffers. With our low-swing channel model, the VC mesh network offers 16% more throughput per unit power than the deflection network for the mesh and 18% more for the FBFly. Also, the VC network offers comparable (1% more) throughput per unit area for the mesh, and 6% more for the FBFly. This increase in area efficiency for the VC network is due to differential signaling, which doubles the channel area, thus reducing the percentage of the total area occupied by the buffers to 19%. Moreover, the deflection network consumes less power for flit injection rates smaller than 11% for the mesh, and 8% for the FBFly. However, compared to the VC network, the power consumed by the deflection network is never less than 98.5% for the mesh and 99% for the FBFly. Figure 2.12 illustrates the results for the mesh. The networks are configured as in Section 2.4.1.

2.4.4 Deadlock and Endpoint Buffers

In a network with a request-reply protocol, destinations might be waiting for replies to their own requests before being able to serve other requests [42]. Those replies might be sent from a distant source or might face heavy contention. Therefore, arriving requests might find the destination's ejection buffers to be full, without a mechanism to prevent or handle this scenario.

Preventing this in the worst case requires ejection buffers able to cover for all possible sources and their maximum outstanding requests. As an example, in a system with 64 processors where each node can have 4 outstanding requests to each of four cache banks (16 requests total), each processor and cache bank needs to buffer 256 requests. This requires a total buffer space of 128KB, whereas an 8×8 2D

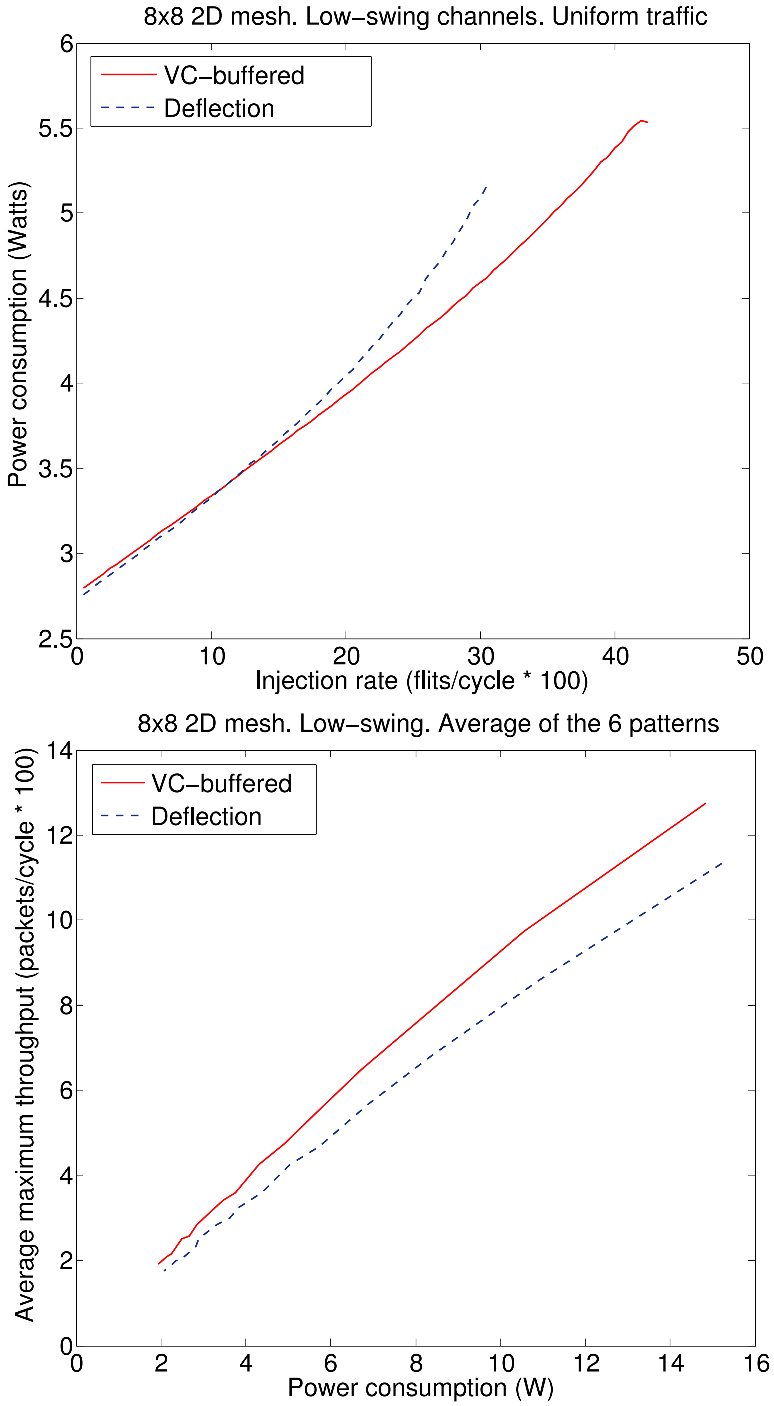


Figure 2.12: Power consumption with varying injection rate and throughput-power Pareto-optimal curves with low-swing channels.

mesh with 2 VCs, each having 8 64-bit buffer slots, needs 20KB. Note that 20KB is only a small fraction of the system-wide SRAM memory in many designs (e.g. chip multiprocessors (CMPs) with multi-megabyte caches). Alternatively, flits that cannot be buffered can be dropped, deflected back to the router, or extra complexity needs to be added, such as feedback to the router so that flits are sent to the ejection port only if there is buffer space for them. This issue becomes more severe with more complex protocols.

2.4.5 Flit Injection

Injection in deflecting flow control requires feedback from the router because at least one output port must be free [83]. However, acquiring this information is problematic, specially if the round-trip distance between the router and the source is more than one clock cycle. Therefore, this extra mechanism not only imposes additional implementation costs, but also risks making inefficient decisions thus wasting energy (if flits are injected at inappropriate times) or reducing performance (if flits are not injected when they could be). Alternatively, flits can be deflected back to the source if there is no free output. However, this causes contention with ejecting flits and costs extra energy. In any case, the injection buffer size may need to be increased to prevent the logic block (e.g. the CPU) from blocking.

2.4.6 Credits

Bufferless networks lack credits and credit channels. Thus, they lack the constraints introduced by them based on the round-trip delay between routers. However, extra congestion-sensing mechanisms need to be developed to route adaptively, as in [89]. These mechanisms represent a tradeoff between performance gain and added cost, and increase complexity.

2.4.7 Process Technology

Our evaluation uses a 32nm high-performance ITRS-based process as a worst case due to its high leakage current. Designs that use other technology libraries may find that the deflection network will consume less power than the VC network for flit injection rates considerably less than 5% and 7%, quoted in Section 2.4.1. By using such libraries, bufferless flow control has an even narrower window in which it provides marginal benefits; this may also increase throughput per unit power and possibly area in favor of the VC network.

To illustrate the other extreme, we use the commercial 45nm low-power library used for synthesis in Section 2.3.1. Its leakage current is negligible. With empty buffer bypassing, the deflection network never consumes less energy than the VC network. Both consume approximately the same amount of power even for very low injection rates. Furthermore, the VC mesh described in Section 2.4.1 provides 21% more throughput per unit power and 10% more throughput per unit area. Therefore, there are no design points that would make the deflection network more efficient in this 45nm low-power library.

Changing process technologies affects the buffer to overall network power cost ratio. Extremely costly buffer implementations would increase this ratio in favor of the bufferless network. In such processes, the bufferless network might be the most efficient choice. However, even the 32nm high-leakage process we used does not fall in this category. In any case, design effort should first be spent on implementing the buffers more efficiently before considering bufferless networks.

2.5 Discussion

Our quantitative evaluation covers the design parameters that are most likely to affect the tradeoffs between buffered and bufferless networks. However, it is infeasible to characterize the full design space quantitatively. In this section we qualitatively discuss the effect of varying additional parameters.

Traffic classes : Systems requiring a large number of traffic classes or VCs may have allocators slower than the age-based allocator of Section 2.3.1. In addition, more traffic classes increase the demand for endpoint buffering discussed in Section 2.4.4. Therefore, the complexity and endpoint buffering demands of bufferless networks increase with the number of traffic classes the system requires. For a single traffic class, we have shown that at least a buffered network with 2 VCs is more efficient than a deflection network. Therefore, additional VCs will favor buffered networks when compared to bufferless networks.

Network size : While network size affects the relevant tradeoffs, smaller networks provide fewer deflection paths. Fewer deflection paths translate into less flexibility when routing flits back to their intended destinations. Network size affects similarly the deflection and buffering probabilities. Thus, none of the two networks is clearly favored by varying network size.

Sub-networks : A deflection network design could be divided into sub-networks to make it more efficient, but the same is true for the VC network. For each sub-network of the deflection network, we can apply our findings to design a similar and more efficient buffered network.

Dropping flow-control : Dropping flow control faces different challenges. For example, its allocators are not constrained to produce a complete matching. However, dropping flow control requires buffering at the sources because sources need to be able to retransmit dropped packets. Dropping, similarly to deflecting, causes flits to traverse extra hops, which translates to energy cost and increased latency. Therefore, the fundamental tradeoff between buffer and extra hop costs remains. However, the number of extra hops in dropping networks is affected by topology and routing more than in deflection networks. In general, dropping flow control may be more or less efficient than deflection flow control, depending on a particular network design.

Self-throttling sources : In our evaluation, traffic sources do not block under any condition (e.g. if a maximum number of outstanding requests is reached). Self-throttling sources are more likely to be blocked when using a deflection network due to its latency distribution, as discussed in Section 2.4.1. Blocking the sources hides the performance inefficiencies of the network by controlling the network load. This favors network-level metrics, but penalizes system performance. For example, in a CMP, blocking the CPUs increases execution time, which is the performance measurable by end users. Realistic system implementations are likely to use self-throttling sources. Therefore, performing an equitable comparison requires taking the number of cycles that sources are blocked into account. Otherwise, the network may operate at an optimal load and therefore satisfy network-level performance metrics, but this will be at the expense of the rest of the system, especially the CPUs, which provide the performance end users care about.

2.6 Summary

This chapter compared state-of-the-art buffered (VC) and deflection (BLESS) flow control schemes. We improve the bufferless network by proposing MDR to reduce deflections. This reduces average latency by 5% in an 8×8 2D mesh, compared to DOR. We also assume efficient SRAM-based buffers that are bypassed if they are empty and there is no contention. Since one deflection induces two extra hops, deflections consume 6.7× the dynamic energy for buffering contending flits instead. Therefore, the deflection network with MDR consumes less power only up to a flit injection rate of 7%. However, the deflection network never consumes less power than 98.7% of that of the VC network. Networks constantly operating at low injection rates are likely overdesigned as they don't need such large datapaths.

In the same 8×8 2D mesh, VC flow control provides a 12% smaller average latency compared to deflection flow control. At a flit injection rate of 20%, the average VC network blocking flit latency is 0.75 cycles with a standard deviation of 1.18, while for the deflection network the average deflection latency is 4.87 cycles with a standard deviation of 8.09. The VC network achieves a 21% higher throughput per unit power. Low-swing channels are favorable for BLESS, making it consume less power for a flit injection rate of up to 11%. However, the deflection network still does not consume less than 98.5% of the VC network power. Furthermore, the BLESS allocator has an 81% larger cycle time than a separable input-first round-robin speculative switch allocator. This increases the allocator timing path compared to a VC router with speculative switch allocation, which is likely to increase the router critical path. Finally, bufferless flow control needs large buffering or extra complexity at network destinations in the presence of a communication protocol.

Our work extends previous research on deflection flow control by performing a comprehensive comparison with buffered flow control. Our main contribution is providing insight and improving the understanding of the issues faced by deflection flow control.

Our results show that unless process constraints lead to excessively costly buffers, the performance, cost and complexity penalties outweigh the potential gains from removing the router buffers (which represent a small fraction of the overall storage capacity present in the chip). Even for the limited operation range where the bufferless network consumes less energy, that energy is negligible (up to 1.5%) and is accompanied by the shortcomings presented in this paper. Therefore, bufferless flow control is not an appropriate response to the rising buffer costs.

Chapter 3

Elastic Buffer Flow Control

Bufferless flow control fails to properly address the need to mitigate buffer cost. However, removing the cost and complexity overhead of buffers would be a major step towards reducing the on-chip network's power budget. The ideal flow control technique would provide buffering in the network such that contention is handled efficiently, but without the associated cost. To approach this ideal flow control as closely as possible, this chapter proposes elastic buffer (EB) flow control which uses pre-existing pipeline flip-flops (FFs) for buffering in lieu of input buffers [75]. Thus, the only added cost is that of the necessary control logic. EB flow control increases the performance over cost efficiency of the network, due to trading area and energy savings from removing the input buffers for wider datapaths. This chapter also analyzes in depth the effect of removing input buffers, and thus virtual channels (VCs), which significantly simplifies router design, but also requires duplicate physical channels for traffic separation and separate congestion-sensing mechanisms for adaptive routing. To reinstate VCs, this chapter also proposes a hybrid EB-VC router for networks which require a large number of traffic classes.

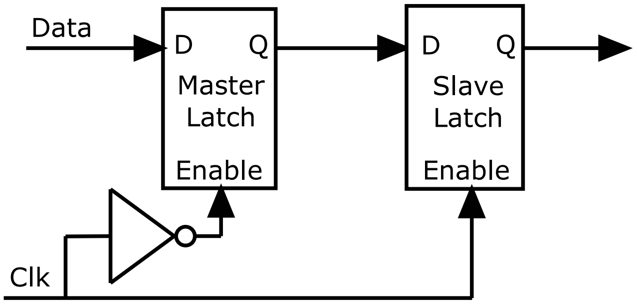


Figure 3.1: A DFF implemented with a master and a slave D latch.

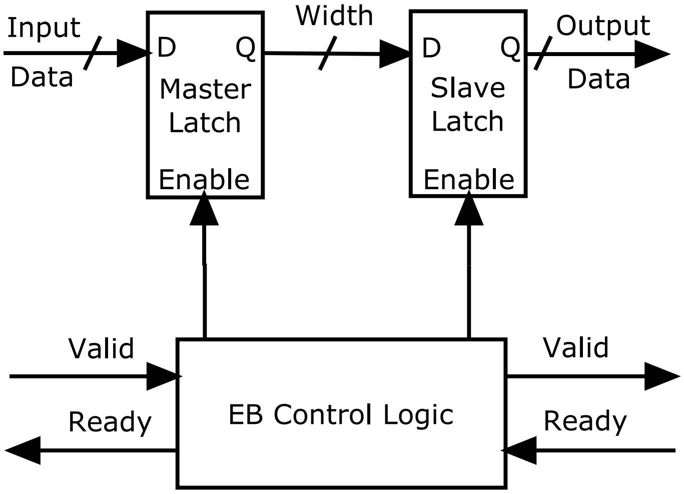
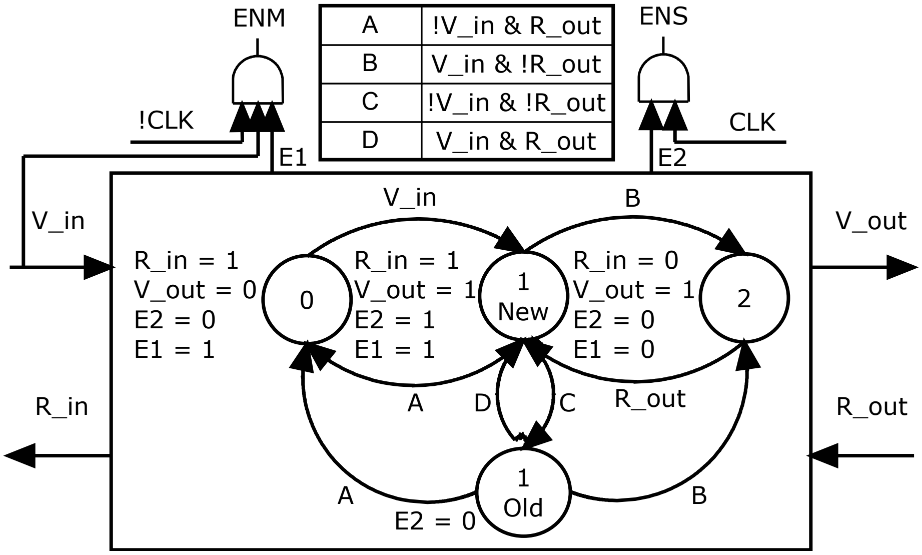


Figure 3.2: Expanded view of the EB control logic as a synchronous FSM for two-slot EBs.



A	!V_in & R_out
B	V_in & !R_out
C	!V_in & !R_out
D	V_in & R_out

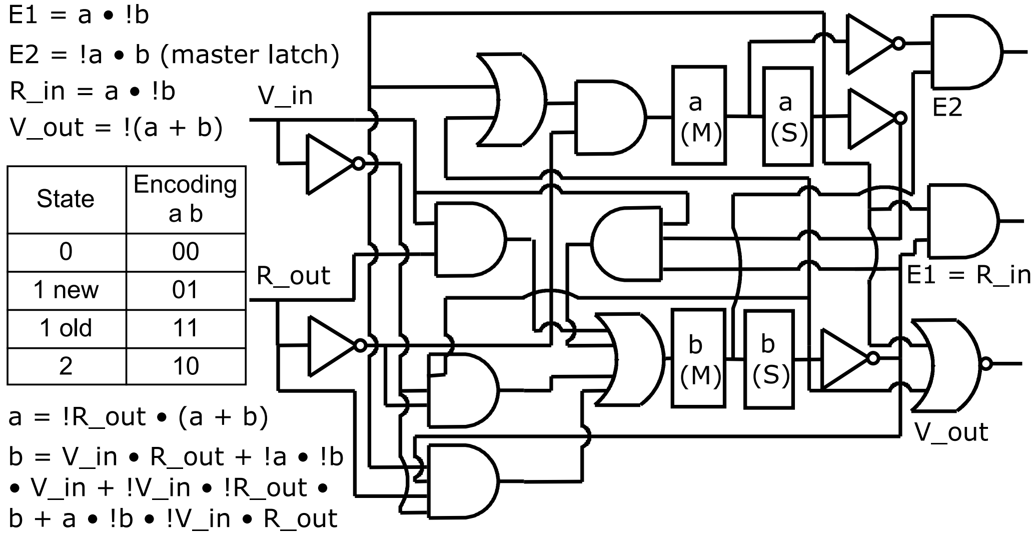
Figure 3.3: Expanded view of the EB control logic as a synchronous FSM for two-slot EBs.

3.1 Elastic Buffer Channels

Figure 3.1 shows a master-slave DFF. The addition of control logic to independently drive each latch enable input allows it to behave as an EB and use each latch independently for buffering. This is illustrated in Figure 3.2. Thus, FFs become EBs and act as two-slot FIFOs.

Consecutive EBs make network channels act as distributed FIFOs. Flits advance to the next EB using a ready-valid handshake. An EB asserts its *ready* signal routed upstream to indicate that it has at least one free storage slot. Furthermore, an EB asserts its *valid* signal routed downstream to indicate that it is driving a valid flit. When ready and valid are asserted between two EBs at a rising clock edge, a flit has advanced. This timing requires at least two storage slots per EB to avoid unnecessary pipeline bubbles.

The finite state machine (FSM) for the control logic is presented in Figure 3.3, and the corresponding implementation is shown in Figure 3.4. The latch enable inputs are qualified by the clock in the same manner as FFs. To avoid latching invalid data into the master latch and thus save power, the master latch enable input is further


E1 = a • !b
E2 = !a • b (master latch)
R_in = a • !b
V_out = !(a + b)
V_in
R_out
a = !R_out • (a + b)
b = V_in • R_out + !a • !b • V_in + !V_in • !R_out • b + a • !b • !V_in • R_out
a (M)
a (S)
b (M)
b (S)
E2
E1 = R_in
V_out

State	Encoding a b
0	00
1 new	01
1 old	11
2	10

Figure 3.4: Logic diagram of the EB control FSM with 2 FFs and 10 gates.

qualified by the incoming valid (V_in). EBs have been verified to operate correctly using RTL models. EBs can be implemented as custom cells to increase efficiency and guarantee that the two latches can never be enabled simultaneously.

The initial state, state *0*, encodes that there are no valid flits stored. Once a flit arrives, the FSM enters state *1 new*, because the newly-arrived flit has been stored into the master latch and thus needs to be moved to the slave latch at the beginning of the next cycle by asserting E2. If this flit doesn't depart and no new flit arrives, the FSM transitions to *1 old* so the flit that resides in the slave latch is not overwritten. In state *2*, the EB is full and thus both latch enables are deasserted. The logic implementation of this FSM uses 2 FFs and 10 logic gates. The EB control logic does not affect the channel datapath. It only requires two additional wires for ready and valid. Since flow control is applied at a flit granularity, the control cost is amortized over the datapath width. For example, the control logic increases the area of a 64-bit channel by only 5%.

All components use the same clock. To avoid penalizing its frequency, we hide the FSM setup time and clock-to-q latency by splitting state FFs into their master and slave latches. Because E2 needs to be stable in the first half of every clock cycle, we use the master latches to generate E2 which allows E2 to stabilize before the end of the previous clock cycle. Since the master latch is enabled only when the clock is low, the first half of the clock cycle suffices to generate and propagate E1 to the master latches in the datapath. Furthermore, to cover the timing overhead of the ready and valid ports, the ready and valid wires are engineered more aggressively for reducing propagation delay [75]. In other words, those wires use larger drivers and repeaters or higher metal layers with wider wires which thus have lower resistance, which reduces propagation delay, but also increases power. The extra power consumption from this is outweighed by the datapath's power consumption.

3.2 Elastic Buffer Routers

This section presents two EB router designs. Later sections describe hybrid EB routers which contain input buffers.

3.2.1 Two-Stage Router

A *two-stage* router is illustrated in Figure 3.5. Only one input and one output are shown in detail. EBs are used between pipeline stages; they serve both as storage and pipelining elements. Therefore, *valid* (V) and *ready* (R) signals are used to facilitate the transfer of flits from one EB to the next inside the router. Ready and valid signals in the second pipeline stage (switch traversal) have to be routed appropriately such that intermediate EBs receive the ready signal of the output that the flit at the head (slave latch) of the intermediate EB is requesting, and output EBs receive the valid output of the intermediate EBs belonging to the input granted by that output's arbiter. The two-stage router emphasizes throughput by reducing cycle time due to pipelining. This router was initially proposed in [77] as the *enhanced two-stage router*, improving on the *baseline two-stage router* of [75].

The input EB receives flits from the channel and drives them to the first pipeline stage. Look-ahead routing (LAR) [32] is used at routers to calculate the output at the next hop for all incoming packets. Therefore, head flits already contain their desired output when entering a router. In parallel with routing, flits send a request to their

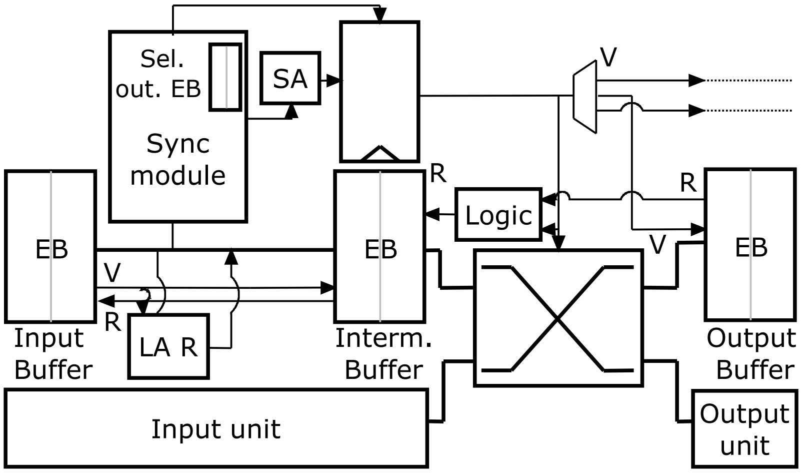


Figure 3.5: A two-stage EB router.

desired output's arbiter. Therefore, switch arbitration (SA) and routing occur in the first pipeline stage. The output arbiters deassert their grants as long as their output EB is non-ready (full). However, flits advance to the intermediate EB using the ready-valid handshake as soon as it has a free storage slot, without waiting for a switch grant. Therefore, each input port can buffer up to four flits in the input and intermediate EBs.

Because flits advance to the intermediate EB as soon as it is ready, extra care must be taken to maintain alignment between flits arriving at the switch traversal stage and their grants. That is because flits may receive a grant from the switch allocator before or after they are stored into the intermediate EB, depending on contention, and may have flits of other packets ahead of them in the intermediate EB traversing the crossbar. Flits may receive a grant while in the intermediate EB if they faced contention and there was space available in the intermediate EB. However, flits may also receive a grant while in the input EB, if the switch allocator is able to satisfy their request before they get a chance to be stored into the intermediate EB.

Alignment is maintained by the synchronization module (sync module). It maintains the selected output ports for the flits stored in the intermediate EB in a separate *selected output EB*. The selected output ports are used to route non-head flits. The output port contained in the slave latch (which contains the oldest of the two entries in the EB) of the selected output EB is always that of the current packet (oldest to have flits remaining in the input or intermediate EB). The selected output EB may also contain in its master latch (youngest entry) the selected output port of the next packet as shown in Figure 3.6 (left). In the worst case, only the tail flit of the current packet remains and there are two more single-flit packets. The most recent to arrive is contained in the intermediate EB, and the other is driven in the router's first stage. In that case, the selected output EB stores the current packet's selected output port in its slave latch and the next packet's selected output port in its master latch. The third packet's selected output port is driven as an input. It will be enqueued when the third packet is enqueued into the intermediate EB, a cycle after the tail flit of the current packet departs.

The synchronization module detects when the tail flit of the current packet is about to depart from the intermediate EB and propagates the selected output port of the next packet to the switch arbiters. Arbiter outputs are registered to shorten the critical path such that it does not extend past the first pipeline stage. Thus, propagating the selected output port is done one cycle in advance to avoid inserting bubbles, as shown in Figure 3.7.

Depending on the next packet's time of arrival, it may either have its head flit stored in the intermediate EB, or driven in the first router stage. In the former case, the next packet's selected output port will be stored in the selected output EB's master latch. In the latter case, the selected output port will be driven as an input to the selected output EB. However, if the next packet's head flit has remained in the first stage for more than one cycle because the intermediate EB is full, the selected output EB is non-empty because there is no intervening packet, and the next packet's selected output port will be stored in the selected output EB. This is illustrated in Figure 3.6 (right). The synchronization module propagates the selected output port of the next packet from the selected output EB's master latch, slave latch, or the input to the master latch, knowing if the intermediate and selected output EBs are non-ready (full) or non-valid (empty).

Flits at the slave latch of each input's intermediate EB traverse the switch if

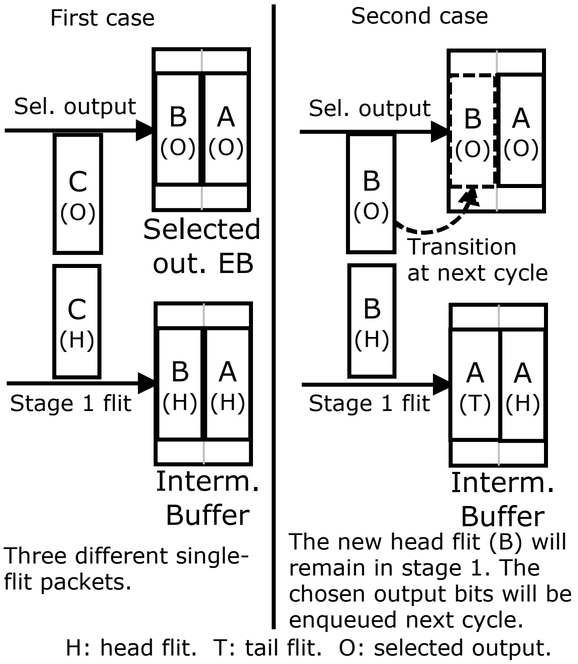


Figure 3.6: Two-stage EB router synchronization module operation.

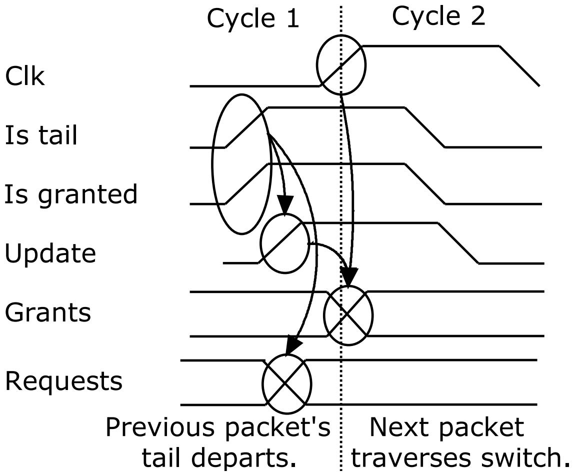

Cycle 1
Cycle 2
Clk
Is tail
Is granted
Update
Grants
Requests
Previous packet's tail departs.
Next packet traverses switch.

Figure 3.7: Updating output grants.

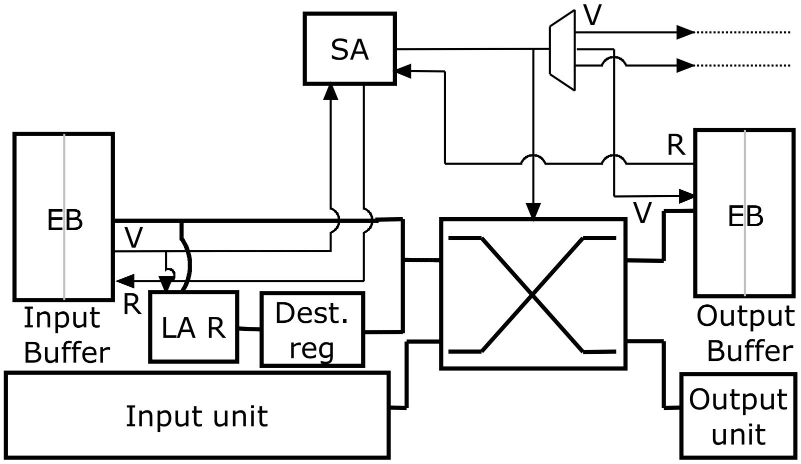


Figure 3.8: A single-stage EB router.

they have a grant from their selected output and that output's EB is ready (non-full). In that case, a *ready* signal is asserted to the intermediate EB to release the flit. In addition, the *valid* output of the intermediate EB is demultiplexed to the proper output EB. Grants are made on packet boundaries and then gated by the selected output EB's *ready* output. When a tail flit is traversing the switch, that input's synchronization logic asserts an *update* signal to all outputs. An output which receives an update signal from the input it is granting has its grant registers clocked at the next clock edge, thus updating the grants driven to the other router components. This is illustrated in Figure 3.7. Arbiters also have their grant register clocking enabled if they are currently granting no input, to assure that grants for newly-arrived packets will be propagated. An extra storage slot in the output EBs is not required because the decision to have the flit traverse the switch is made in the same cycle as the flit would arrive at the output EB.

3.2.2 Single-Cycle Router

A *single-stage* router is shown in Figure 3.8 [77]. It prioritizes latency instead of throughput and avoids pipelining overhead by merging the two stages of the two-stage router. Removing the intermediate EBs also removes the synchronization logic. Incoming flits request their outputs from the arbiters, calculated in their previous

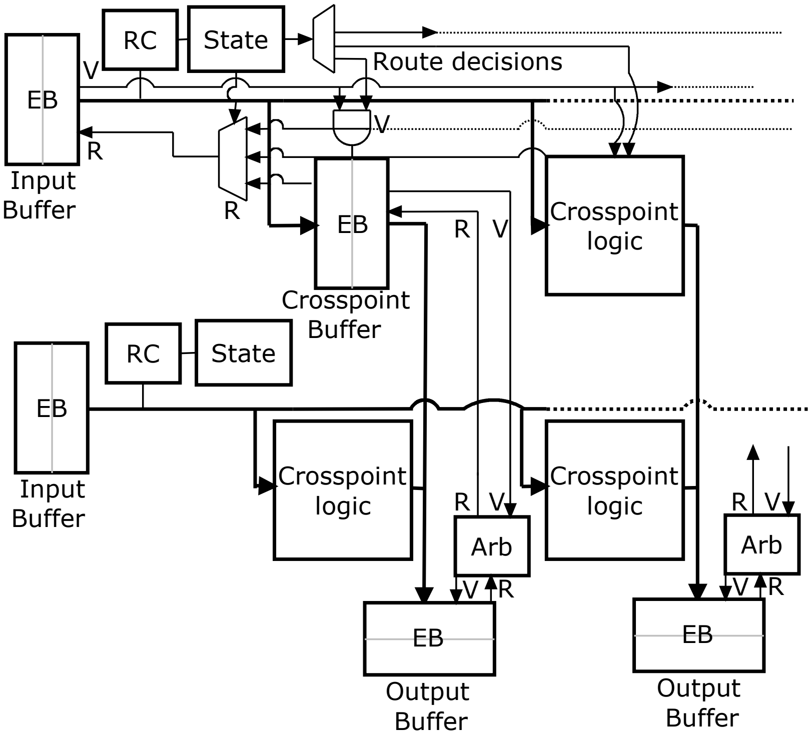


Figure 3.9: Buffered crossbar router. Only the upper left crosspoint is illustrated in detail. Two inputs and two outputs are illustrated.

hop by the look-ahead routing logic for head flits or stored in the destination register for non-head flits. In parallel, LAR also calculates outputs for head flits at their next hop. Only output arbiters with ready (non-full) output EBs can assert grants. Flits traverse the switch and are stored in the output EB in the same cycle that they are granted.

3.2.3 Buffered Crossbar EB Router

In our study we also briefly consider an EB router with EBs at the crosspoints of the switch, as shown in Figure 3.9. Incoming flits are stored at the appropriate crosspoint for their desired output. The ready-valid handshake facilitates movement to and from the crosspoints. This design improves performance if entire packets can be buffered at the crosspoints. Such packet-sized buffers, however, are prohibitively expensive for realistic packet sizes. Also, since a FF is 5.3 times larger than an inverter of twice the driving strength in our technology library, adding an EB at every crosspoint makes the crossbar area dominated by the crosspoints even for a small number of incoming and outgoing wires. This effect becomes worse when trying to design a low-swing crossbar due to doubling the area for differential wiring. Apart from the increased area, a pair of ready and valid wires to and from each crosspoint EB is required, causing an increased power consumption. Therefore, even though the buffered crossbar router can be beneficial for short packets, we do not consider a buffered crossbar further.

3.3 Deadlock Avoidance and Traffic Classes

This section discusses traffic separation techniques in EB networks due to the lack of VCs and motivates the hybrid EB-VC design described in Section 3.4.1.

3.3.1 The Interleaving Deadlock

With the removal of VCs, tail flits may get blocked behind head flits of other packets due to the FIFO nature of EB channels. The same is true for wormhole networks because their input buffers are single-lane FIFOs. Therefore, packet interleaving is infeasible in both networks [13]. This is illustrated in the example of Figure 3.10. A head flit of a new packet requires a free register to store its desired output (dependency A). However, a register cannot be released until a tail flit arrives (dependency C). On the other hand, no tail flit may bypass the blocked head flit to arrive at the input (dependency B). Disabling packet interleaving does not degrade network performance assuming that sources transmit flits of the same packet contiguously. To the contrary, since packets are considered delivered once their tails arrive, this may decrease average packet latency. However, disabling packet interleaving may raise fairness issues in the presence of long packets. Interleaving of packets in different VCs is allowed for the hybrid EB-VC router described in Section 3.4.1, similarly to VC routers.

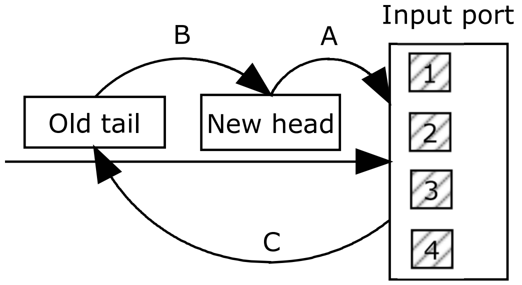


Figure 3.10: The interleaving deadlock cyclic dependency.

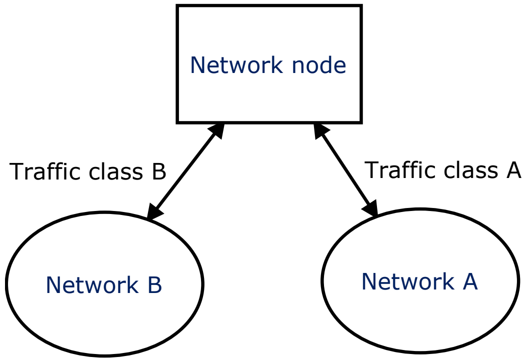


Figure 3.11: Using one subnetwork per traffic class.

3.3.2 Duplicating Physical Channels

Duplicate physical channels define traffic classes in the same way as duplicate VCs. All relevant literature on VCs is applicable to prevent cycling dependencies within or across traffic classes. Network destinations have to be able to eject traffic from all classes required to prevent protocol deadlocks [24, 13].

Duplicate channels can be efficiently provided by instantiating separate physical subnetworks, as shown in Figure 3.11. Each subnetwork carries traffic from a single traffic class. In the general case, the subnetworks are independent; network endpoints connect to all of them. Therefore, network endpoints must be able to eject from all subnetworks and inject to any one subnetwork during any clock cycle. Hierarchical approaches are possible to reduce the radix of the endpoints. In this scheme, network endpoints would connect to concentrators which essentially would multiplex and demultiplex a fixed number of subnetworks into and from a single endpoint module port.

To facilitate packets switching traffic classes, channels can connect parts of some subnetworks to other subnetworks. Such a design should be fully customized for a specific topology and routing algorithm to allow only the required transitions between traffic classes. Such an example is described in Section 3.5.2.

With duplicate subnetworks, each subnetwork should have a reduced datapath width such that the design with the duplicate subnetworks has the same energy or area cost as a single network. This ensures a fair comparison because it shows the performance per cost efficiency of each design. Narrowing the datapath results in a more cost efficient network because of the crossbar's quadratic cost relationship with the number of incoming and outgoing wires. For instance, doubling the number of subnetworks and halfing the datapath width will result in a net reduction in area because both the input side and the output side of the crossbar now have half the length, therefore the crossbar now occupies a quarter of the area. This assumes that crossbar area is dictated by wire pitch, which is most often the case for not overly narrow datapaths. Therefore, using multiple subnetworks both provides traffic classes and increases cost efficiency.

Cost efficiency is increased by duplicating subnetworks up to a certain number of subnetworks. Above that number, the radix of the network endpoints becomes significant, or the datapath becomes narrow enough such that control overhead makes the design inefficient. The number of subnetworks that produce diminishing returns due to the effects mentioned above depends heavily on implementation technology, endpoint design and network constraints and therefore needs to evaluated in specific chip designs and implementation technologies. Furthermore, narrowing the datapath increases serialization latency which can be an important consideration. However, narrowing the datapath may reduce router cycle time if the switch is in the critical path. Finally, care must be taken to load the duplicate subnetworks equally to avoid idling network resources while others are oversubscribed. Load balancing requires properly assigning traffic classes to subnetworks, or adjusting the datapath width of each subnetwork individually. However, balancing load becomes a harder problem as the number of traffic classes increases.

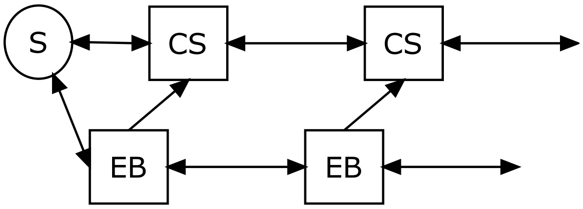


Figure 3.12: Hybrid EB-CS network.

Other approaches are possible but are less cost efficient [75]. An alternative is duplicating physical channels between routers but multiplexing them into the same router port. To maintain non-interleaving, multiplexers and demultiplexers must select on a per-packet basis. Moreover, routers must duplicate output EBs to prevent flits of different classes from interacting. However, this option provides a minimal benefit for a disproportional channel cost increase. Also, duplicating channels and switch ports increases the crossbar cost quadratically, outweighing the performance benefits. Recent work has also proposed duplicating just the switches in the routers, but not channels [33].

3.3.3 Using Circuit-Switching in EB Networks for Traffic Separation

An alternative approach to providing an arbitrary number of traffic classes in EB networks is illustrated in Figure 3.12. With this scheme, there is an EB and a circuit-switched (CS) network. Traffic sources inject traffic into the EB network. Packets that reach their destinations through the EB network cause an acknowledgment to be sent back to the source. All kinds of acknowledgments and other control information use separate narrow control channels. The EB network functions as already described, with the exception that a head flit that is blocked for more than a predefined number of cycles at a router input is drained from the channel and dropped. A negative acknowledgment is sent back. This causes the first router in the path to transmit a request to establish a circuit through the CS network from the source to the destination. When this occurs, the source is notified and retransmits the packet through the CS network. The source then erases it from its local injection buffer. CS routers tear down the circuit when they transmit the tail flit downstream.

When packets cause requests to be transmitted in the CS network, they retain their traffic class and priority level. Routers satisfy the higher priority request. If a lower priority circuit has been established at a router but the acknowledgment to the source has not passed through that router, no data can be in transit. Therefore, the low priority circuit can be preempted, causing negative acknowledgments to be sent both ways to tear down the remaining of that circuit. Otherwise, a control signal can be sent to the source to stop transmitting the low priority packet and submit the rest in the future as a separate packet. This way, while low priority flits interact with high priority flits, the latter still have a significant advantage over the former.

This scheme suffers from high latencies under contention, because dropped packets have to be retransmitted and wait to establish a circuit. Compared to the hybrid EB-VC network discussed in Section 3.4.1, retransmitting packets costs more than simply buffering them locally in routers. Furthermore, this EB-CS hybrid approach uses two subnetworks but doesn't distribute the load evenly. However, adjusting datapath widths may alleviate this concern given prior knowledge of the load imposed to the network. Setting the number of cycles a head flit needs to be blocked for until its dropped to a low number causes high latencies, while setting it to a high number causes underutilization of the CS network and performance very similar to a single EB subnetwork. We have found this scheme to be less efficient than the EB-VC hybrid approach. However, we note that this scheme may become more attractive if technology library or other constraints increase the buffer to overall network power ratio (because that would make the EB-VC router more expensive), or in network topologies with small zero-load latencies because then the extra latency may not be important.

3.4 Hybrid EB Routers

This section presents two hybrid EB routers. Both of these designs have input buffers. Firstly, the EB-VC router reinstates buffers at inputs as well as VCs and is used to

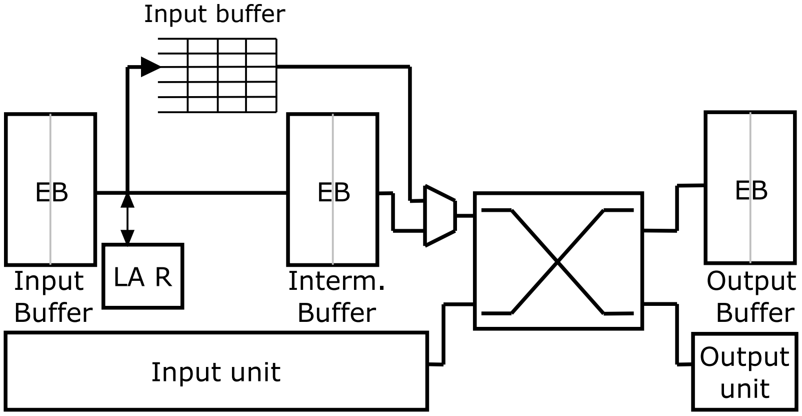


Figure 3.13: Hybrid EB-VC router.

provide an arbitrary number of traffic classes without duplicating physical channels beyond the point of diminishing returns, as described in Section 3.3.2. Finally, the EB-wormhole router is used to compare against EB routers to illustrate the importance of removing input buffers in performance over cost efficiency increase, since wormhole routers do not have VCs and thus their complexity is similar to EB routers.

3.4.1 Hybrid EB-VC Routers

To provide an arbitrary number of traffic classes, we add buffers at each input to EB routers. Each input buffer contains an independent FIFO for each VC. We base this hybrid EB-VC design, shown in Figure 3.13, on the two-stage EB router, because the buffer introduces logic complexity which necessitates pipelining. Adding an input buffer to the single-stage EB router would extend the critical path significantly due to the control logic and data output timing overhead of buffers.

The buffer drains flits that are blocked in the input EB for a predefined number of cycles, *BL CYCL*. *BL CYCL* is a design-time parameter that affects how often the buffer is used. Each input has a counter per traffic class. A flit blocked in the input EB will cause the counter of that traffic class to be incremented by one. A flit leaving the input EB to be stored into the intermediate EB causes the same counter to be decremented by one. Counters are also decremented if the flit at the head (slave latch) of the input EB is from a different class. If a counter is equal or greater than

$BL\ CYCL$, flits of that class leave the input EB and are stored into the buffer. When the counter is incremented and becomes equal to $BL\ CYCL$, it is further incremented by a predefined value. This creates a bias towards draining. Otherwise, interleaved flits from other classes will cause the counter of the class under contention to be decremented. Therefore, when the next flit of the class under contention arrives, it will be blocked for a few more cycles. In our implementation, counters are incremented by the number of traffic classes as soon as they are incremented and become equal to $BL\ CYCL$. Therefore, incrementing a counter from a value of $BL\ CYCL - 1$ to $BL\ CYCL$ will result in an end value of $BL\ CYCL + N_{TC}$, where N_{TC} is the number of traffic classes. This efficiently handles the case of having a flit from every other class interleaved before the next flit from the class under contention. Otherwise, this scenario would cause the counter to be decremented while flits of other classes arrived at the input, thus causing unnecessary blocking latency when the next flit of the class under contention arrives, because the counter has to be incremented again.

To prevent buffer overflow, a backpressure signal for each channel is routed upstream via dedicated wires. The backpressure signal instructs the traffic source upstream to pause sending flits of the class (or VC) under congestion. That occurs as soon as the free buffer slots barely suffice to drain the channel from flits of the class under congestion in the worst case. The worst case is all flits in the upstream channel belonging to that class, with more to be transmitted until the congestion signal arrives. Therefore, the buffer always has enough free slots to drain incoming flits of the class under congestion, similar to a skid buffer. Once the number of free slots for a class increases above the threshold, a signal is sent upstream to resume transmission of that class. Since only one flit can arrive to an input during any given cycle, no more than one traffic class will be required to submit a backpressume signal upstream during any cycle, and thus backpressure signals refer to only one traffic class. This is similar to an Xon/Xoff scheme per traffic class [86].

Flits bid for the switch from the intermediate EB and the buffer. Therefore, a switch allocator is required because each input may request multiple outputs. The allocator ignores requests to outputs from flits beloging to traffic classes that have received a backpressure signal. A VC allocator is required if flits are permitted to transition to different classes, or if multiple VCs are provided per class.

Routers cannot predict at transmission time if flits will be stored into the buffer of the downstream router. Thus, using credit flow control and consuming a credit when transmitting a flit would be pessimistic because it would assume that all flits would use the buffer. Therefore, hybrid EB-VC routers do not use credits but only use the ready-valid handshake, which is required for EBs.

The presence of buffers makes the occupied area almost identical to VC routers for equally-sized buffers. Moreover, the hybrid routers need an allocator and extra control channels for the backpressure signal. Therefore, the hybrid EB-VC router has comparable complexity to VC routers, especially if a VC allocator is required.

Flits may momentarily block flits of other traffic classes in the EB channels. Therefore, while packet priorities can be provided by the hybrid EB-VC router, in the worst case there can be considerable interaction with lower-priority packets.

The worst case blocking latency in the hybrid EB-VC network for a flit which just traversed the switch occurs if the flit at the head of the router's output EB (slave latch) and every flit ahead of it blocks for *BL CYCL* cycles. For a single hop, the worst-case blocking latency solely because of flits of other classes is $((EBs+2)\times 2-1)\times(BL\ CYCL)$, where “EBs” is the number of EBs in the channel. This takes into account the downstream router's input EB and the output EB the flit is currently in. This theoretical worst case is extremely rare, thus not affecting performance in practice.

The hybrid EB-VC router consumes almost the same dynamic energy as the two-stage EB router in the common case. EBs remain the primary means of storing flits. Buffers are only used to to alleviate head-of-line blocking and resolve situations which would otherwise result in a deadlock. These situations form when packets of different traffic classes contend in the EB channels and form either cyclic dependencies in the network or at the endpoints due to the communication protocol [86, 24]. The buffer is used to resolve those dependencies by draining flits. The choice of *BL CYCL* represents a tradeoff. With a low value the buffer is used similarly to VC networks. With a high *BL CYCL* value the buffer is used rarely, resembling EB networks.

A proper choice of *BL CYCL* makes the hybrid EB-VC router provide higher throughput per unit power than VC routers because buffers are not used in the common case as well as than EB routers because head-of-line blocking is alleviated. However, the buffers still extend the timing path of the first pipeline stage and impose an area overhead compared to EB routers. Also, in certain buffer implementations or high-performance technology libraries, leakage power in the buffers may become a concern.

The hybrid EB-VC router provides an arbitrary number of traffic classes. If only a few traffic classes are needed, duplicating subnetworks is more efficient as explained in Section 3.3.2; duplicating subnetworks also avoids buffer area and timing overhead as well as does not allow interaction between flits from different classes in EB channels. However, the hybrid EB-VC design can be combined with duplicate subnetworks to alleviate head-of-line blocking in the common mode of operation where input buffers are not used. Choosing among the hybrid design, duplicate subnetworks or a combination thereof requires an evaluation of the different performance and cost metrics specific to each network design and implementation technology.

3.4.2 Hybrid EB-Wormhole Routers

Compared to VC routers, EB routers are significantly simpler and can therefore operate at higher clock frequencies and have a lower cost, as shown in Section 3.7. However, EB routers also shorten cycle time and reduce cost due to removing the input buffers. To better understand the relative significance of these two factors, we compare EB routers with wormhole routers. Wormhole routers also lack VCs and therefore are comparably complex to EB routers. Thus, comparing them with EB routers isolates the contributions of removing the input buffers.

The wormhole router implemented for this study is shown in Figure 3.14; we use a two-stage pipeline in order to avoid excessive critical path delay. Wormhole routers typically use credit-based flow control and thus require input FIFOs to be at least deep enough to cover the credit round-trip and processing delay in order to avoid channel under-utilization [8]. Credit handling logic is placed at each output port and drives a single-bit signal to the output's arbiter, inhibiting grants if no

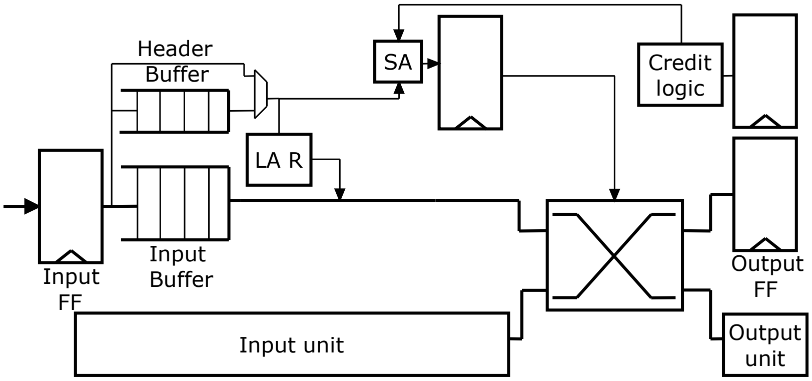


Figure 3.14: The wormhole router.

credits are available. For energy efficiency reasons, our router design does not have an intermediate pipeline register between the input buffer and the switch[1]. Switch arbitration in a given cycle sets up the switch control signals for the next cycle; hence, when a tail flit is at the head of the buffer and begins switch traversal, the control signals driving the switch arbitration logic must be generated from the head flit behind it. In order to be able to handle this situation without adding either a second read port to the input buffer or a stall cycle between successive packets, we track routing information for each packet in a separate header buffer. This buffer's small width allows us to choose an implementation that is optimized for fast read-out at the cost of slightly increased energy, and to thus minimize the delay going into the switch arbiter logic. As with the baseline EB designs, we include input- and output-side register stages to ensure that the full clock cycle is available to the adjacent channel segments for signal propagation. Without such registers, the FIFO write delay–caused by address decoding, internal fan-out to the individual storage elements, and their respective setup time requirements–would have to be borrowed from the preceding channel segment, and both the FIFO's read delay and the switch traversal delay would have to be borrowed from the subsequent one; this reduces the time available for signal propagation and thus the maximum channel length. Since

[1] Note that even with such a register, allowing the flit buffer to be fully bypassed adversely affects timing, as the bypass logic depends on the outcome of switch arbitration, which in turn is not available until late in the cycle.

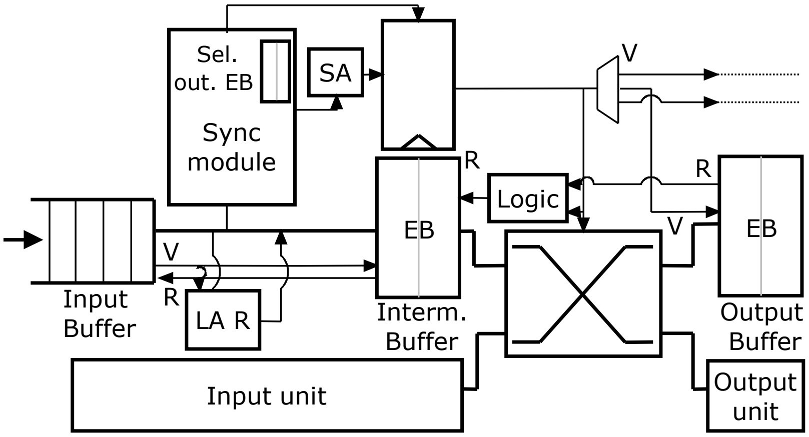


Figure 3.15: Two-stage hybrid EB-wormhole router.

the transfer of flits from the input register to the FIFO is overlapped with switch allocation, it does not incur additional pipeline delay.

The hybrid EB-wormhole router designs we propose in this study replace the input EB with a FIFO. This two-stage hybrid router design is illustrated in Figure 3.15. Credit-based flow control is replaced with the ready-valid handshake of EB channels. Therefore, the input FIFO also interfaces using the ready-valid handshake, and the FIFO depth is not restricted by channel length. The illustrated design has an intermediate EB and synchronization logic functioning in the same way as the two-stage EB router described in Section 3.2.1. While the intermediate EB is not required, removing it would require the router to read from FIFO locations other than the head or have a separate structure, as described in the previous paragraph. This was found to be more costly than the intermediate EB with the synchronization logic. Therefore, the total number of flits that the hybrid router can store at each input is given by the number of entries in the FIFO plus two for the intermediate EB. Unlike the baseline EB and wormhole designs, this hybrid design writes directly from the channel into the input buffer, reducing energy consumption at the cost of additional timing pressure on the preceding channel segment. This choice is based on preserving the zero-load latency through the hybrid router to two cycles. While writing directly from the channel into the input buffer favors the hybrid router in terms of energy and cost because we ignore the extra time borrowed from the channel, we show in

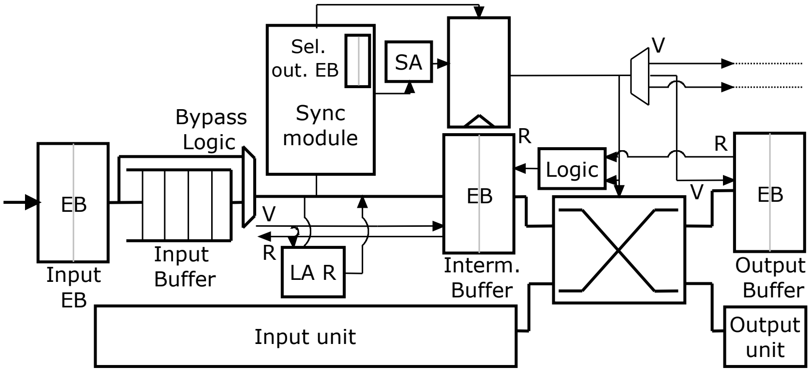


Figure 3.16: Two-stage hybrid EB-wormhole router with input EBs.

Section 3.7.5 that at least one of the two baseline EB routers is still preferable to it in each case.

Further hybrid designs are possible. For example, the input EB can be preserved in addition to the FIFO, as illustrated in Figure 3.16. This makes the FIFO smaller for the same amount of total input buffering. It also enables flits that do not face contention to bypass the FIFO and be stored directly to the intermediate EB if the FIFO is empty. However, if the FIFO cannot be bypassed, additional energy is expended for traversing the extra input EB and multiplexing logic. On the other hand, if the FIFO is bypassed, it may be small enough to have a comparable energy overhead with the input EB and the bypassing logic complexity. Buffer bypassing is focused at networks which operate under lower loads and since our study does not focus on low loads, we do not use buffer bypassing in this work for the wormhole and hybrid routers. However, buffer bypassing should be considered for lightly-loaded networks depending on their buffer sizes.

To fully explore the design space, we also briefly consider hybrid routers based on the single-stage router.

3.5 Congestion Sensing

With the removal of VCs and input buffers, EB networks no longer have credits which are widely used to sense congestion downstream. Congestion sensing is used by adaptive routing algorithms which route around congested areas. This distributes load more evenly and makes better use of network resources, thus reducing latency and increasing throughput. To support adaptive routing in EB networks, extra mechanisms have to be developed for measuring congestion. This section describes such mechanisms and then outlines an application of adaptive routing using those mechanisms as well as duplicate physical channels.

3.5.1 Congestion Sensing Mechanism

Congestion sensing in EB networks must measure channel occupancy to estimate congestion because credits are not available. To further alleviate contention, the mechanism must take into account packets that have been routed to an output but are still bidding for the switch. Otherwise, all inputs wanting to send to an unblocked output will be considered a low congestion scenario. Moreover, the mechanism must adapt quickly to current network status.

We evaluate five different congestion metrics for EB networks that can be used instead of the credit counts used by VC networks: blocked cycles, blocked ratio, output occupancy, channel occupancy, and channel delay.

Blocked Cycles is a running average of the number of cycles an output is blocked. Once an output is blocked, a counter keeps track of the number of clock cycles until it unblocks. At that point it updates the running average. The new value for running averages is calculated as $newvalue = 0.3 \times oldvalue + 0.7 \times newsample$. These factors were chosen for best performance after evaluating the alternatives.

Blocked Ratio is the ratio of the number of cycles an output has been blocked, divided by its unblocked cycles. We calculate this for the 20 most recent clock cycles.

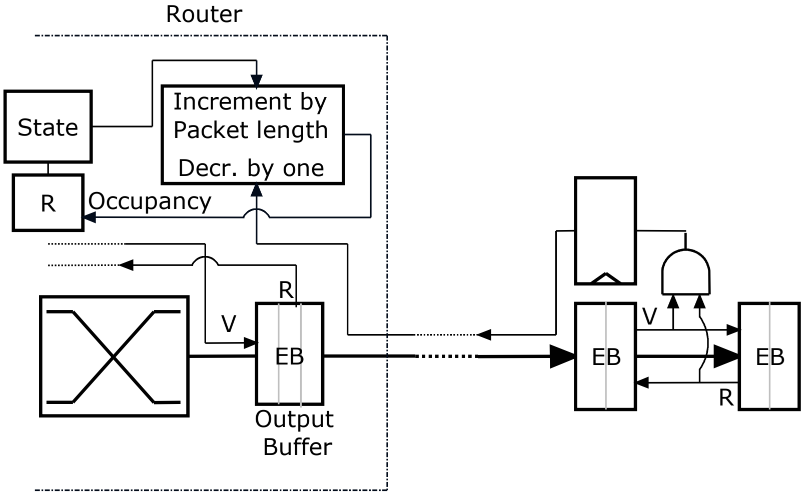


Figure 3.17: Measuring output occupancy in EB networks.

Output Occupancy counts the flits currently in a segment of each output channel, called the observation region. When the head flit of a packet is routed, the occupancy counter for its output is incremented by the packet length in flits. Output counters are decremented by one for each flit leaving the observation region, the output channel segment for which we keep track of flits in transit.

The logic for tracking the number of flits in the observation region is shown in Figure 3.17. Flits leaving the observation region are detected using an AND gate whose inputs are the *ready* and *valid* signals between the last EB of the region and the next EB. The AND gate's output propagation delay affects the reaction time to congestion. Similarly to the *ready* and *valid* wires, that wire can also be engineered more aggressively for delay. For this study, the observation region is the same length as the shortest network channel. The AND gate's output wire propagation delay is half of that channel's delay in cycles, rounded up.

Channel Occupancy is similar to output occupancy, except that an output's counter is incremented when a flit arrives at the output EB instead of when its packet is routed to that output.

Channel Delay measures the average number of cycles needed by flits to leave the

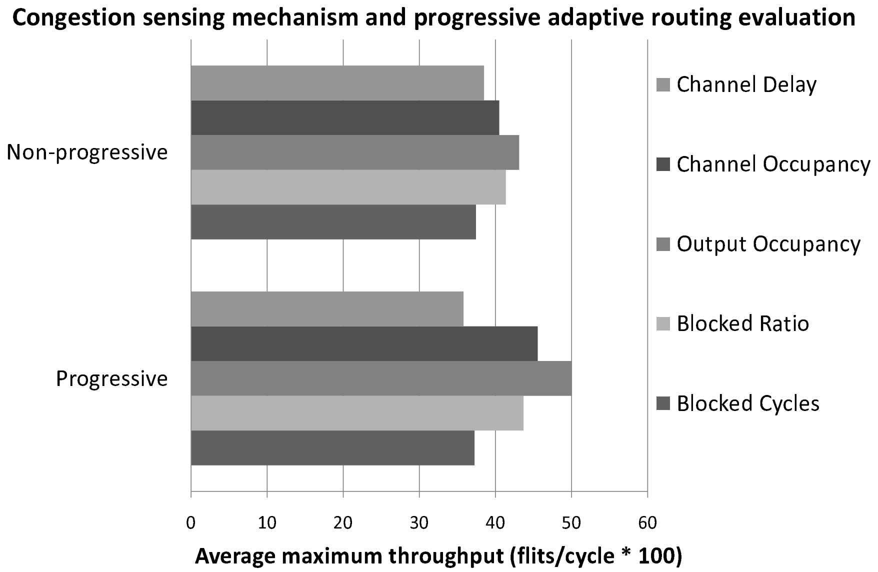
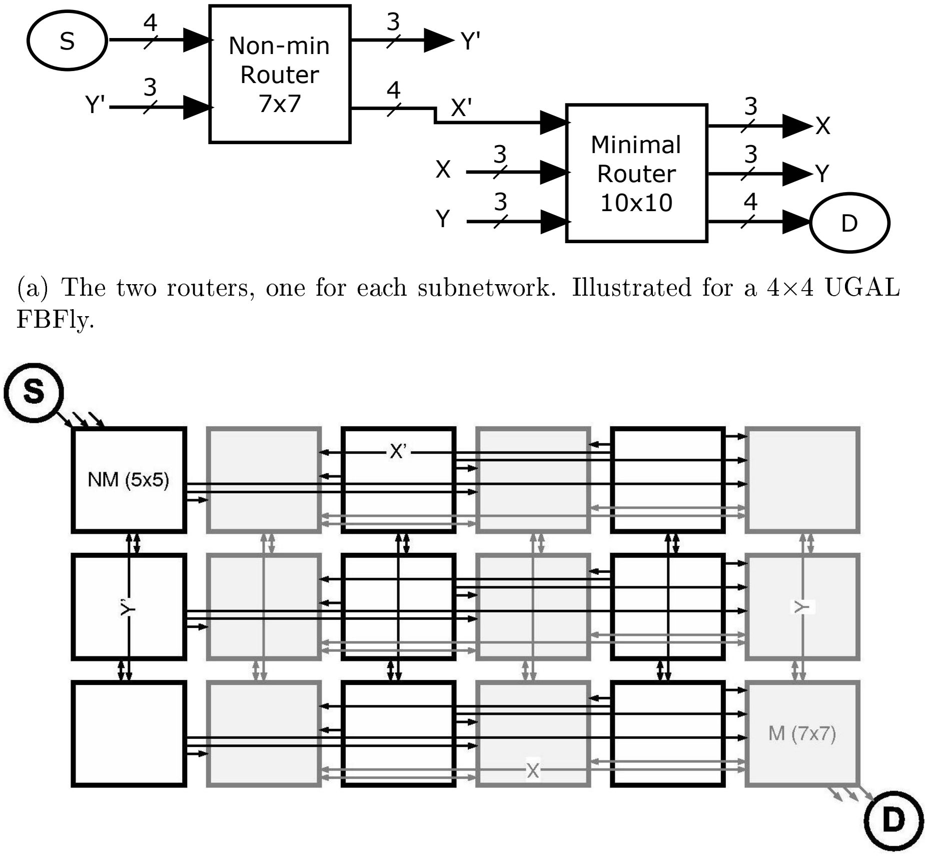


Figure 3.18: EB network congestion sensing mechanism comparison using UGAL in a FBFly.

region of observation. A FIFO at each output stores timestamps of flits entering the output EBs. When a flit is detected to leave the region, the timestamp at the head of this FIFO is removed and used to determine the delay and update the running average.

Figure 3.18 compares the performance of these five congestion sensing mechanisms using universal globally adaptive load-balancing (UGAL) [100] in a flattened butterfly (FBFly) [57] topology. The setup used for the simulation is explained in Section 4.2. *Progressive* refers to progressive adaptive routing (PAR), which revisits the adaptive routing decision at every hop and is explained in detail in Section 3.5.2. The metric of choice is *output occupancy*; this metric is used throughout this study for congestion sensing in EB networks. Occupancy metrics are also the simplest. Metrics using running averages fail to adapt quickly to the current network status. Output occupancy is preferable to channel occupancy because it is important to account for flits that have been routed and waiting for an output as well as flits that have advanced to it. Otherwise, a router input is not aware of other inputs' choices. Thus, an output channel which is never blocked appears as not loaded even though many



(a) The two routers, one for each subnetwork. Illustrated for a 4×4 UGAL FBFly.

(b) A 3×3 UGAL FBFly. Non-minimal router (NM) connect to minimal routers (M – shaded) via X' channels.

Figure 3.19: The UGAL FBFly with EB flow control. Two traffic classes are defined by two subnetworks.

flits may be blocked waiting for that output because all inputs are making the same choice.

3.5.2 Adaptive Routing

Any adaptive routing algorithm is applicable to EB networks by using the congestion sensing mechanism described in Section 3.5.1. Depending on the number of the required traffic classes, network designs should use duplicate subnetworks, the hybrid EB-VC router, or a combination. As an example of customizing to the routing algorithm in EB networks, we apply UGAL [100] to a FBFly [57] topology. To define the *minimal* and *nonminimal* traffic classes required for UGAL, we use two subnetworks interconnected as shown in Figure 3.19.

Packets are injected into the *nonminimal* subnetwork. Then, a random intermediate destination I is chosen. A packet is routed to I (nonminimally to the final destination D) if the load of the output port on the route to I, multiplied by the nonminimal hop count to D, is larger than the load of the output port on the route to D multiplied by the hop count of the minimal route to D. That decision is never revisited. When routing to D or I, packets take up to one hop in each dimension and follow dimension order. In our implementation, output load is measured by the *output occupancy* metric described in Section 3.5.1.

Packets in the *nonminimal* network are routed in Y'X' dimension order to I, and then in YX dimension order to D in the *minimal* subnetwork. Even though within each of the two subnetworks deterministic dimension-order routing (DOR) is used, adaptive routing (UGAL) is applied overall. Packets escape to the *minimal* subnetwork from the *nonminimal* subnetwork once they reach I because routing to D also in the *nonminimal* subnetwork would violate dimension order and cause deadlocks. Packets that do not choose to route to an intermediate destination proceed directly to D in X'Y dimension order. The X' hop places them in the *minimal* subnetwork. Because flits traverse from the nonminimal class to the minimal but not vice-versa, X' channels are not bidirectional. If a nonminimal route has been chosen but there is no X' hop, the first hop after reaching I is X' instead of X since flits would have reached I in the *nonminimal* subnetwork. Therefore, routing becomes Y'X'Y. The channel connecting a nonminimal router to the adjacent minimal router is used when a traversal to the *minimal* subnetwork is required but there is no X' hop. Thus, an extra hop is created. Routing Y'X'YX results in better column load balancing than routing Y'X'XY, since flits in the *minimal* subnetwork traverse the column of the randomly-chosen I. Routing Y'X'XY makes flits use D's column.

Because routers are unaware of distant congestion, we extend UGAL by applying PAR [56] [2]. For every hop towards I, another intermediate destination is randomly

[2] A slightly different form of progressive adaptive routing was first proposed by Steve Scott of Cray for routing in a Dragonfly network.

chosen that would result in an output in the same axis as the output towards I, to preserve dimension order. UGAL is applied to choose the best option between I and the new intermediate destination; that option then becomes I. Packets take up to one hop in each dimension to ensure forward progress. Also, when taking an X' channel to traverse to the minimal network, another X' channel is randomly chosen and UGAL is applied to choose among the two. PAR is not applied in the *minimal* subnetwork because the final destination is constant. Disabling PAR reduces maximum throughput by 14% with our chosen congestion metric [75].

Routers in the *nonminimal* subnetwork have a smaller radix. Thus, they have shorter critical paths and more cycle time to make the complex adaptive routing decisions. On the other hand, adaptive routing decisions are made in the *nonminimal* subnetwork and cannot consider the load of the *minimal* subnetwork because it is distant.

VC networks share channels between the different traffic classes, and thus average out channel utilization. For the EB network, PAR is a cause of load imbalance in the *minimal* subnetwork, because it makes the intermediate destination choice not truly random. Otherwise, traffic to the *minimal* subnetwork would be uniform random which, on average, balances load. Finally, traffic that needs to be ejected from the same router it was injected to needs to traverse to the *minimal* subnetwork, using a X' link. This makes that traffic contend with other traffic. This issue can be alleviated by increasing network cost to add more channels, such as ejection ports in the *nonminimal* subnetwork.

3.6 Methodology

We use a modified version of Booksim [24] for elastic buffer (EB), hybrid EB-virtual channel (VC) and hybrid EB-wormhole networks. We use two topologies: a 2D mesh with dimension-order routing (DOR) and a 2D flattened butterfly (FBFly) [57] with universal globally adaptive load-balancing (UGAL) routing [100], as described in Section 3.5.2 for the EB network. We assume the same amount of requests and replies in the network. Requests are divided into read requests and write requests. Similarly, replies are divided into read replies and write replies. Read requests and write replies are carry only control information, whereas write requests and read replies also carry data.

The mesh is configured as a 4×4 or an 8×8 grid. Each router has a single node attached to it. The FBFly has four nodes attached to each router (concentration factor of four), and routers arranged in a 4×4 grid. Injection and ejection channels have a single cycle of latency. The 4×4 mesh channels have two cycles of latency, while the 8×8 mesh channels have one cycle of latency. For the FBFly, short, medium and long channels have 2, 4 and 6 cycles of latency, respectively. The channel connecting two adjacent routers of different subnetworks for the EB UGAL FBFly has one cycle of latency. Also, *nonminimal* routers are 7×7 while *minimal* ones are 10×10. progressive adaptive routing (PAR) is applied to both the EB and VC FBFly. In all networks, each cycle of latency represents 2mm of physical distance.

Sources generate fixed-size 512-bit packets and enqueue them into the injection buffer of the proper subnetwork. Each buffer can inject and eject up to a single flit per cycle. No communication protocol was assumed; therefore, we used a single physical network defining a single traffic class. The set of traffic patterns [24] used for the evaluation is uniform random, random permutations, shuffle, bit complement, tornado and neighbor traffic for the mesh. For the FBFly we also include transpose and an adversarial traffic pattern. The adversarial traffic pattern aims to load a small subset of network channels by making all sources connected to a router send to destinations connected to a single other router. For the EB networks, transpose illustrates the effect of traffic destined to the same-coordinate router now contending with other network traffic to switch to the *minimal* subnetwork by taking an X' hop, as described in Section 3.5.2.

For the Pareto-optimal curves shown, the datapath width is swept from 29 to 171 bits such that packets consist of 3-18 flits. Flits are of the same width as the router datapath (they consist of 1 phit). Also for the Pareto-optimal curves, the maximum throughput is the average of the maximum throughput of each traffic pattern; the consumed power is the average of the power consumptions at the maximum throughput of each traffic pattern. Percentage summaries are calculated by calculating the average distance between sampling points of different networks, dividing by the normalized aspect, and averaging among all sampling points. Pareto-optimal curves focus on maximum throughputs to extract the throughput per unit area or power for each network. We discuss latency and power before saturation separately.

VC networks use a two-stage router design [85]. The first stage consists of input buffering, look-ahead routing [32], VC and speculative switch allocation [85]. The second stage is switch traversal. VC and switch allocators are separable input-first with round-robin arbiters, executing a single iteration per cycle. EB networks also use round-robin arbiters. We do not assume input buffer bypassing [107]. Our energy evaluation focuses at the saturation points at which this has minimal effect. Buffer bypassing would require extra pipeline flip-flops (FFs) and may extend some timing paths, so at high loads it would have a slightly detrimental effect. When comparing the VC and the hybrid EB-VC networks, all data points assume one VC per traffic class and an equal number of VCs between the two networks. Eight buffer slots are statically assigned to each VC for both networks. For each datapath width used when comparing EB to VC or hybrid EB-VC networks, we sweep the number of VCs and buffer slots statically assigned to each to maximize throughput per unit power. We only consider buffer depths that cover the credit round trip latency to avoid penalizing latency and channel under-utilization. For a 64-bit datapath, 4 VCs is the optimal choice for the UGAL FBFly, of 10 slots each for full-swing and 8 slots for low-swing channels. For the mesh, the optimal choice is 4 VCs of 9 slots each for full-swing and 8 for low-swing channels. The FIFOs of the wormhole routers have eight slots; seven of these are necessary to cover the buffer turnaround delay, and adding an extra entry to bring the buffer size to a power of two allowed us to simplify buffer control logic. The hybrid EB-wormhole router was configured with a six-entry FIFO in addition to its intermediate EB to also provide a total of eight buffer slots. In order to maintain the aforementioned properties, we kept the number of buffer slots constant when adjusting the datapath width.

Area and power results are based on cost models or placement and routing. Comparisons among EB routers but not with the VC and hybrid EB-VC routers use placement and routing. Comparisons with the VC or hybrid EB-VC networks use the cost models described in [6]. Comparisons between the VC and the two-stage EB routers use device and interconnect parameters from a 65nm general-purpose CMOS technology in the typical case. However, comparisons among the EB routers as well as the hybrid EB-VC router use device and interconnect parameters from a commercial 45nm low-power technology library, under worst case conditions for both energy and timing. Since the 45nm library is commercial, it more clearly represents an implementation technology used in practice, whereas the 65nm library was more appropriate for comparisons against the VC router because it is not low-power and therefore gives realistic estimates for the buffer leakage power. To clearly illustrate the reduced complexity of EB routers compared to VC routers, we also include placement and routing results. All placement and routing comparisons are performed using the same 45nm library.

The area and power models route wires above other logic and report only the area of the repeaters and FFs. Therefore EB channels have the same area and power as pipelined channels with FFs because they have the same number of latches in the datapath. Router area is estimated using detailed floorplans, and input buffers are implemented as custom SRAMs. Critical devices in the channels and router datapaths, such as repeaters used to drive large wire capacitances, are sized to ensure circuits will operate at the clock frequency. The power model includes the major devices in the channels and routers, and includes leakage currents. The FFs in the channels are clock-gated locally. We also present results for low-swing channels. Aggressive low-swing channel designs can achieve up to a 10× traversal power per bit reductions compared to full-swing [45]. As a conservative estimate, our low-swing channel model has 30% of the full-swing repeated wire traversal power, and double the channel area.

To perform the placement and routing comparisons, we synthesize a single router instance using Synopsys Design Compiler and place and route (PnR) the synthesized netlist using Cadence Silicon Encounter. Placement and routing captures cost not regarded by the cost models, such as the cost for arbitration or allocation, credits, the synchronization logic presented in Section 3.2.1 as well as the ready-valid handshake logic. Clock frequencies are determined by static timing analysis using post PnR parasitics. Due to technology usage constraints, input buffers were implemented from FF arrays in the PnR flow. Routers are optimized for minimum cycle time in the PnR flow. Energy per transferred bit was calculated by driving the post-PnR netlists with pseudo-random input traffic under an equal cycle time and flit injection rates, using a test environment separate from the cycle-accurate network simulator that we discuss above. Low-power optimizations, such as clock gating the FIFO FFs, were automatically applied by the synthesis and PnR tools. The initial floorplan utilization is set to 70%. Primary input and output driving strengths, loads and timing constrains are specified to realistically assume network channels at router ports. Router ports are placed in the floorplan according to the inter-router connections of the assumed network. The switch is implemented using multiplexers.

To illustrate the effects of removing the buffers, simulations comparing VC routers against EB or hybrid EB-VC routers assume a clock frequency of 2GHz. Comparisons among EB routers as well as the hybrid EB-wormhole routers use each router's minimum cycle time from PnR for each router and datapath width. This is a design choice which prioritizes cycle time, instead of energy or area. The minimum clock cycle for each router was determined after performing static timing analysis with post-PnR parasitics. Furthermore, all comparison results (including Pareto-optimal curves) with the hybrid EB-wormhole routers use area and energy results from PnR, instead of cost models. However, to illustrate the influence of cycle time, we also present throughput curves assuming an equal clock frequency for the EB routers. These curves use a cycle time of 4.45ns. Even with the same cycle time, the routers are still optimized for minimum cycle time in the PnR flow. Throughput and latency

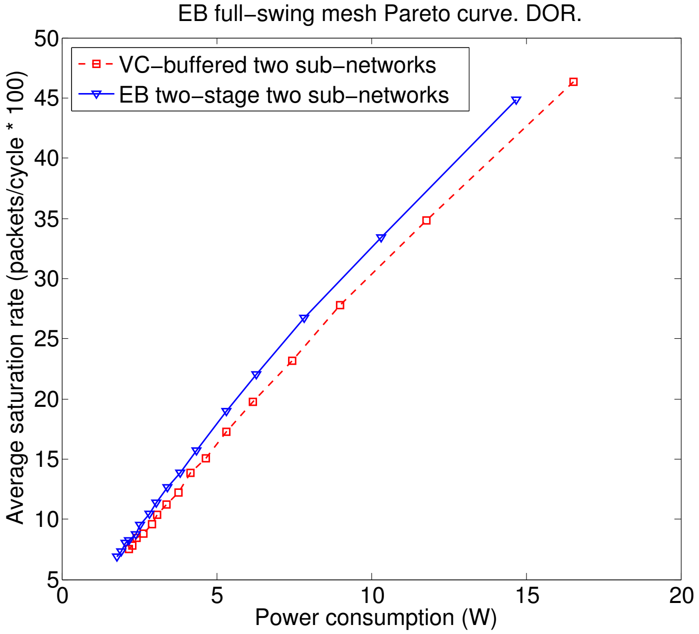


Figure 3.20: Power-throughput Pareto-optimal curve for the mesh and full-swing channels.

are measured in absolute time for curves using different cycle times.

3.7 Evaluation

This section presents evaluation results for the EB and hybrid routers.

3.7.1 EB and VC Network Comparison

First, we compare the VC router with the two-stage EB router. Figure 3.20, Figure 3.21, Figure 3.22 and Figure 3.23 show Pareto-optimal curves for the 4×4 2D mesh and the UGAL FBFly, with routers operating at an equal clock frequency. Therefore, these curves ignore benefits from the reduced cycle time of EB routers, discussed later. These curves illustrate Pareto-optimal design points which show the maximum throughput achieved by the two networks given a certain area or power budget, as well as the area or power required to achieve a certain maximum throughput. Table 3.1 summarizes the percentage gains. Rows indicate which aspect was

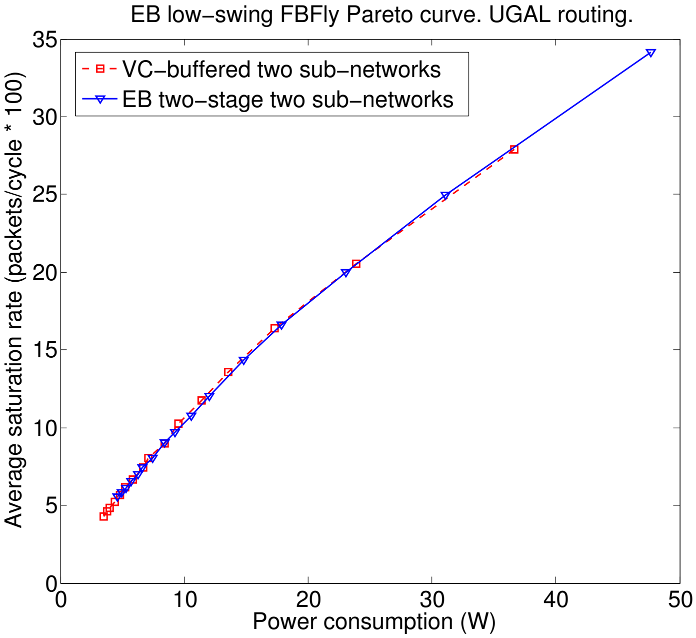


Figure 3.21: Power-throughput Pareto-optimal curve for the FBFly and low-swing channels.

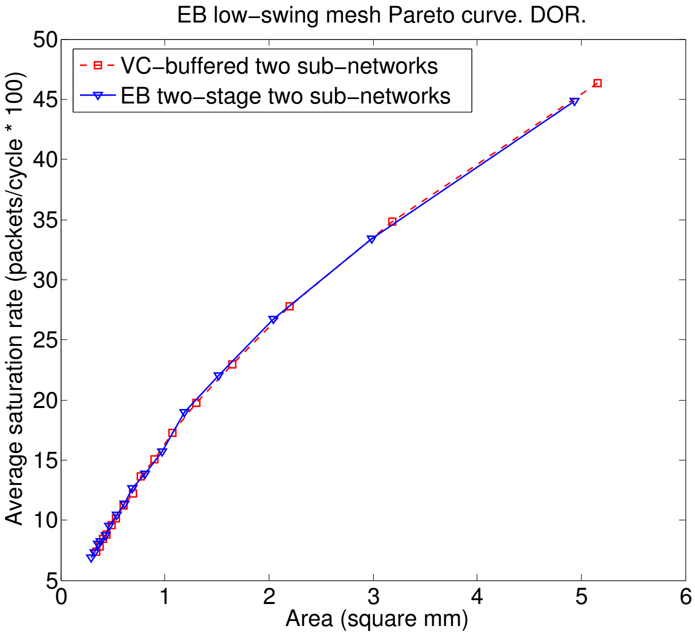


Figure 3.22: Area-throughput Pareto-optimal curve for the mesh and low-swing channels.

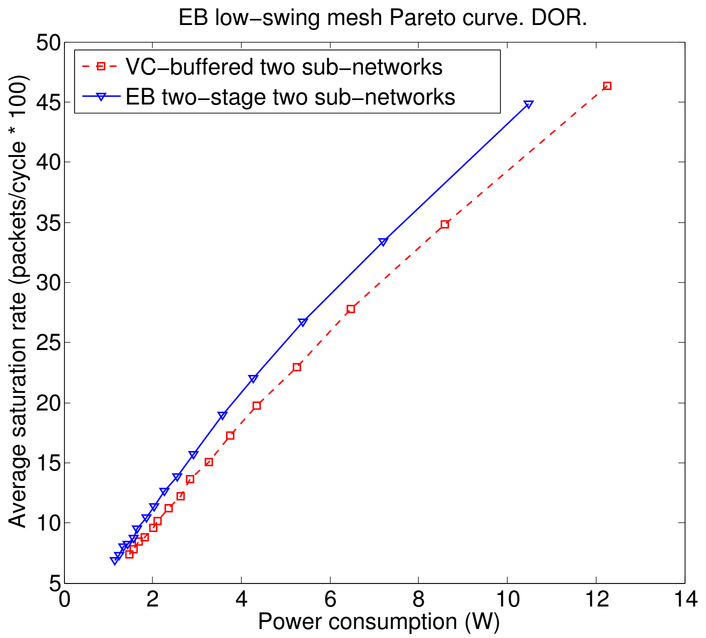


Figure 3.23: Power-throughput Pareto-optimal curve for the mesh and low-swing channels.

Table 3.1: Two-stage EB network percentage gains compared to VC.

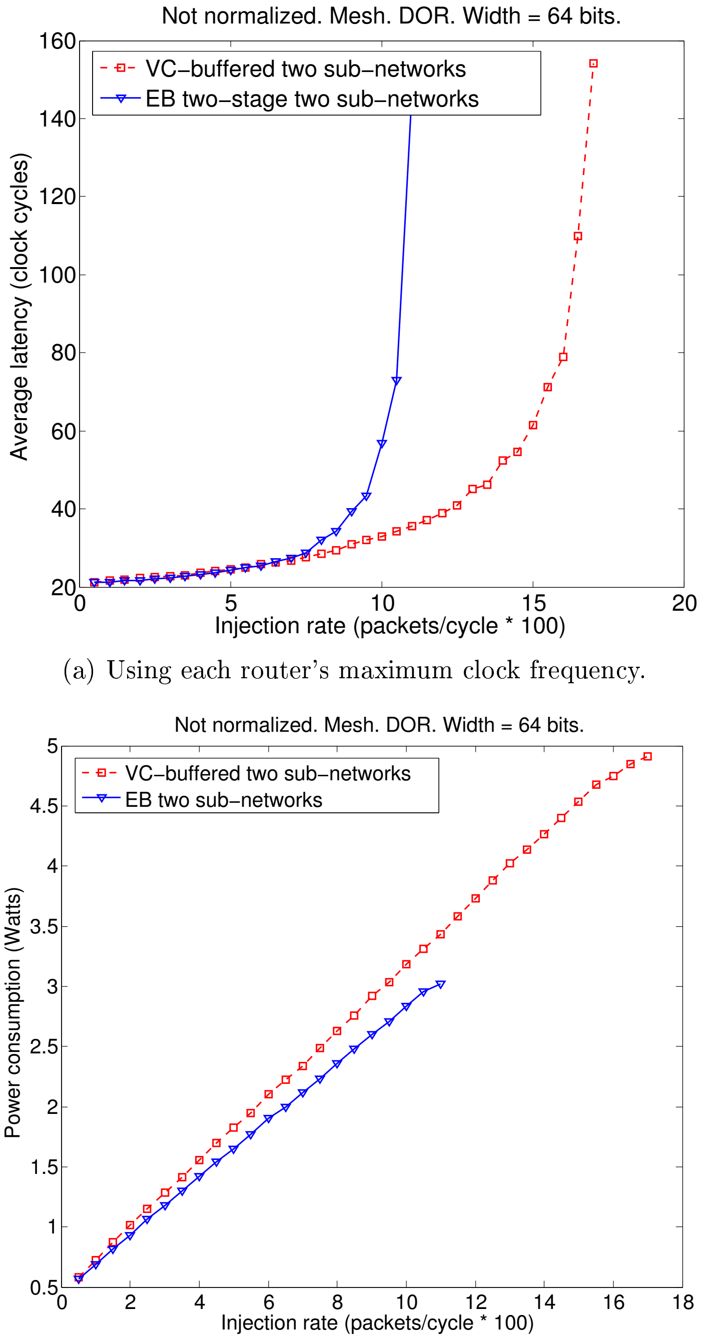
Norm	DOR Mesh			UGAL FBFly		
Comp:	Area	Thr.	Power	Area	Thr.	Power
Full-swing						
Area	-	1%	7%	-	-10%	10%
Throughput	2%	-	8%	-20%	-	-3%
Power	-11%	8%	-	-16%	-2%	-
Low-swing						
Area	-	2%	10%	-	-11%	15%
Throughput	2%	-	12%	-23%	-	0%
Power	-15%	10%	-	-24%	0%	-

equalized between the EB and VC networks. Columns show the percentage gains for the mesh and the FBFly. Positive percentages indicate gains for the EB network. Section 3.7.2 offers more insight for buffer costs which result in the performance per unit cost increase of EB networks.

Even though the VC network does not need multiple subnetworks to define traffic classes, both the VC and EB networks use two subnetworks for a fair comparison. One subnetwork in each case carries requests, while the other replies. This produces a fair comparison because splitting a single network into two subnetworks increases throughput per unit power due to the crossbar's quadratic cost relationship with the number of incoming and outgoing wires, as explained in Section 3.3.2. The VC network still has multiple VCs per subnetwork to alleviate head-of-line blocking.

Figure 3.24(a) shows latency as injection rate increases for a 64-bit datapath. Zero-load latency is equal because the two routers have the same number of pipeline stages and EB channels have the same latency as channels with an equal number of FFs. In the UGAL FBFly, zero-load latency is 3% higher for the EB network due to using the channels to transition from the *nonminimal* to the *minimal* subnetwork. Due to the lack of input buffers and VCs which results in increased head-of-line blocking, the EB network is not able to reach the VC network's maximum channel utilization rate. Therefore, the EB network saturates at a 34% lower injection rate than the VC network. Latency increases in a similar manner in the VC and EB networks with injection rate.

EB networks consume less power than VC networks for equal injection rates which do not saturate either network, as shown by Figure 3.24(b). At the EB network's saturation rate, the VC network consumes 14% more power due to the input buffers. VC networks which bypass input buffers when they are empty and under no contention [107] would still not consume less power than EB networks, because flits would traverse as many pipeline FFs as EBs in the two-stage EB router. Thus, the datapath energy would be comparable. The single-stage EB router would consume less power because it lacks pipelining overhead. Furthermore, buffer bypassing would add logic and cost overhead to the VC router under high load, where bypassing buffers is rare.



(a) Using each router's maximum clock frequency.

(b) Equal clock frequencies.

Figure 3.24: Latency and power by increasing the injection rate for the 4×4 mesh under uniform random traffic.

Table 3.2: Two-stage EB and VC router implementation comparison.

Aspect	VC router	EB router (2-stage)
Number of ports	703	683
Number of nets	16202	6117
Number of gates	60010	12269
Number of cells	15943	5691
Area (μm^2)	63515	15080
Cycle time	3.3ns (41FO4)	1.8ns (22FO4)

As explained in Section 3.7.2, EB networks trade cost savings for wider datapaths to increase throughput. Therefore, EB networks that have equal throughput or cost with a VC network will have a lower serialization latency. EB networks using the single-stage design would further reduce zero-load latency.

Table 3.2 presents PnR results for 5×5 mesh routers using DOR. The VC router has 2 VCs of 8 buffer slots each. Results show a 76% decrease in occupied area and an 45% decrease in cycle time. The reduced cycle time enables the network to be clocked at a higher frequency, thus achieving higher throughput in absolute time, or lower zero-load latency if the pipeline stages in the VC router are increased.

The VC router cycle time is constrained by the VC and switch allocators. Increasing the number of VCs increases the complexity of the first stage of the VC router. This shows the gain in cycle time of reducing router complexity by removing VCs and replacing allocation with arbitration. This is only slightly affected by the buffer implementation since the critical path begins at buffer read. However, that timing overhead remains considerably larger compared to EB read.

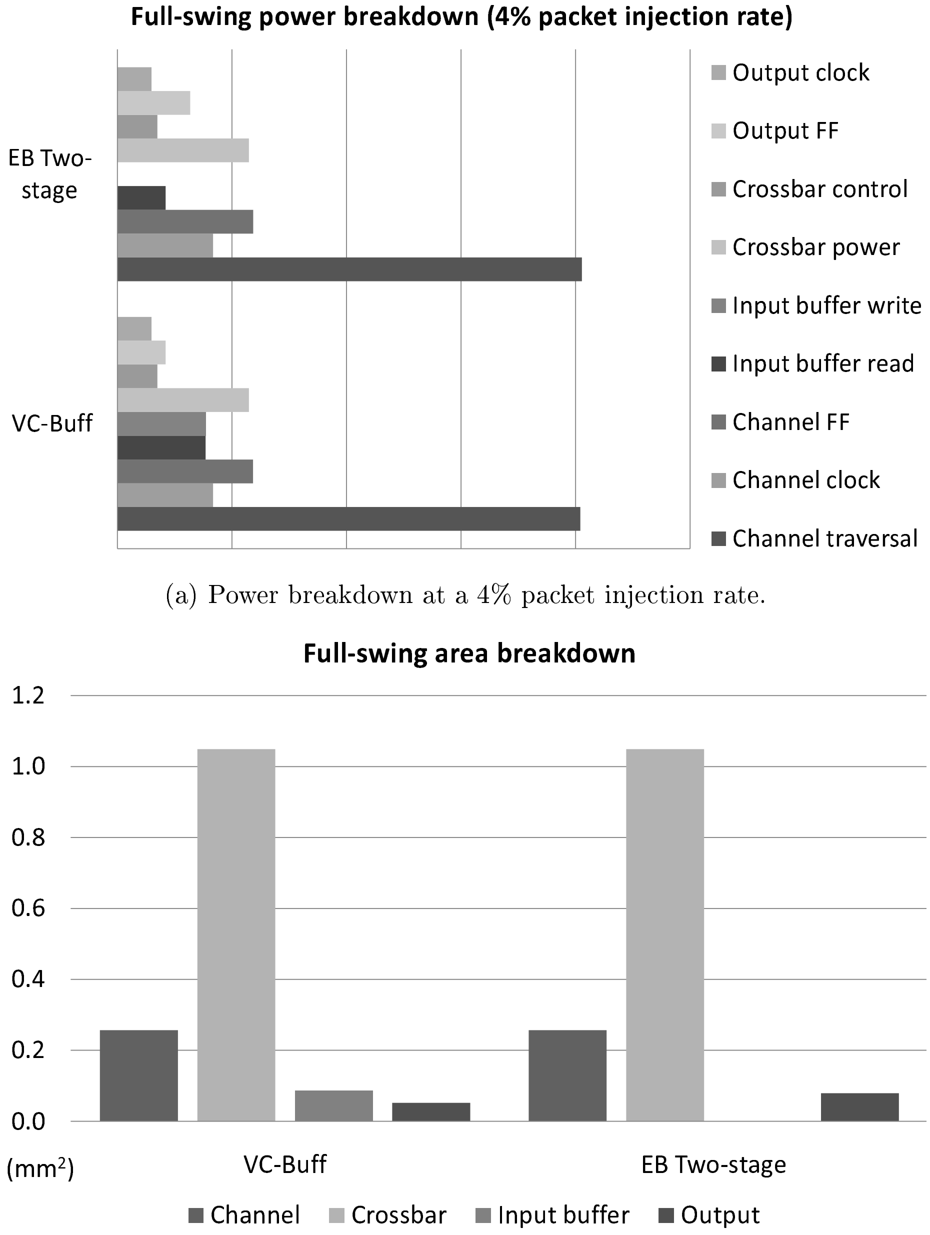
The amount of buffering in EB channels scales directly with channel length. Topologies with double the channel length have an increased throughput of 8-10% in the UGAL FBFly using the two-stage EB router. On the other hand, they have almost double the channel power for a total power increase of approximately 60%. This small change in throughput shows that the dominant factor affecting maximum throughput in EB networks is the contention in the bufferless routers. Therefore, networks with short (even single-cycle) channels are still good candidates to use EB flow control, because it will be more performance-over-cost efficient compared to buffered networks. However, designers might still find it beneficial to add storage in EB channels depending on their topology, layout and router radix. This can be done by adding EBs, adding latches to existing EBs, or using repeaters for additional storage [82].

3.7.2 Buffer Cost Impact

Figure 3.25 shows an area and power breakdown for a FBFly using DOR and full-swing channels. DOR is chosen for fairness to have a single EB subnetwork and ensure that flits traverse the same paths in the EB and VC networks. Figure 3.26 presents the same power breakdown for low-swing channels. *Channel traversal* refers to the power to traverse a segment with repeaters. For the EB network, *input buffer read* power is the traversal power for the intermediate EB shown in Figure 3.5. Channel and switch area and power remain the same. The difference between the buffer power and the intermediate EB power is the amount saved by removing the buffers. Buffer power in the VC network is 15.5% with full-swing channels and 21.5% with low-swing channels of the overall power. Low-swing channels double the channel area due to differential signaling, making buffer area a smaller percentage of the overall area. They also reduce channel traversal power, making the buffers a more significant ratio of the network power. This increases the EB network's power gains from removing the buffers.

Removing router buffers provides power and area savings for EB networks. Those savings can be traded for a wider datapath. This way, power or area can become equal to the VC network. Then, performance and other aspects can be compared. Therefore, the increase in performance per unit area or power of EB networks is dictated by the area or power of the overall network that is consumed by the buffers.

Section 3.7.5 compares EB and wormhole routers and shows that removing the buffers is the dominant factor for the EB network cost savings [76]. Since EB and wormhole routers have directly comparable complexity due to the lack of VCs, this



(a) Power breakdown at a 4% packet injection rate.

(b) Area breakdown (mm^2).

Figure 3.25: Cost breakdowns for a DOR FBFly with 64-bit full-swing channels under uniform traffic.

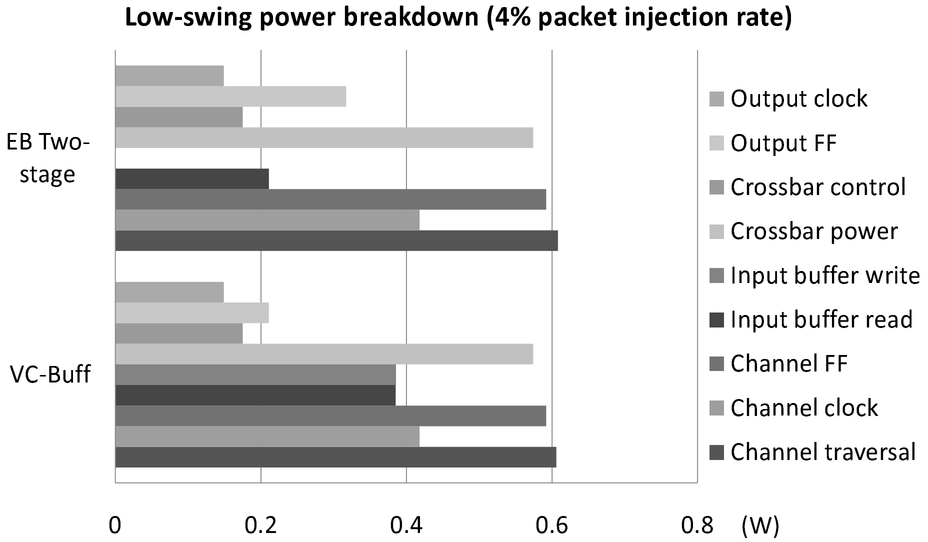


Figure 3.26: Low-swing power breakdown. Same case as Figure 3.25.

isolates the contribution of buffer cost from the extra complexity of VC flow control. On the other hand, simplifying allocation primarily reduces cycle time.

Topologies with a higher number of average hops consume more power in the input buffers because packets traverse more buffers until their destinations, and thus are more beneficial to EB networks because more energy is traded for a wider datapath when switching to EB flow control. Figure 3.27 illustrates this by showing a power breakdown in the mesh with low-swing channels. In this case, input buffers consume 25% of the network power.

The ratio of area and power consumed by the buffers depends on every part of the network. Therefore, circuit optimizations of various network components will reduce the contribution of those components and will increase the ratio of energy consumed in the buffers, making EB networks more attractive. Architectural choices also affect buffer power. For example, apart from the average hop count of a topology, routing algorithms also affect hop count and schemes for dynamic buffer sharing reduce buffer size, thus reducing the buffer read and write energy [87].

Finally, buffer cost heavily depends on the implementation. Implementing buffers from FF arrays requires the least implementation effort. On the other hand, efficient custom SRAM buffers consume less power. The comparisons performed in our study

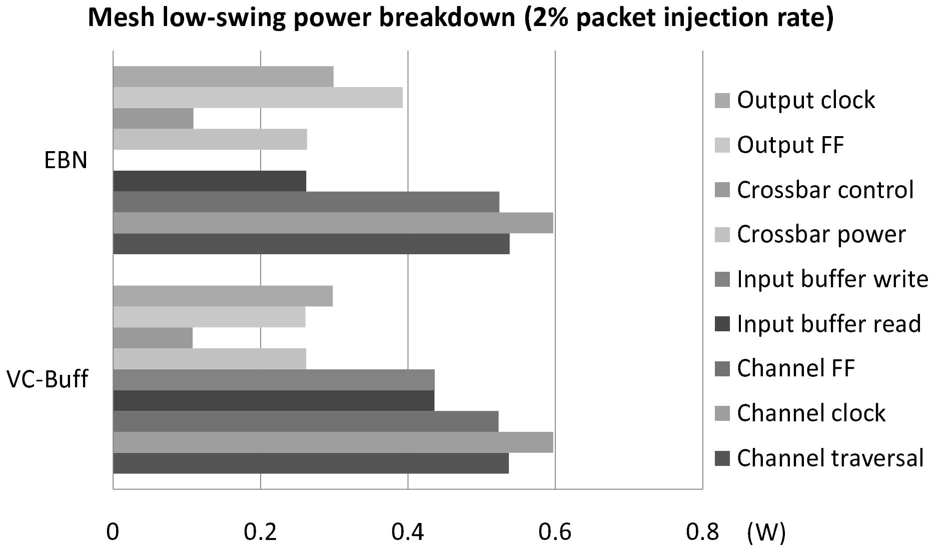


Figure 3.27: Low-swing power breakdown for the mesh.

would be more in favor of EB networks if we had assumed more expensive buffer implementations than our efficient custom SRAMs. However, we note that due to implementation or technology library constraints, some designers may be forced to use costly buffers. Finally, buffer cost can be reduced if the design is focused at some end of the operating spectrum. For example, empty buffer bypassing saves the majority of buffer dynamic power under low loads [107, 81].

3.7.3 Elastic Buffer Router Design Comparison

This section compares the two-stage and single-stage EB router designs. Figure 3.28, Figure 3.29 and Figure 3.30 show PnR results. These curves are noisy because the software tools for the PnR flow use randomized algorithms with heuristics (such as simulated annealing) to perform optimizations on discrete values (such as cell sizing). The two-stage router has a cycle time reduced by 26% compared to the single-stage router. The single-stage router requires 19% less energy per transferred bit compared to the two-stage router. Finally, the single-stage router occupies 30% less area than the two-stage router.

The reduced area and energy of the single-stage EB router is due to the lack

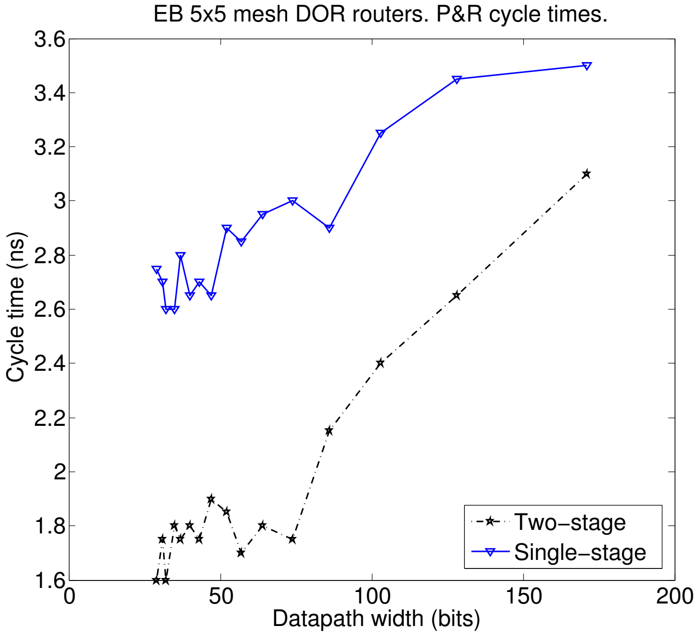


Figure 3.28: EB router cycle time PnR implementation results.

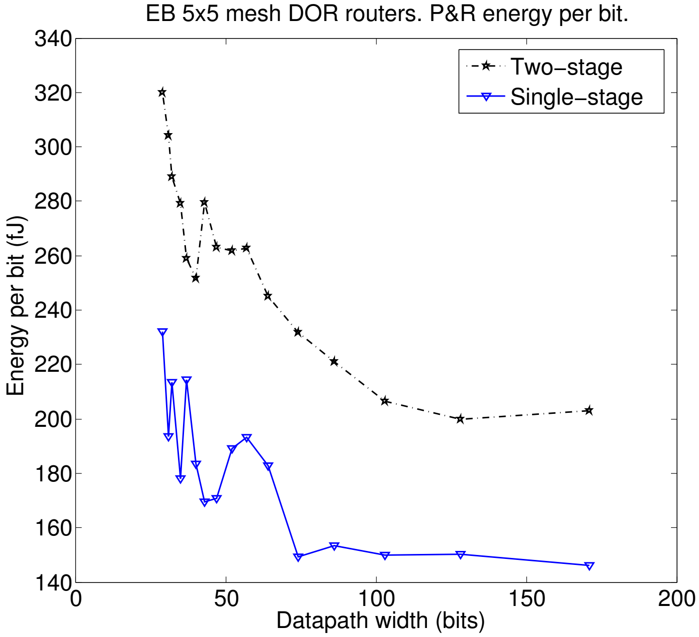


Figure 3.29: EB router energy per bit PnR implementation results.

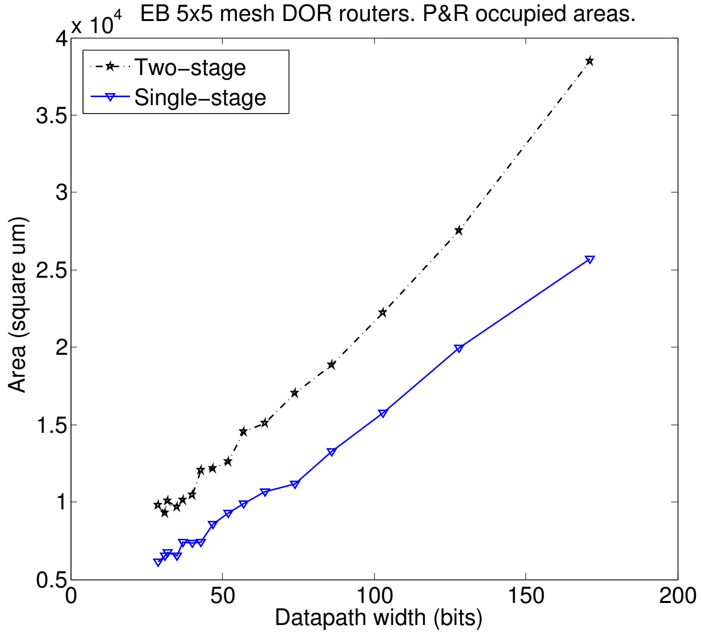
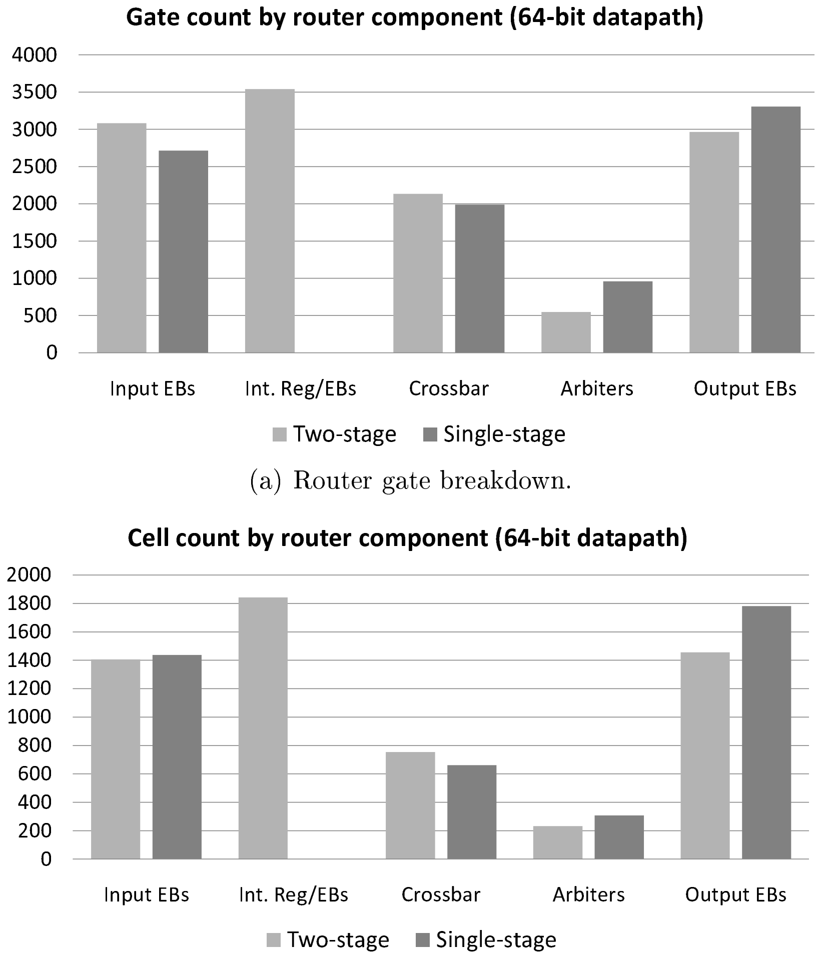


Figure 3.30: EB router area PnR implementation results.

of pipeline overhead. Since the synchronization module of the two-stage EB router operates only on chosen output port bits, its contribution is small compared to the datapath. The increased energy cost of the two-stage router is attributed to the addition of the synchronization logic, splitting the intermediate register cell into two latch cells to implement the intermediate EB, and to the increased clock frequency, which forces the cells to have greater driving strength. This also affects the switch. The single-stage router is much simpler, reducing its energy consumption.

The lack of pipelining is also the reason that the single-stage EB router has a larger cycle time than the two-stage EB router. However, the increase is still less than a factor of two. The difference in cycle time has a small effect on occupied area because it only affects cell sizing and placement. Instead, the dominant factor is component complexity.

The two-stage EB router is constrained by the first stage only for datapath widths smaller than 47 bits. This means that further optimizations should focus on other aspects of the router for larger datapath widths. As the datapath width increases, the cycle times of all routers converge. This is because the switch dominates the cycle



(a) Router gate breakdown.

(b) Router cell breakdown.

Figure 3.31: Router PnR gate and cell breakdown.

time at large widths and all routers use identical switches. Techniques such as switch slicing thus will allow the two-stage router to be clocked at smaller cycle times for large datapath widths. However, routers with large critical paths on the first stage will have their cycle times unaffected.

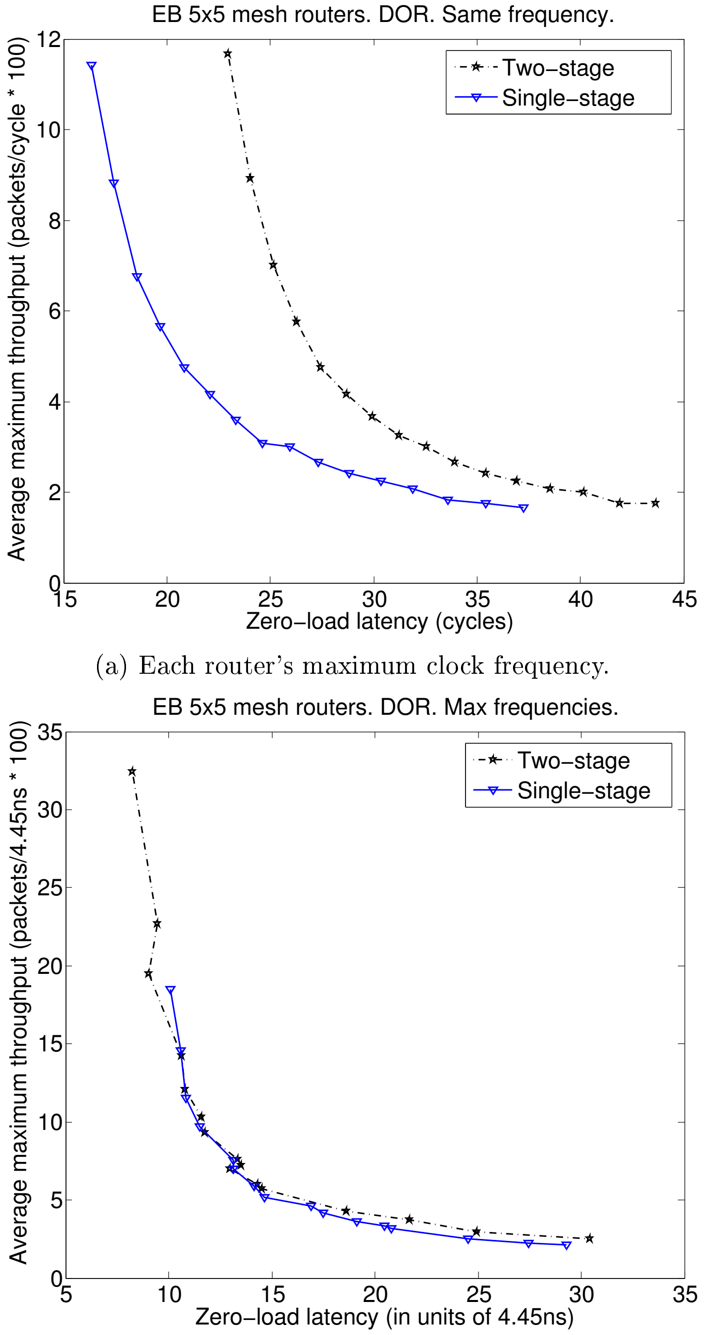
Figure 3.31 shows a gate and cell count for the EB routers. The cell count of the intermediate EBs of the two-stage router includes the synchronization module logic. As shown, the various components of the two routers have a directly comparable number of gates and cells. However, the two-stage router has 39% more gates and

44% more cells due to the intermediate EB and synchronization module logic. Furthermore, the gate and cell count of the switch arbiters is low (61% fewer gates than the mux-based crossbar), illustrating the low arbitration complexity of EB routers.

As shown in Figure 3.32(a), the single-stage EB router offers the smallest zero-load latency per unit throughput, on average. This is because of the single pipeline stage. Its effect is directly dependent on the average number of hops of our network, therefore especially important in our multi-hop 2D 8×8 mesh. However, the difference with the two-stage EB router significantly decreases when clocking each router at its maximum frequency, shown in Figure 3.32(b), compared to clocking them at an equal frequency. This is because the network with the two-stage router also has higher-frequency channels, which have lower latency in absolute time. However, these results rely on the channel latency in clock cycles remaining equal when increasing the clock frequency (the number of retiming elements remains constant). While this is true for our clock frequencies and physical channel lengths, other network settings might find that increasing the clock frequency also increases the channel latency in clock cycles. In that case, the single-stage router will provide an additional reduction in latency compared to the two-stage router. Networks with a small average hop count or channels with many pipeline stages will have their latency influenced more by channels than by routers.

Since the routers were placed and routed for maximum clock frequency, their occupied areas remain constant regardless of the clock frequency they operate at. At an equal clock frequency, the single-stage router provides the most throughput per unit area because it occupies the least area. Therefore, the single-stage router has an increased datapath width compared to the two-stage router occupying the same area and provides a higher throughput. However, if the routers operate at their maximum frequencies, the two-stage router provides more throughput per unit area due to its reduced cycle time.

From the two EB routers, the two-stage router is optimal for area and consumes the least energy. However, it is closely followed by the single-stage router which carries cycle time, latency and area benefits. Designs with zero-load latency in mind should take into account the average number of hops and the effect on channel latency in



(a) Each router's maximum clock frequency.

(b) Equal clock frequencies.

Figure 3.32: EB router latency-throughput comparison.

Table 3.3: Which of the two EB routers is optimal in our 8×8 2D mesh depending on design priority.

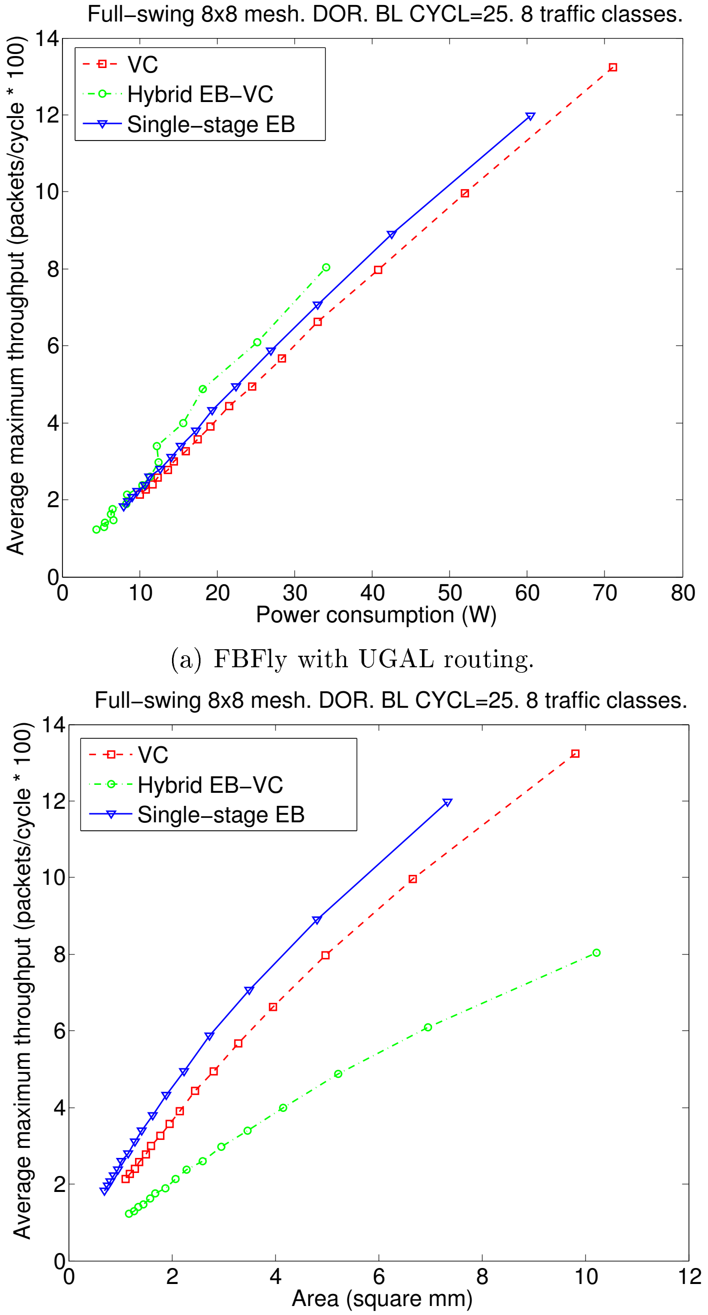
Priority	Router choice
Operate at maximum frequencies	
Area	Two-stage
Energy	Single-stage
Latency	Single-stage (depends on effect on channels)
Operate at the same frequency	
Area	Single-stage
Energy	Single-stage
Latency	Single-stage

clock cycles when applying the two-stage router's maximum clock frequency. Network designs which would clock all routers under the same frequency have the single-stage router as their optimal choice for area and latency. Examples of such designs can be systems-on-chip, which may not require a higher clock frequency or may keep the network clock synchronized to a slower system-wide clock to avoid multiple clock domains. Table 3.3 summarizes which EB router is the optimal choice depending on design priorities.

Using the optimal EB router and the shortest cycle time for each comparison, EB networks provide an up to 45% shorter cycle time which also translates to 22% increase in throughput per unit area compared to VC networks.

3.7.4 Hybrid EB-VC Networks

Figure 3.33 shows Pareto-optimal curves for a network with the hybrid EB-VC router. The number of blocking cycles before a flit is drained (*BL CYCL*) is set to 25. There are 8 traffic classes with 8 buffer slots per class. Clock frequencies are equal. Only



(a) FBFly with UGAL routing.

(b) 2D mesh with DOR.

Figure 3.33: Hybrid EB-VC Pareto-optimal curves.

results for the single-stage EB router are included because it's the optimal choice for equal clock frequencies, as shown in Table 3.3. The hybrid router provides 21% more throughput per unit power compared to the VC router, and 12% more compared to the single-stage EB router. On the other hand, the VC router provides 41% more throughput per unit area while the single-stage provides 49% more. Zero-load latency is equal for the hybrid and VC routers.

Smaller *BL CYCL* values reduce the throughput per unit power of the hybrid router. For instance, with *BL CYCL* set to 4, the single-stage EB router provides 5% more throughput per unit power. On the other hand, increasing *BL CYCL* above 25 has no effect on throughput per unit power, but increases the worst-case blocking latency. Different values of *BL CYCL* do not considerably affect maximum throughput, because blocking a flit from a different class even for a few cycles penalizes throughput significantly. However, they do affect how often the buffer of the hybrid router is used. For small values, the buffer is used even in the common case, while increasing *BL CYCL* above 25 makes no difference because buffers are still used only in situations which would otherwise result in a deadlock and to alleviate severe contention. Therefore, small *BL CYCL* values make the hybrid router's power consumption comparable to the VC router's, but without a proportional increase in throughput due to head-of-line blocking.

Increasing the number of buffer slots per traffic class to 16 makes the EB router marginally (2%) more throughput per unit power efficient than the hybrid router. While buffers can hold more flits, this rarely happens because by the time 15 other flits of the class being drained arrive, the oldest drained flit has traversed the switch. Also, increasing the buffer size increases the energy cost for accessing it. On the other hand, increasing the number of traffic classes to 16 reduces throughput per unit power of the hybrid router only marginally (2%) compared to the VC. This percentage increases to 8% for small values of *BL CYCL*. This is because increasing traffic classes increases the probability of blocking in the EB channels. This causes more flits to be drained, increasing power. However, area efficiency for the hybrid router is also marginally increased (2%) because more draining increases throughput without affecting area.

The hybrid EB-VC router increases throughput per unit power because it uses buffers only to alleviate head-of-line blocking and to resolve cases which would otherwise result in a deadlock. In the common case, buffers are not used and so the hybrid router is almost as energy efficient as the two-stage EB router. To retain the power efficiency of EB networks, EBs remain the primary means of buffering. To accomplish this, *BL CYCL* should not be small. We note that our experiments used a low-power library, which has negligible leakage power. Increasing leakage would reduce the power efficiency of the hybrid router compared to EB routers. However, this effect would be very small because in a similar network as our 8×8 2D mesh, buffer leakage was only 1.5% of the overall network power, which was dominated by channel power [81]. Regardless of technology library, the buffers occupy area and may extend the critical path.

3.7.5 Wormhole and Hybrid EB-Wormhole Routers

This section compares EB, wormhole and hybrid EB-wormhole routers described in Section 3.4.2. Because wormhole routers also lack VCs and thus have comparable complexity to EB routers, this comparison isolates and illustrates the effect of removing input buffers to the performance and cost efficiency of the network.

Figure 3.34, Figure 3.35 and Figure 3.36 show PnR implementation results. Similarly to Section 3.7.3, the curves are noisy as a result of the heuristic algorithms that EDA tools use to perform optimizations on discrete values, e.g. for performing cell sizing.

Comparison results of the single and two-stage EB routers are consistent with Section 3.7.3 [77]. The wormhole and hybrid routers have a 27% and 34% larger average cycle time compared to the two-stage router, respectively. Likewise, they require 1% and 21% more energy per bit and occupy 2.3 and 1.5 times the area compared to the single-stage router. These comparison results are averaged across all datapath widths.

The two-stage router requires the most energy because it was placed and routed to meet its small cycle time. This increases the sizing of cells in the router. The

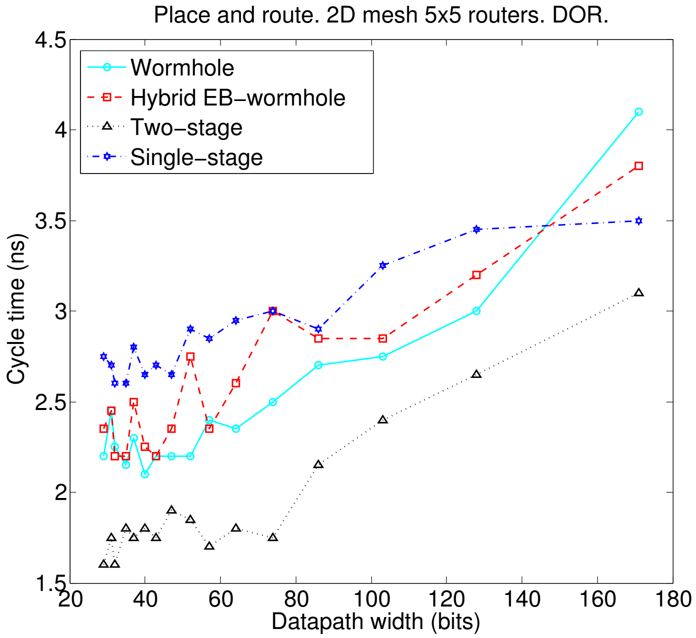


Figure 3.34: EB and wormhole routers cycle time after PnR.

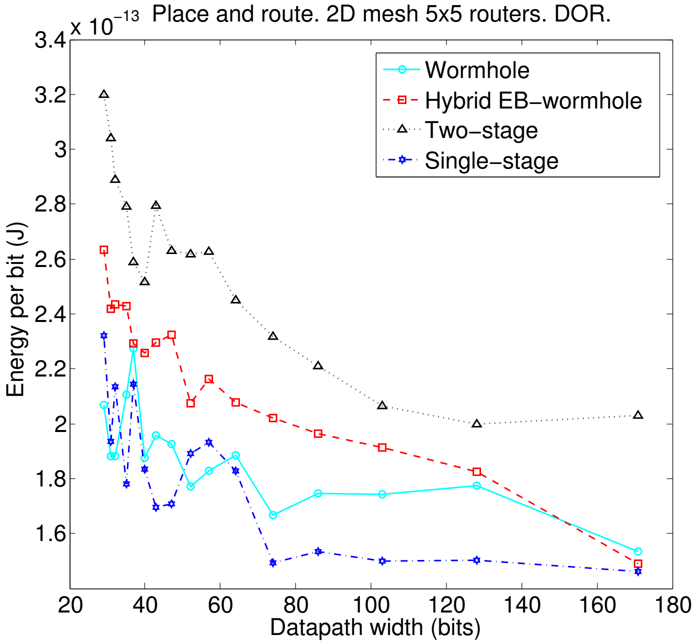


Figure 3.35: EB and wormhole routers energy per transferred bit after PnR.

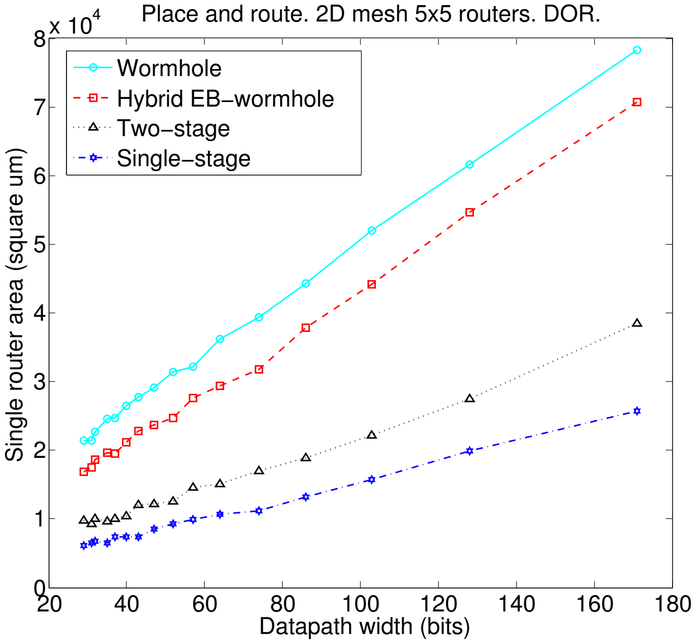
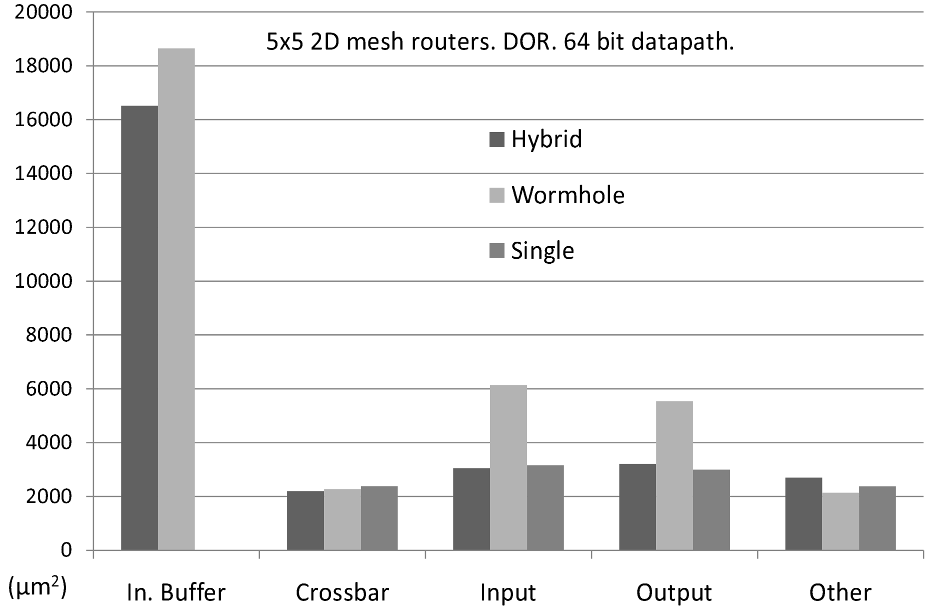


Figure 3.36: EB and wormhole routers occupied area after PnR.

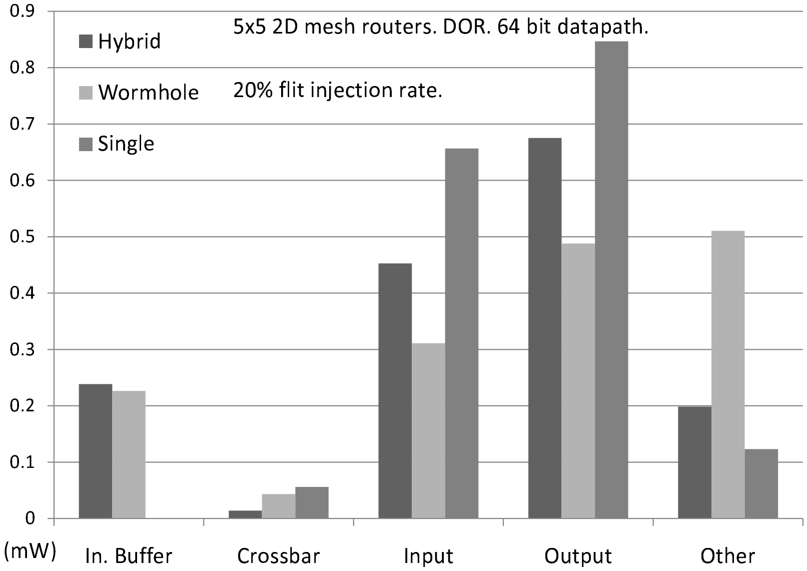
critical paths of the wormhole and hybrid EB-wormhole routers begin at the FIFO or EB read, go through the switch, and terminate at the output EB or register. The hybrid router occupies less area than the wormhole router because its FIFO, which is the dominant factor, is smaller by two slots. However, it requires more energy per bit compared to the wormhole router.

To further illustrate the FIFO overhead, Figure 3.37 shows the area and power breakdown for the wormhole and hybrid EB-wormhole routers. The single-stage router is included for comparison. The input module bars include routing computation, credit handling as well as the intermediate EB and associated logic for the hybrid router. The output module bars include the FFs or EBs between the crossbar and the channel. They also include credit handling logic located at the output side. All other control logic, including arbitration, is included in the "other" bars.

As shown, 56% of the area and 21% of the power of the wormhole router are in the FIFO. The FIFO in the hybrid EB-wormhole router constitutes 60% of the area and 17% of the power. The credit logic is the primary contributor for the increase in input and output area for the wormhole router. On the other hand, using two library



(a) Area breakdown after PnR.



(b) Power breakdown after PnR

Figure 3.37: Power and area breakdown after PnR.

Table 3.4: Gates, cells and nets after PnR for EB and wormhole routers. 64 bit datapath. 5×5 mesh routers with DOR.

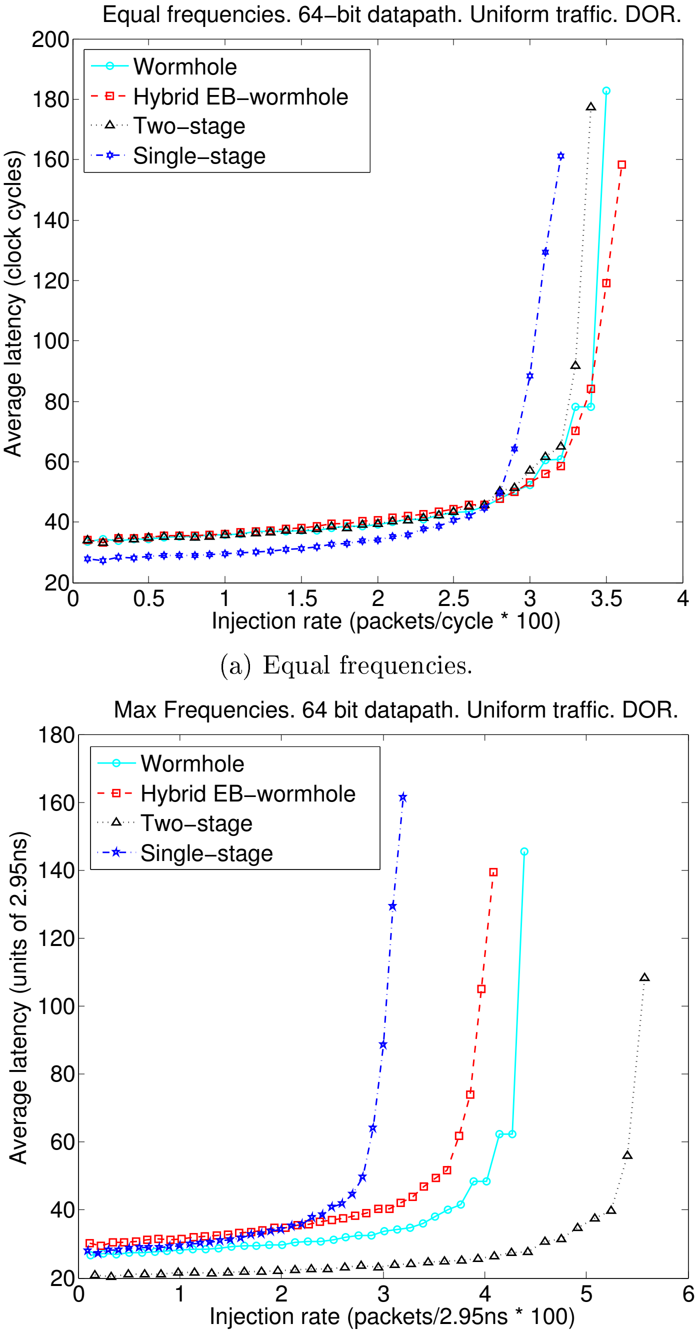
	Single-stage EB	Two-stage EB	Hybrid EB-wormhole	Wormhole
Cells	4499	6353 (+41%)	8977 (+100%)	8813 (+96%)
Gates	10073	14247 (+41%)	27710 (+175%)	34176 (+239%)
Nets	4080	5979 (+47%)	9249 (+127%)	8550 (+110%)

latch cells instead of a single FF cell causes the increase in input and output power for the hybrid EB-wormhole and EB routers.

Table 3.4 contains the cell, gate and net counts of the four routers with a 64 bit datapath, and shows the relative differences compared to the single-stage EB router. The input FIFOs lead to significant increases compared to the EB routers. The difference between the wormhole and hybrid EB-wormhole routers is not significant. However, this is not true for the number of gates because of the two extra FIFO slots and the credit handling logic in the wormhole router.

The PnR results highlight the adverse effects of the input FIFO in terms of area, energy and cycle time. This is especially apparent when comparing the two-stage EB and hybrid EB-wormhole routers because they differ only in that the input EB is replaced by a FIFO. This demonstrates that explicitly adding FIFOs in the wormhole routers carries a significant additional cost compared to using existing channel FFs as EBs.

Figure 3.38 plots injection rate versus latency for uniform traffic. It assumes an equal datapath width for all routers. For equal clock frequencies, the single-stage EB router has a 19% reduced zero-load latency compared to the other three routers. However, assuming that each router operates at its maximum clock frequency, the two-stage EB router has a 25% reduced zero-load latency—measured in absolute time—compared to the single-stage router. The wormhole and single-stage routers have comparable zero-load latencies.



(a) Equal frequencies.

(b) Maximum frequencies.

Figure 3.38: Injection rate vs. latency for EB and wormhole routers.

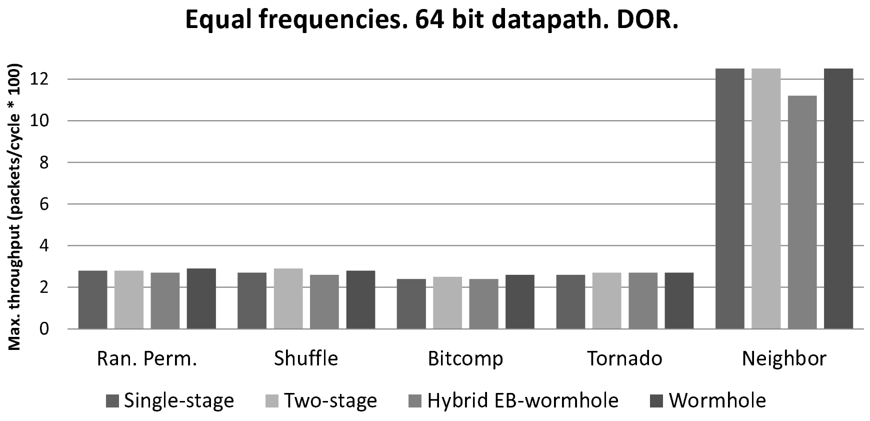


Figure 3.39: Maximum throughput by traffic pattern for EB and wormhole routers.

The maximum throughput of each router for a given datapath width and clock frequency is primarily affected by the number of available buffer slots. This accounts for the 13% and 6% increased throughput of the wormhole router compared to the single-stage and two-stage EB routers, respectively, when running at the same clock frequency. On the other hand, while both credits and ready signals have the same propagation delay in cycles, the effective buffer turnaround time is one cycle higher for the wormhole router as credits are consumed during the switch arbitration stage one cycle before the flit actually leaves the buffer; this artificially increases buffer occupancy. Furthermore, the EB routers effectively provide additional buffer capacity in the form of the output EB. Together, these factors increase the hybrid EB-wormhole router's maximum throughput beyond that of the wormhole router. If routers operate at their maximum frequencies, the two-stage router has a 27% and 36% higher maximum throughput—measured in absolute time—compared to the wormhole and hybrid EB-wormhole router, respectively. This illustrates that the difference in cycle time has a significant impact. However, this comparison does not take into account the impact on area and power.

To ensure that the four routers behave consistently under traffic patterns other than uniform random, we repeat the above experiment for each of the other five traffic patterns in our set. The results for maximum throughput are summarized in Figure 3.39. As shown, performance remains consistent with previous observations.

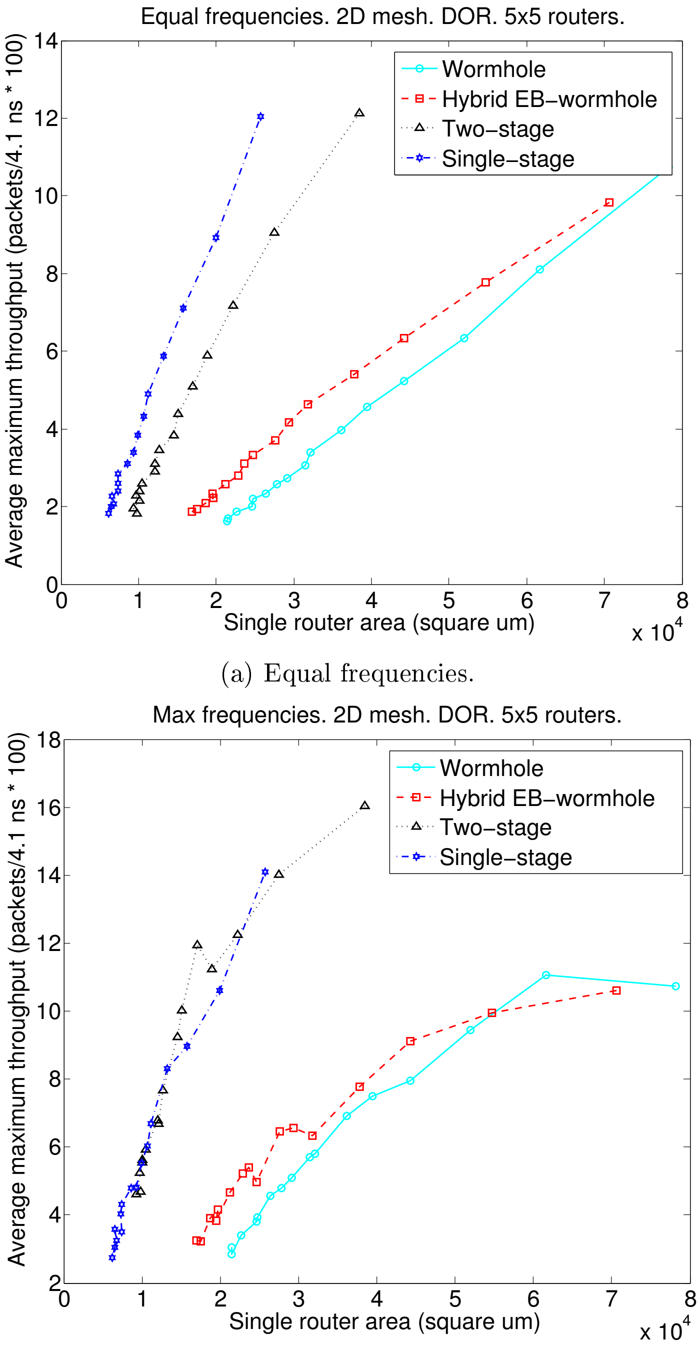
The same is true for latency as a function of the injection rate. Consequently, we can safely average between traffic patterns for the rest of this Section.

In order to perform a fair comparison of the efficiency of the four routers, we equalize throughput, area or energy by modifying datapath width. Therefore, the four networks will have different datapath widths, and consequently each packet will consist of a different number of flits.

For our first comparison, we equalize the maximum throughput. With equal maximum throughput and routers operating at their maximum frequencies, the single-stage EB router requires the same amount of energy to transfer a single bit from an input to an output as the wormhole router, and 17% less compared to the hybrid router. Due to the single-stage router's low energy overhead, it has a wider datapath but the wormhole and hybrid routers have more buffering slots. Therefore, even though FIFOs are more costly, in our network they provide twice as much buffering compared to the single-stage router. This increases the maximum throughput of the wormhole and hybrid routers.

The trends for router area are similar to those for energy. The Pareto-optimal curves relating maximum throughput and area are shown in Figure 3.40. When operating all routers at their maximum frequencies, time is given in units of the largest cycle time among all data points (4.1ns). In this case, the single-stage EB router requires 66% less area for the same throughput compared to the wormhole router, and 65% compared to the hybrid EB-wormhole router. If we equalize for router area or energy, the datapath of the single-stage router is wider than those of the wormhole and hybrid routers. As a result, the single-stage router offers 3% more throughput per unit energy compared to the wormhole router and 18% more compared to the hybrid router. The single-stage router also offers 67% more throughput per unit area compared to the wormhole router, and 62% more compared to the hybrid router. The trends hold when operating all routers at the same clock frequency. In this case, however, the single-stage router provides more throughput per unit area because the two-stage EB router no longer benefits from its higher maximum operating frequency.

In a Pareto-optimal point comparison with routers operating at their maximum frequencies, the single-stage EB router has a 3% increased zero-load latency for the

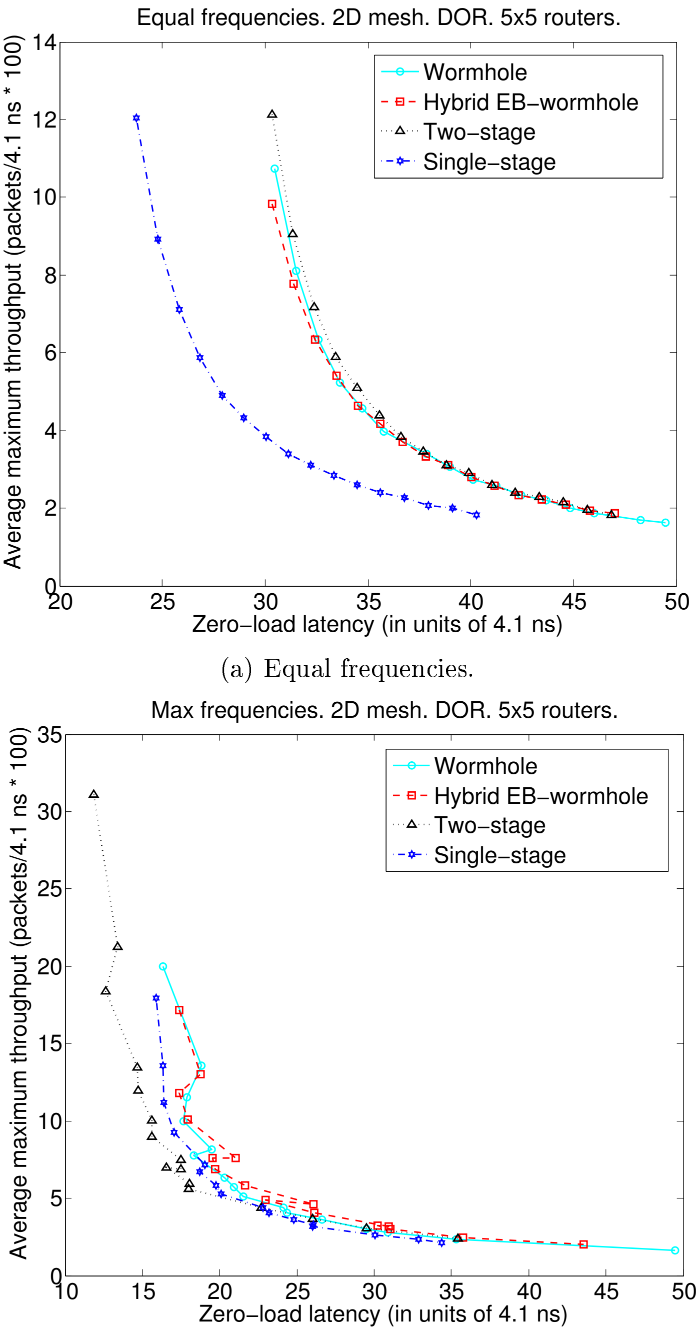

Equal frequencies. 2D mesh. DOR. 5x5 routers.

(a) Equal frequencies.

Max frequencies. 2D mesh. DOR. 5x5 routers.

(b) Maximum frequencies.

Figure 3.40: Injection vs. area for EB and wormhole routers.



(a) Equal frequencies.

(b) Maximum frequencies.

Figure 3.41: Throughput vs. latency for EB and wormhole routers.

same maximum throughput compared to the two-stage router. This result differs from Section 3.7.3 because our network for this comparison has two-cycle channels that are clocked at the same frequency as the routers, and thus favors the two-stage EB router. However, compared to the single-stage router, the wormhole and the hybrid EB-wormhole routers still have a 10% higher zero-load latency for the same saturation throughput. This is because in our 2D mesh with a large measured average hop count of 6.2, the difference in cycle time between the wormhole and hybrid routers on the one hand and the single-stage router on the other hand is not large enough to outweigh the latter's smaller pipeline depth. Therefore, while the two-stage router has a lower zero-load latency than the three other routers, the single-stage router is still preferable to the FIFO-based routers. The Pareto-optimal curves are shown in Figure 3.41. Note that these curves do not depict tradeoffs like the previous Pareto-optimal curves, because networks with wider datapaths also provide higher throughput and cost is not illustrated. However, these curves still compare maximum throughput for an equal zero-load latency, and zero-load latency for an equal maximum throughput.

With routers operating at the same clock frequency, the single-stage EB router leverages its reduced pipeline depth and offers lower zero-load latency for the same maximum throughput. The other three routers are comparable because they all have two pipeline stages; the slight differences in performance are due to different amounts of buffering provided by each and the resulting differences in maximum throughput.

We also compared a hybrid EB-wormhole router which has no intermediate EB and is based on the single-stage router. However, because of the significant timing overhead introduced by the FIFO and because of the lack of pipelining to split the critical path, this design had a 65% increased cycle time compared to the two-stage hybrid EB-wormhole design we evaluate in this section. Moreover, the addition of a FIFO defeats the primary advantage of the single-stage router: its design simplicity. Consequently, we did not investigate this particular design point further.

As discussed in Section 3.7.1, topologies with short channels provide less buffering opportunities to EB networks. However, we have shown that even in a topology with short channels such as our 2D mesh, EB networks are more efficient, and that adding extra buffering in the form of input FIFOs decreases network efficiency because the additional cost outweighs the extra throughput. Topologies with longer channels provide more buffering opportunities for EB networks, while the associated impact on cost is comparable for EB and VC or wormhole networks, with a constant buffer size. Also, longer channels require wormhole routers to use bigger input buffers in order to ensure that the longer credit roundtrip time can be covered. Hybrid routers, on the other hand, use the ready-valid handshake, and are thus not subject to this requirement.

Changing the network diameter or router radix will affect the fraction of the power and area that is consumed by the FIFOs, and will thus affect our comparison results. However, as long as FIFOs remain more expensive than EBs, EB routers will likely continue to be more cost efficient; otherwise, hybrid EB-wormhole designs will prove beneficial since they combine increased buffering with credit-less flow control and still take advantage of the FFs already present in the channel as distributed storage. The use of SRAM-based—rather than FF-based—FIFOs could also have impacted our results; however, for the FIFO sizes and technology library considered here, FF-based FIFOs are more area and energy efficient than compiler-generated SRAMs. Even for larger sizes, SRAM-based buffers may have larger power overheads than our results of 21% for the wormhole router and 17% for the hybrid router [54]. The exact SRAM implementation, e.g. compiler-generated vs. custom-designed, may also significantly affect the SRAM cost. However, we did not investigate the implementation of efficient custom SRAMs in this study.

Overall, our results show that the single-stage EB router is preferable to the two-stage EB, hybrid EB-wormhole and wormhole routers in terms of throughput efficiency, area and energy. The two-stage router provides slightly better zero-load latency; however, the single-stage router remains preferable to the wormhole and hybrid EB-wormhole routers in all aspects. This is true regardless of whether the routers operate at maximum or equal clock frequencies. Despite the single-stage router's larger cycle time, its area and energy savings can be traded for a wider datapath, enabling higher throughput and lower serialization latency. This highlights that design simplicity can result in decreased overhead and higher performance efficiency. Design simplicity also offers other advantages, such as reduced PnR flow turnaround time.

Thus, we have illustrated that EB flow control is beneficial compared to wormhole flow control, although the optimal type of EB router may differ depending on the network configuration. This clearly shows that the performance-over-cost efficiency gains of EB networks are mostly attributed to removing the input buffer overhead instead of simplying router design; however, simple router designs reduce cycle time which increases throughput in absolute time.

3.8 Summary

This chapter presented EB flow control. EB flow control uses the pipeline FFs in the channels for buffering flits instead of router buffers. Traffic separation is provided by duplicate physical channels, instead of VCs. Thus, switch and VC allocators are replaced by a switch arbiter for every output. This significantly simplifies router design. However, this option becomes inefficient for a large number of traffic classes. To make EB networks efficient for a large number of classes, we also propose a hybrid EB-VC router which has input buffers only used to drain flits in case of deadlock or heavy head-of-line blocking. Thus, in the common case hybrid routers operate as EB routers, and as VC routers otherwise.

By using the optimal EB router and shortest cycle time for each comparison, a 2D mesh EB network provides 43% more throughput per unit power, 22% more throughput per unit area or has an up to 45% shorter cycle time compared to a similar VC network. Gains for EB networks are proportional to the area and power cost of the buffers in VC networks. Design simplicity from removing VCs, allocators and credits primarily affects cycle time, but is not a major contributor to the area and power efficiency increase. This is illustrated by comparing EB routers to wormhole routers, which highlights that the dominant factor for EB flow control's performance-over-cost increase is the removal of the input FIFOs. Hybrid EB-VC routers offer 21% more throughput per unit power than VC routers, and 12% more than single-stage EB routers, because input buffers are used only to resolve deadlocks and alleviate head-of-line blocking. However, hybrid EB-VC routers carry the area and timing overheads of input buffers.

EB flow control is more efficient than VC and wormhole flow control even in a topology with short channels such as our 2D mesh. Topologies with longer channels will provide more buffering in the EB channel, while affecting the cost of the EB and VC or wormhole networks similarly.

EB routers are always more efficient than VC and wormhole routers. The choice of EB router among the single-stage EB, two-stage EB and hybrid EB-VC described in this paper should be based on design constraints. Table 3.2 identifies the optimal EB router without input buffers and VCs, depending on design priorities. To provide traffic classes, EB networks without input buffers and duplicate physical subnetworks should be considered first, because of their small area and complexity. The number of traffic classes above which this option becomes infeasible depends significantly on a variety of factors and therefore should be studied for specific chip designs and implementation technologies. If more classes than that number are desired, the hybrid EB-VC router should be used. Finally, note that the hybrid EB-VC router can be more energy efficient compared to EB routers without input buffers, because input buffers alleviate head-of-line blocking. Therefore, designs focusing on energy with few constraints on other cost factors should consider the hybrid EB-VC router.

EB flow control is a simple and elegant solution to the increasing buffer costs; it provides buffering in the network without the timing overhead and cost of router buffers, and without the complications of bufferless flow control explained in Section 2. Moreover, router simplicity has numerous advantages, primarily cycle time. EB flow control is applicable to a wide range of systems and consistently provides higher throughput per unit cost compared to VC and wormhole networks.

Chapter 4

Packet Chaining

The performance of a network is sensitive to the quality of the allocators used in routers. Allocators are responsible for generating grants to requesting packets so that they cross the switch without conflicting with other packets [24], as well as to generate grants to assign virtual channel (VC) to incoming packets [22, 85]. To maximize throughput, allocators strive to maximize the number of packets traversing each router in each cycle, while taking care of other considerations such as fairness and quality of service. The performance of a network-on-chip (NoC) is extremely sensitive to the matching efficiency of the allocators. Due to their complexity, allocators are in the critical path of many routers [10]. Therefore, the choice of allocator is a tradeoff between matching quality and timing and cost overhead. More complex allocators maximize throughput but prolong the allocation timing path, as well as require more area and energy. While this is a minor consideration for off-chip networks, such as datacenter or supercomputer networks, which use dedicated router chips and thus are hardly constrained by the cost of the allocators, this tradeoff is critical for on-chip networks. This chapter addresses this by proposing a novel technique to increase allocation efficiency without lengthening the allocation timing path.

Many NoCs use separable allocators [24] which use separate input and output arbiters in sequence to perform allocation. iSLIP separable allocators [74] use round-robin arbiters and update the priorities of each arbiter when that arbiter generates a winning grant. Thus, in each cycle, input (output) arbiters prioritize outputs (inputs) that are likely not to be prioritized by other input (output) arbiters. This way, fewer arbiters grant the same input or output port. Separable allocators are often used because they can operate within an aggressive cycle time. However, making arbitration decisions independently at each port degrades matching efficiency. While a separable allocator's matching efficiency can be increased by performing multiple iterations, this is typically not feasible within a single clock cycle because adding iterations multiplies the timing path by the number of iterations [10, 24]. Wavefront [103] and augmenting path [31] allocators also increase matching efficiency by guaranteeing maximal and maximum matchings, respectively. A matching is maximal if an input-output pair cannot be granted without releasing an input-output pair which was already granted; maximum matchings are matchings which grant the maximum number of packets possible. These guarantees increase the complexity of the allocator and thus delay and cost [10, 46], making such allocators infeasible within a tight timing budget.

To provide the efficiency of multi-iteration allocation without extending cycle time, past work has proposed pipelined [41] and incremental [84] allocations. In both schemes, allocation extends over multiple cycles, during any of which new requests can be added. In pipelined allocation, results are only available at the end of the last iteration. In contrast, incremental allocation makes the results of each iteration available, such that intermediate grants can be generated. Incremental allocation has been implemented using a separable, single-iteration allocator and holding resources (inputs and outputs) granted for the duration of a packet [84, 63]. This excludes the granted resources from future allocation, thus functioning similarly to subsequent iterations of a multi-iteration separable allocator which also do not consider resources granted in previous iterations of the same cycle. However, because resources are held for the duration of a packet, both schemes provide no benefits to single-flit packets, and small benefits to short packets.

Many systems, such as typical cache-coherent chip multiprocessors (CMPs), send primarily short packets. For instance, 53% of the packets in the applications we simulated in this chapter were single-flit and received no benefit from incremental allocation. Short packets are important because they transfer time-critical control messages. CPUs and cache blocks frequently stall waiting for those control messages;

thus, these control messages in many cases directly affect average memory access latency. Short packets are also challenging for the network because they stress allocators due to increasing the number of head flits which correspond to new allocator requests even with incremental allocation. Long packets are affected by short packets because there still are many requests to the switch allocator every cycle since short packets reserve resources for very short periods of time (unless they are single-flit in which case they do not reserve resources), causing inefficient matchings, and also because long packets can be blocked behind short packets in the same VC. In addition, long packets can starve other packets [84, 63].

To increase allocation quality without extending cycle time, in this chapter we introduce *packet chaining* [78, 79], a method for improving allocation efficiency for iterative allocators that is particularly suited to networks with short packets and short cycle times.

4.1 Detailed Description

4.1.1 Conventional Allocation with Short Packets

With short packets, a conventional single-iteration separable allocator gives poor matching efficiency because it is frequently restarting the allocation process and in a single iteration is not able to compute an efficient matching. While multiple iterations or more complex allocators could improve matching efficiency, they are typically not feasible within a tight timing budget.

The poor allocation efficiency of a conventional iSLIP allocator in the extreme case of single-flit packets is illustrated in Figure 4.1. The figure shows three cycles of allocations for a 6×6 router. Each input port has four VCs each containing a single one-flit packet. Each packet is labeled with the output port it requires. No additional packets arrive over the three cycles illustrated. Dark squares denote both requests and grants generated by the input and output arbiters. With input-first allocation, an input arbiter first selects one request from each row (input grants) and then an output arbiter selects one surviving request from each column (output grants). After

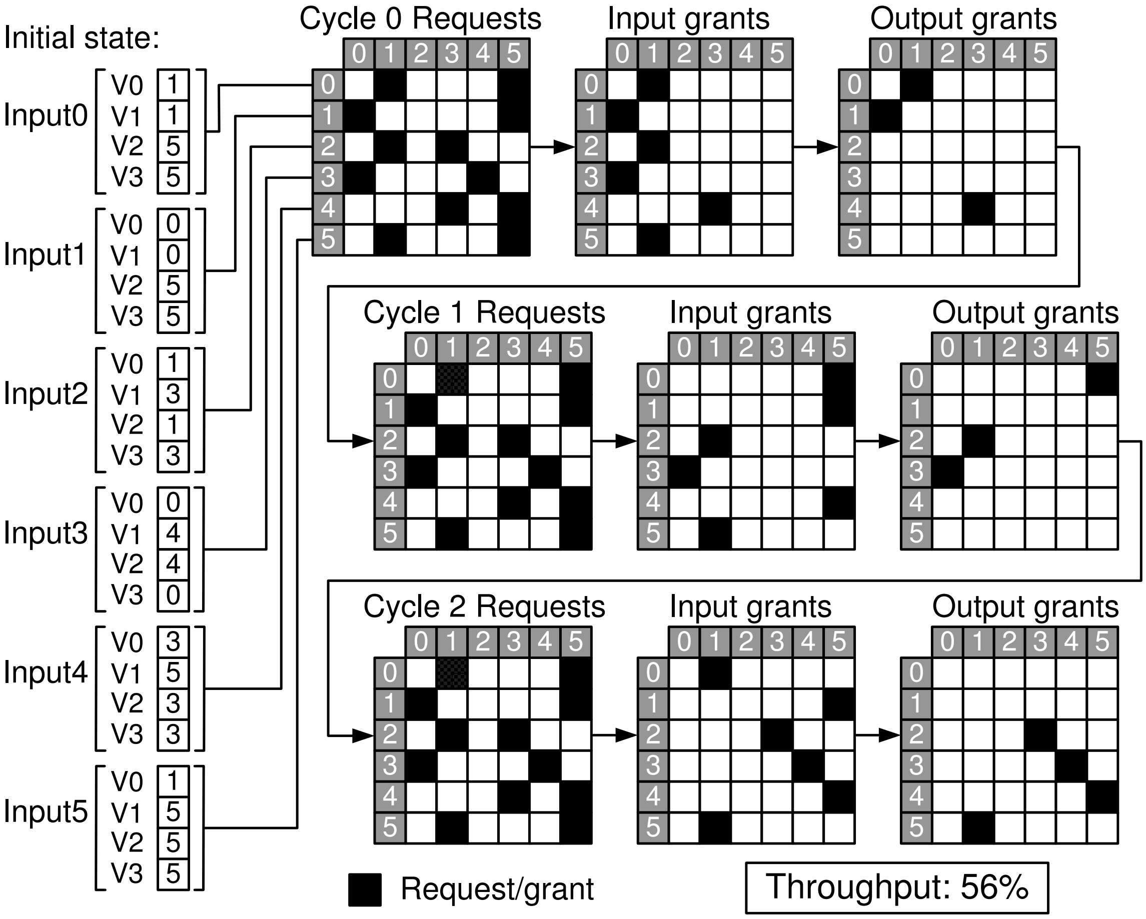


Figure 4.1: Example allocation of iSLIP without packet chaining.

a single iteration the resulting allocation is poor—with just three grants out of a possible five.

The situation is repeated in cycles 1 and 2, but with the iSLIP allocator rotating the input and output arbiter priorities for ports that are granted [74]. Outputs are left idle because many input arbiters picked the same output, which can only serve a single input at a time. If there was time for multiple iterations, these idle outputs would be connected to unmatched inputs.

Figure 4.3(a) shows the activity on the output channels. For a cycle in which an output is busy, a dark rectangle is labeled with the input and the VC that is using that output. Output 2 is idle for all three cycles because no packet requests it. There are five other idle output cycles. A better allocator could fill most of these cycles resulting in higher throughput.

4.1.2 Packet Chaining Description

Packet chaining [78, 79] performs more efficient allocation without increasing cycle time by starting with the existing set of connections and holding any finishing connections that can be used by waiting packets. Packet chaining in effect *chains* packets together, even if they are from different inputs or VCs, so they look like one longer packet to the switch allocator. The switch allocator does not start from scratch, but from this initial state of chained connections. The result is comparable to running multiple iterations of a conventional allocator, spread over an equal number of cycles and independent of packet length.

Packet chaining finds a new packet, potentially from any input and VC, destined to the same output to chain onto a departing tail flit. A new waiting packet is suitable if (a) it has been routed to the same output as the tail flit, (b) there is a free output VC it is eligible to use, and (c) there is at least one credit for that output VC. The chained packet need not be at its start; partially transmitted packets can be chained, in which case the only eligible output VC is the one to which the packet is already assigned, as stored in control state logic of input VCs.

Figure 4.2 illustrates the same example as Figure 4.1 with packet chaining. In this figure, an X denotes a connection during the previous cycle and a dot (•) denotes requests and grants. In cycle 0, the allocator starts with five connections inherited from cycle -1. Three of these connections are reused by new packets requesting the same output and thus chaining onto the departing packet as denoted by a square in the cycle 0 request matrix with both an X and a •. The other two connections, denoted by an X without a •, are terminated because no packets request the connected output. The three chained packets eliminate competing requests for the same input and output ports before the input arbiters. The result of arbitration at the inputs shown includes the three chained packets as well as additional requests. Only one of the new requests is granted by the output arbiter, giving four packets transmitted in

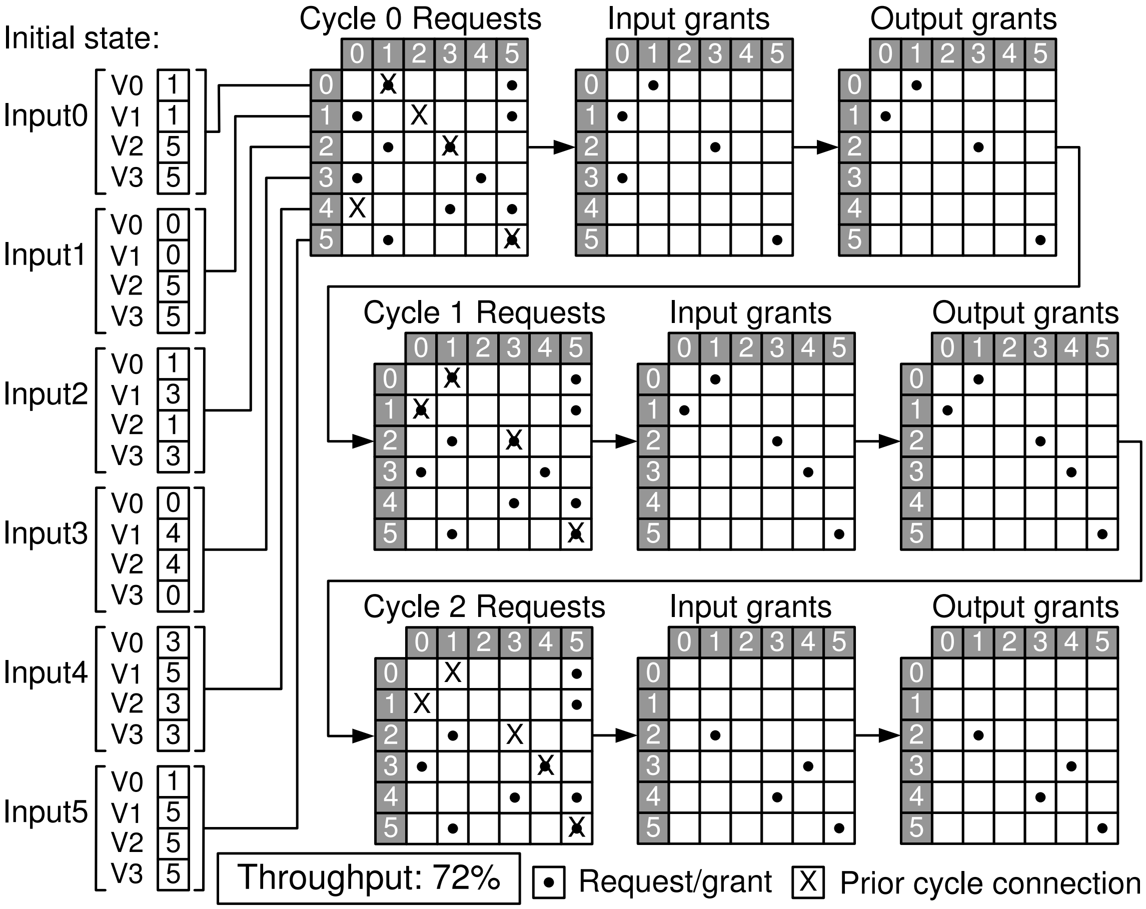
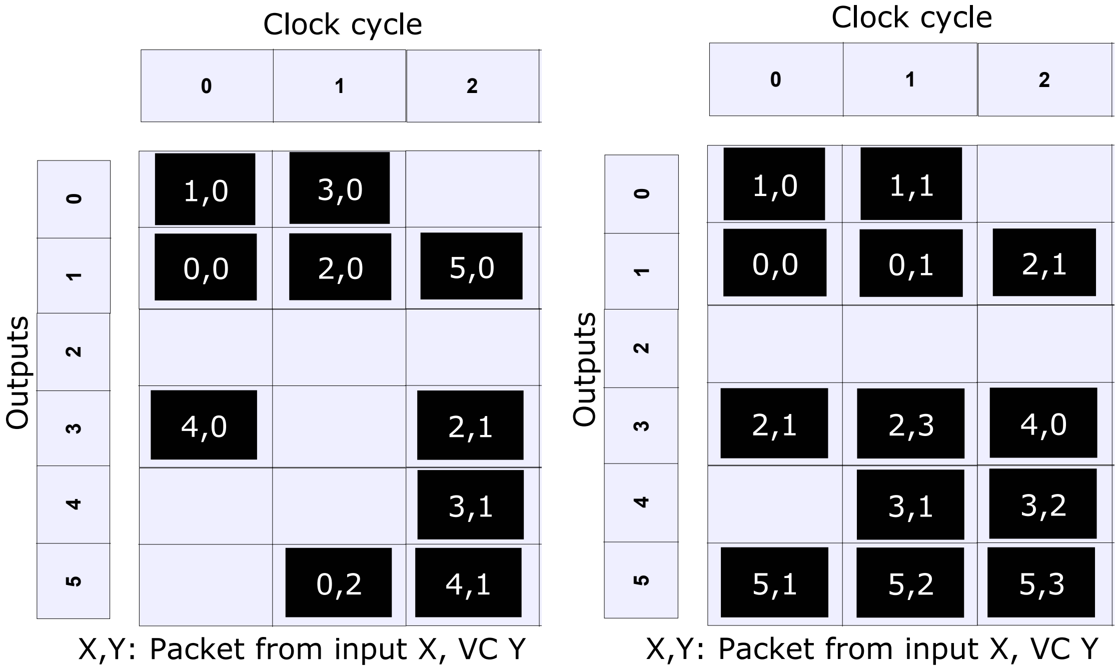


Figure 4.2: Example allocation of iSLIP with packet chaining.



Outputs	0	1	2
0	1,0	3,0	
1	0,0	2,0	5,0
2			
3	4,0		2,1
4			3,1
5		0,2	4,1

(a) Without packet chaining.

Outputs	0	1	2
0	1,0	1,1	
1	0,0	0,1	2,1
2			
3	2,1	2,3	4,0
4		3,1	3,2
5	5,1	5,2	5,3

(b) With packet chaining.

Figure 4.3: A timeline of output port usage using the same example.

cycle 0.

As shown in the second row of Figure 4.2, all four connections from cycle 0 are chained in cycle 1. The allocator makes one additional grant giving a total of five packets transmitted in cycle 1. In cycle 2, only two of the five connections are chained and the allocator makes two additional grants, resulting in four packets being transmitted.

Figure 4.3(b) shows the activity on the output channels. Other than output 2 being idle because there are no requests for it, the figure shows two other idle output cycles, compared to five for allocation without chaining. Of these two idle cycles, only the first—on output 4 in cycle 0—is avoidable. A better allocator could have assigned (3,1) or (3,2) to this channel. An idle cycle on either channel 0 or channel 4 in cycle 2 is unavoidable since only input 3 has requests for these two outputs. The allocator thus generates maximum matchings in cycles 1 and 2.

Overall, packet chaining increases the quality of allocation, resulting in more packets being sent (13 vs. 10 in this example). This is accomplished by reusing existing connections where possible rather than returning the inputs and output ports to the switch allocation pool where they run the risk of being idled due to inefficient allocation as described in Section 4.1.1. This is equivalent to performing multiple allocator iterations, one per cycle, while at the same time using the results of each iteration, as in incremental allocation [84] but independent of packet length. Packet chaining does not always result in maximal or maximum matchings. For example, an additional allocator iteration would add a grant from input 3 to output 4 in cycle 0. Note that even with uniform random traffic, a significant number of packets request the same output at every router and thus can be chained.

Packet chaining and incremental allocation provide little benefit to allocators whose matching does not improve with multiple iterations. In those cases, packet chaining would reserve resources for the duration of the packet, which would decrease the allocation problem. For allocators which provide higher-quality matchings for fewer inputs and outputs, packet chaining and incremental allocation would provide a small benefit.

Connections are released if they cannot be used productively either because the output VC has no more credits or the input VC becomes empty [63]. Therefore, resources do not remain reserved if they cannot be used, such that output under-utilization is avoided. To accommodate higher-priority traffic, a connection is released if a higher-priority request exists for the connected output. This is done to avoid adversely affective flows with higher-priority, for example as part of a quality of service (QoS) scheme. Chained packets may bypass older packets residing at another VC, but this is also possible without packet chaining if there is more than one VC. Therefore, packet chaining does not cause out-of-order delivery of packets or flits if they would be ordered without packet chaining.

We implement packet chaining on top of a combined switch-VC allocator [63] that reserves output VCs only for packets that win switch allocation. This leaves more output VCs free compared to performing VC allocation in advance, therefore giving more flexibility to packet chaining to find free output VCs. This makes successfully chaining packets more probable. However, packet chaining may also be used in a router with a VC allocator [85].

4.1.3 Chaining Variations

We consider three variations in the set of inputs and VCs that are considered for chaining:

Same input VC : The simplest scheme is to consider only the same input VC as the previous packet that used the connection.

Same input, any VC : This scheme considers all eligible VCs of the same input as the previous packet that used the connection. This multiplies the probability of finding a suitable packet by the number of VCs.

Any input, any VC : This scheme considers eligible packets in any input and any VC. Thus, this scheme increases the probability of finding a packet to chain by the number of other inputs and VCs.

The complexity of the chaining logic depends on the packet chaining scheme. Considering only the same input VC requires just a comparator to check if the next packet requests the same output as the departing tail flit. Considering all VCs of the same input requires a similar comparator for each input VC as well as an arbiter to select among eligible input VCs. Finally, the scheme with the most chaining candidates (any input and any VC) requires a complete and separate packet chaining (PC) allocator similar to the switch allocator. However, all schemes require the same logic to check for active connections, output VCs and credits. Regardless of complexity and chaining scheme, chaining is performed in the PC stage in the manner described below. Also, packets are chained in the same way regardless if they are from the same or another input, as well as the same or another VC.

4.1.4 Packet Chaining Pipeline

Packet chaining adds an extra PC stage to a conventional two-stage VC router with look-ahead routing [32]. Newly arriving packets skip the PC stage and start directly in the switch allocation (SA) stage. Hence, adding the PC stage does not increase router latency. Packets that are not eligible for SA remain in the PC stage and participate in PC allocation. Packets participate in PC allocation if there is an active connection they can use which will be released in the next cycle (the tail flit will be traversing the switch in the next cycle). Packets participate in SA if they are at the head of their input VC and their input and at least one of their desired outputs (in case of packets requesting multiple outputs) is not currently connected. As discussed below, packets in the SA stage may participate in both PC and SA. Packets that get chained advance to the SA stage or remain in it if they were already there in the next cycle because they need to remain behind the preceding tail flit; however, they do not participate in SA. Packets that receive a switch grant advance to the ST (switch traversal) stage.

Figure 4.4 shows a waiting packet X that shares its output with PT (a preceding tail flit). When PT is in the SA stage, requests are submitted to the PC allocator for waiting packets eligible for chaining, to reuse PT's connection. In this example, packet X is granted the connection from the PC allocator during cycle 0. If PT fails SA during cycle 0, PC allocation is cancelled and packets X and PT remain in their

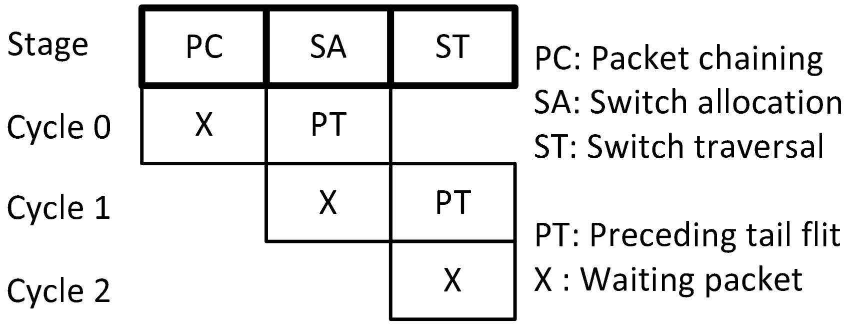


Figure 4.4: An example of flit X being chained to use PT's connection.

respective stages—repeating both allocations in the next cycle. In this example, PT receives a switch grant and traverses the switch during cycle 1 while the head flit of X advances to the SA stage. Since X was allocated the connection during the PC stage, it does not participate in SA during cycle 1. The established connection blocks competing packets from X's input and output. Look-ahead routing is performed for packet X during the SA stage. Finally, in cycle 2, the head flit of X traverses the switch. In our latency-optimized two-stage router pipeline, chained flits may not skip the SA stage even if no flit is ahead of them in the ST stage, because that would require a separate VC allocator and would complicate timing with the input channels, buffers and routing logic.

Because packet chaining operates when PT is in the SA stage, it is guaranteed to chain an eligible packet and remove competing packets from consideration by the switch allocator. Biasing the switch allocator to favor maintaining connections across packets does not achieve the same end because competing packets are not removed from consideration, and thus may impact switch allocator decisions negatively.

Because PC allocation considers only outputs currently connected (which will be released in the next cycle) and connected outputs are not eligible for SA, outputs may not participate in both PC and switch allocation. For the same reason, inputs may not participate in both allocations with chaining schemes which consider only the same input, because with those schemes only a single output—that of the previous packet holding the connection—is considered. However, when considering any input for chaining, an input VC may participate in both allocations only if the packet at its head requests two or more outputs, and one is unconnected while the other is connected and available for chaining. Similarly, an input may participate in both allocations as long as it has multiple VCs, because the packets at the heads of that input's VCs may request different outputs. Therefore, conflicts may arise between the two allocators. If the two allocators grant the same input (regardless if they grant the same input VCs), the PC allocator's decision is disregarded and the connection is released enabling packets to bid for that output through the switch allocator.

Eligibility to participate in PC and switch allocation is determined at the beginning of the cycle. However, the eligibility of a packet for chaining may depend on switch allocator decisions in the same cycle as PC allocation. For example, an input VC may contain a packet eligible for chaining except that the input port which contains that VC is part of another connection to another output which has to be released in order for that packet to be chained. Similarly, a tail flit for which there is no connection needs to be granted by the switch allocator in order to form a connection and provide a chaining opportunity for other packets. In each of those two cases, a packet will become eligible for chaining only by a favorable switch allocator decision in the same cycle. Packets in those two cases generate a lower-priority speculative request to the PC allocator. This way, PC allocator requests which may later have to be invalidated do not take resources away from packets which are definitely eligible for chaining. However, sub-optimal chaining decisions are still possible because only some of the lower-priority requests may actually become eligible, but other lower-priority requests may have been granted instead.

For input VCs participating in SA, PC allocator requests are generated based on the flits *behind* the flits at the head of the buffers. Because eligibility for chaining depends on the flits at the head departing, those PC requests are also marked as lower-priority. Enabling this functionality as well as the other lower-priority PC requests described previously is not required for packet chaining and represents a tradeoff between complexity and the number of chaining candidates. These speculative lower-priority requests are not essential for packet chaining and increase complexity, but also increase the number of chained packets per time. Other types of priorities, such as age priorities, are taken into account within each of the two aforementioned priority classes of PC requests.

4.1.5 Starvation Control

Packet chaining intensifies the fairness and starvation issues of incremental allocation [84, 63] because a connection can remain active indefinitely, since it is no longer limited by packet length. To provide limited fairness, we extend packet chaining with starvation control. Starvation control uses age to increase a waiting packet's priority. Since higher-priority requests cause established connections to be released (potentially mid-packet), starvation is prevented. Age should be increased after a predetermined number of cycles that is large enough to preserve the benefits of packet chaining. A simpler alternative is to release a connection and inhibit packet chaining for the affected input and output if a connection has been held for more than a maximum number of cycles. With this mechanism, connections that will reach the starvation threshold at the next cycle are not eligible for chaining. Thus, switch ports held by a long series of packets are returned to the switch allocator pool to be reassigned to waiting packets. The latter mechanism does not require multiple priority levels and thus reduces PC allocator complexity. However, it may not adequately prevent starvation in the rare scenario where a connection is released after reaching the threshold but it keeps being re-established because the other packets that are waiting for the same output and are getting starved cannot request that output because they are in an input which participates in another connection every time the connection causing the starvation is released. In this case, the starved packet's outputs are always reserved when their own connections are released and thus would be able to bid for their outputs. In practice this scenario can only occur indefinitely for traffic patterns that are specifically constructed such that when starvation control releases a connection, all other packets that are requesting the released output are ineligible because their input is currently participating in another connection. This scenario is extremely unlikely in practice because exploiting this weakness requires precise timing, and also because at low loads input VCs will eventually be left without flits and at high loads output VCs will run out of credits, thus releasing the connection. Therefore, the simpler mechanism is adequate unless sustained adversarial traffic patterns are probable.

4.2 Methodology

Evaluation is performed with a modified version of Booksim [24]. The topologies we use are an 8×8 2D mesh and a 4×4 2D flattened butterfly (FBFly) [57]. Routers are connected to one network terminal in the mesh, and four terminals in the FBFly. Therefore, each FBFly router has 10 ports. In the mesh, all channels have one cycle delay. In the FBFly, injection and ejection channels have a delay of one cycle, whereas short, medium and long channels have two, four and six cycles delay, respectively. For the mesh we use deterministic dimension-order routing (DOR) because it is a simple and popular choice. For the FBFly we use universal globally adaptive load-balancing (UGAL) routing [100].

We use VC flow control [22]. Routers use the pipeline described in Section 4.1.4 and operate at the same clock frequency in all comparisons. Routers require two cycles to generate and transmit credits upstream. In our evaluation, iSLIP [74] and wavefront [103] allocators take into account priorities. The PC allocator uses iSLIP [74] with one iteration (iSLIP-1) because a more complex PC allocator would lengthen the allocation timing path in a router with an iSLIP-1 switch allocator. Unless indicated otherwise, the combined switch/VC allocator also uses iSLIP-1. All separable allocators in our study perform input arbitration before output arbitration. Incremental allocation [84] is used when evaluating networks without packet chaining.

In all our results, the reported throughput is the minimum throughput obtained among all the sources (worst-case throughput); that throughput is a better indication of network performance because, depending on the traffic pattern, some flows may face no contention and achieve full throughput whereas other sources may face severely congested paths. In a CMP, those sources under congestion will likely eventually become the bottleneck and force the other sources to be throttled.

Evaluation is performed using uniform random, random permutation, shuffle, bit complement and tornado traffic patterns [24]. The FBFly also uses transpose and neighbor traffic. These two traffic patterns provide little insight for the mesh due to the absence of concentration. Packet length varies from 1 to 16 flits. Injection rates are given in flits. Our evaluation begins with single-flit packets which clearly illustrate the effect of packet chaining, and then proceeds to discuss multi-flit packets and bimodal traffic. Our default configuration has 4 VCs, with 8 buffer slots statically assigned to each. VCs in the FBFly are divided among the two traffic classes required by UGAL. In the default configuration, all packets have equal priority and starvation control is disabled.

We also present execution-driven simulation results for a typical cache-coherent CMP with 64 superscalar, out-of-order RISC CPUs. The CPUs are two-way multithreaded and allow a large number of outstanding memory requests. We use five PARSEC [12] benchmarks and a parallel implementation of FFT from SPLASH2 [111]. The benchmarks are configured to create two threads per CPU. We use a custom, detailed, and timing-accurate CMP simulator that does not simulate the operating system and that interfaces with Booksim. The CMP simulator has an execution-driven front-end and a performance-modeling back-end. The simulator models detailed temporal effects and performance characteristics of the simulated hardware, and influences the front-end in a realistic way. The front-end uses Pin [72] to instrument a native x86 multithreaded binary. The front-end passes RISC-like instructions to the back-end.

For the application simulations, we use the previously described mesh network, packet chaining among all VCs of the same input, and a 64-bit wide datapath. Therefore, short packets are single-flit, while packets carrying our 32-byte cache lines have five flits. For fairness, connections are released if they have been active for eight cycles. L1 caches are 8KB, four-way set-associative, have a single cycle of latency and are private to the cores. L2 caches are shared, non-inclusive (they act as victim caches for the L1s), four-way set-associative, have 32KBs per core, and have five cycles of latency. There is one directory and one L2 cache slice located at each core. There is one memory controller in every network quadrant. We assume cores optimized for clock frequency that are clocked at a four times higher clock frequency than the network. We use instructions per cycle (IPC) to measure application performance. Results for IPC correspond to those for execution speedup because packet chaining does not

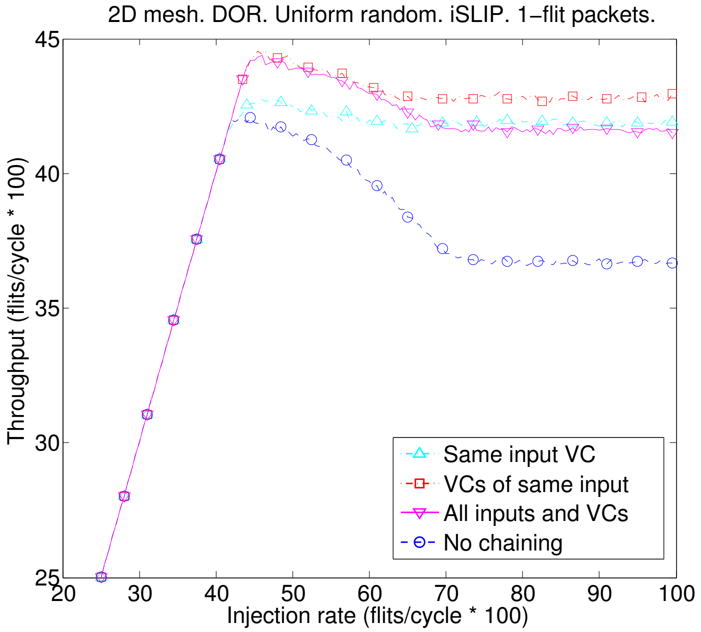


Figure 4.5: Increasing the injection rate beyond saturation illustrates network instability.

affect instruction scheduling or the instruction stream. This has been confirmed with preliminary simulations.

4.3 Evaluation

4.3.1 Throughput Under Heavy Load

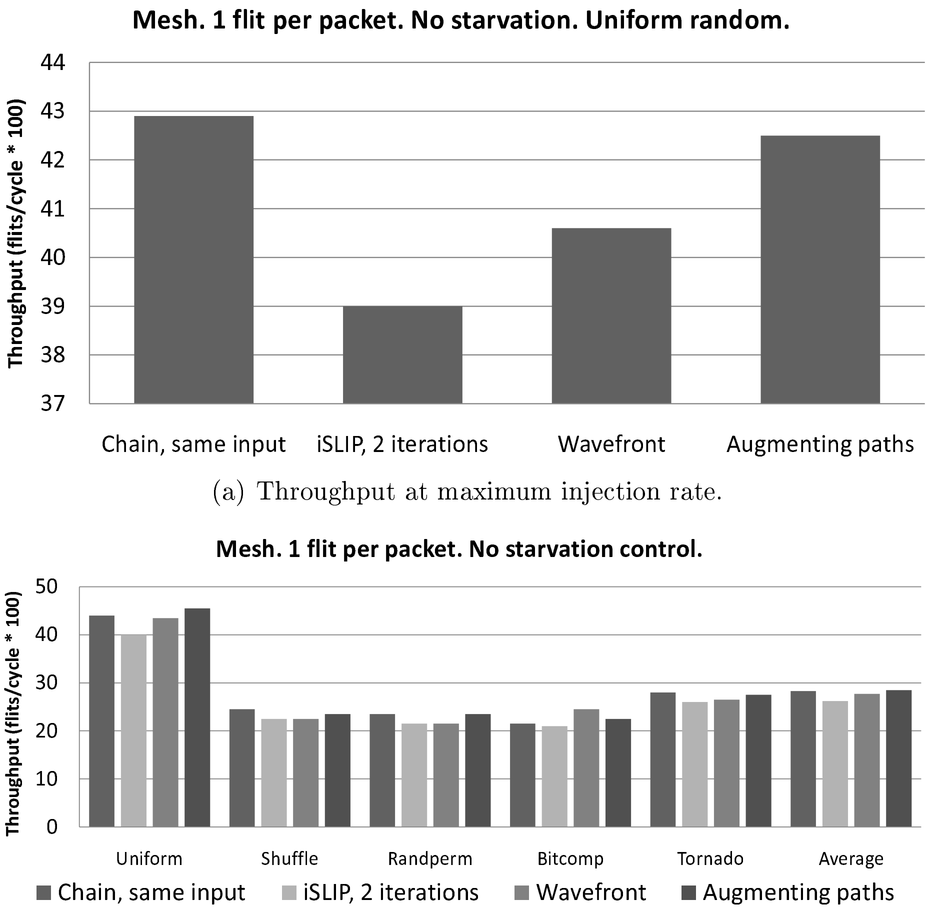
Packet chaining stabilizes the network by reducing throughput degradation past saturation. As illustrated in Figure 4.5, compared to iSLIP-1 (incremental allocation without packet chaining) with single-flit packets, packet chaining increases throughput at maximum injection rate by 15% when considering all VCs of the same input. The throughput illustrated is the minimum among all sources. Throughput peaks at saturation injection rate and then decreases because of multi-hop paths of congested packets forming due to a few hot spots, known as tree saturation [62, 99]. These congestion paths are detrimental to network performance because they block packets from other sources and other destinations. Tree saturation still forms with packet chaining because the PC allocator allocates only with local knowledge, but is less pronounced due to the increased allocation efficiency. Mitigating tree saturation requires some global allocation scheme, such as age-based allocation [68]. With packet chaining, throughput drops only marginally (2.5%) past saturation, which illustrates that the increased allocation efficiency greatly reduces tree saturation.

Packet chaining does not eliminate instability due to globally unfair allocation. For instance, traffic patterns such as transpose and shuffle that cause instability in mesh networks due to the parking-lot problem [24], where some communicating pairs have to compete for resources more times than others and thus get a smaller share of the throughput, are not stabilized by packet chaining. In such patterns, stability can be increased by age-based allocation or starvation control. The FBFly is stable both with and without packet chaining because with minimal routing all packets reach their destinations in two hops. Therefore, the parking-lot problem does not appear and tree saturation is not significant because depending on their destinations, packets can use long links to bypass congested areas.

Performance at maximum injection rate is an important metric because a throttling mechanism might be overly conservative thus reducing available throughput, while the lack of a throttling mechanism may place the network in the instability region. In both cases, the system cannot make use of the maximum throughput the network is capable of at its saturation point. Without elaborate throttling, it is very difficult to consistently operate a network at the point of saturation. Thus, in most systems, network-limited phases of applications operate past saturation where throughput at maximum injection rate dictates performance. This is especially apparent in throughput-oriented CMPs [49].

4.3.2 Comparing with Other Allocators

Figure 4.6 shows that packet chaining offers comparable or higher throughput than three other popular and more complex allocators: wavefront, iSLIP-2 and augmenting path. iSLIP-2 refers to iSLIP with 2 iterations in the same cycle. Additional



(a) Throughput at maximum injection rate.

(b) Saturation throughput.

Figure 4.6: Comparison of packet chaining with other allocators.

iterations do not considerably improve performance but increase cycle time [74, 24]. Wavefront guarantees maximal matchings [103] at the expense of prolonging the allocation timing path. This is intensified for high-radix routers and makes wavefront allocators reasonably feasible only in small configurations or with full-custom implementations. As an example, wavefront consumes up to 6× more power and has an increased delay by 36% compared to a separable allocator in a FBFly, and 3× more power and 20% more delay in a mesh [10]. Augmenting path allocators generate maximum matchings but are too costly for single-cycle implementations [46]. They locate all paths from unmatched inputs to unmatched outputs in the directed bipartite allocation graph [31]. These three allocators, especially augmenting path, are more costly primarily in cycle time compared to iSLIP-1 with packet chaining. They are used to show that packet chaining improves allocation efficiency without the associated cost of more complex allocators.

Figure 4.6(a) shows that at maximum injection rate and when considering all VCs of the same input, packet chaining provides a 10% higher throughput compared to iSLIP-2 and 6% compared to wavefront. Furthermore, packet chaining offers comparable throughput (1% more) to an augmenting path allocator. While an augmenting path allocator guarantees maximum matchings, it optimizes throughput only locally and does not take into account fairness. Thus, requests get passed over as long as selecting them prevents a maximum matching; there is no guarantee that those requests will ever be granted. Therefore, packet chaining is able to offer a slight throughput increase at high loads where the fairness issues with augmenting path lead to increased instability. In addition, packet chaining provides a 22.5% lower average latency than the other allocators—computed as an average from low to saturation injection rates. That percentage becomes 30% for injection rates from 20% to saturation, because at low loads allocators are not stressed and thus latency is comparable for all four allocators.

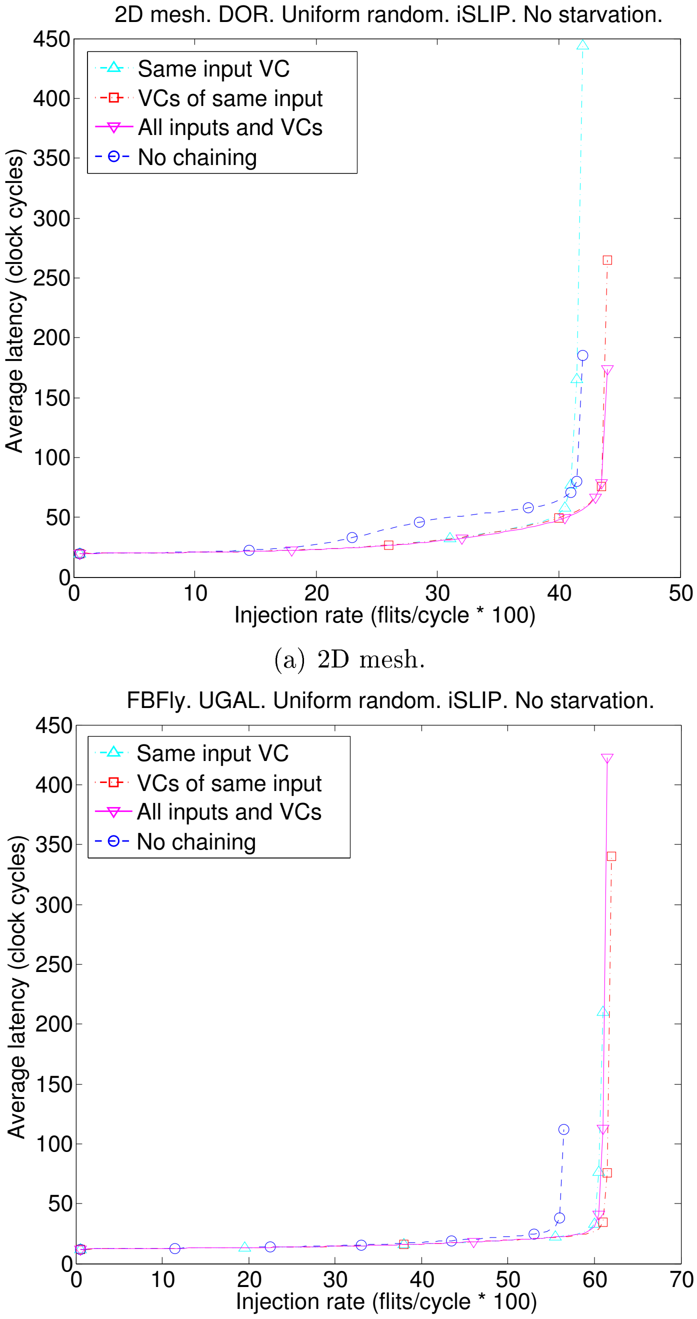
Figure 4.6(b) illustrates the saturation throughput of packet chaining in five synthetic traffic patterns. On traffic other than uniform random, packet chaining provides a 4% to 9% higher saturation throughput (5% by average) compared to iSLIP-2 and wavefront, whereas it is comparable to an augmenting path allocator. These percentage gains increase when evaluating performance at maximum injection rate. The differences between allocators with these traffic patterns are smaller by average compared to uniform random traffic. Uniform random traffic stresses allocators because requests may show up from any input to any output. In contrast, other traffic patterns use only a subset of the inputs and outputs in each router and therefore enable less complex switch allocators to provide efficient matchings, because the allocation problem in each cycle is smaller.

In the FBFly, with single-flit packets and traffic patterns other than uniform random, packet chaining offers a 3% higher saturation throughput than each of the other allocators when selecting among all inputs and VCs. With uniform random traffic, throughput is comparable with an augmenting path allocator, and 3.5% higher compared to iSLIP-2 and wavefront. Finally, average latency is 2%-5% lower with packet chaining. While the relative differences between allocators are the same as the mesh, in the FBFly percentage differences are smaller because packets take fewer hops and therefore bid for the switch less often than in the mesh.

The trends remain the same with longer packets, as shown in Section 4.3.4.

4.3.3 Saturation Throughput and Latency

Packet chaining increases saturation throughput and reduces latency compared to iSLIP-1 because packets spend less time blocked at routers. As shown in Figure 4.7(a), for uniform random traffic, considering all VCs of the same input or all inputs and VCs provides a 5% increase in saturation throughput. By average across traffic patterns, considering all VCs of the same input increases saturation throughput by 6% whereas considering all inputs and VCs by 4%. Packet chaining also provides a 4.5% lower latency by average until saturation. That percentage becomes 16% if statistics for injection rates lower than 20% are excluded. Latency reduction is due to more efficient matchings making flits more likely to advance if their desired output is free. This is similar to the reduction of latency when going from a single to multiple iterations on an iSLIP allocator [24].



(a) 2D mesh.

(b) Flattened butterfly with UGAL.

Figure 4.7: Injection rate-throughput with single-flit packets.

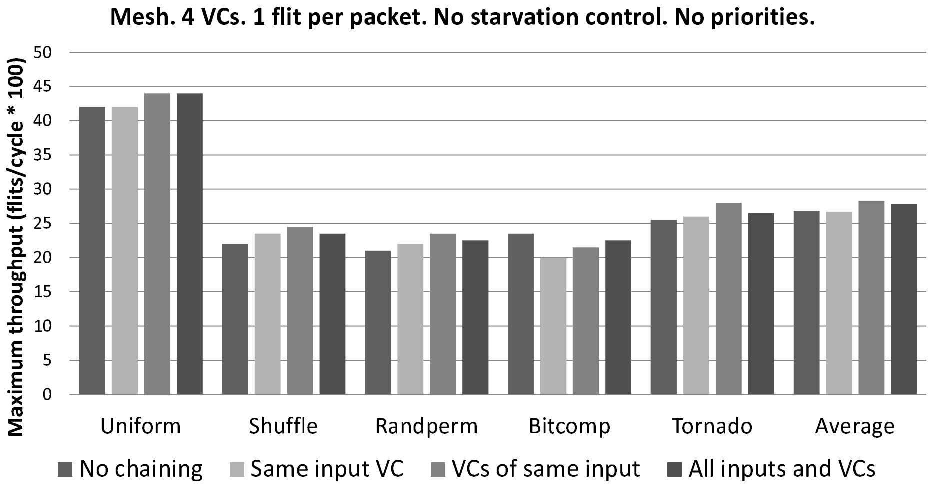


Figure 4.8: Allocator comparison by traffic pattern.

To gain further insight, we extract the number of cycles that eligible head flits wait for the connection to their desired output to be released and for a switch allocator grant to be received. This is measured in the mesh at the saturation injection rate for each case; connections are released after eight cycles to prevent starvation and results are compared to iSLIP-1. By average, packet chaining reduces this blocking latency by 13% for single-flit packets, 21.5% for two-flit packets and 7.5% for four- or eight-flit packets. This highlights the increased matching efficiency from packet chaining since packets wait fewer cycles for a switch grant, and therefore more packets traverse the switch in a given time period.

Results for the FBFly are shown in Figure4.7(b). Selecting among all inputs and VCs increases throughput by 9% for uniform random traffic and 4% by average across traffic patterns, compared to disabling packet chaining. The other selection schemes result in comparable throughput gains.

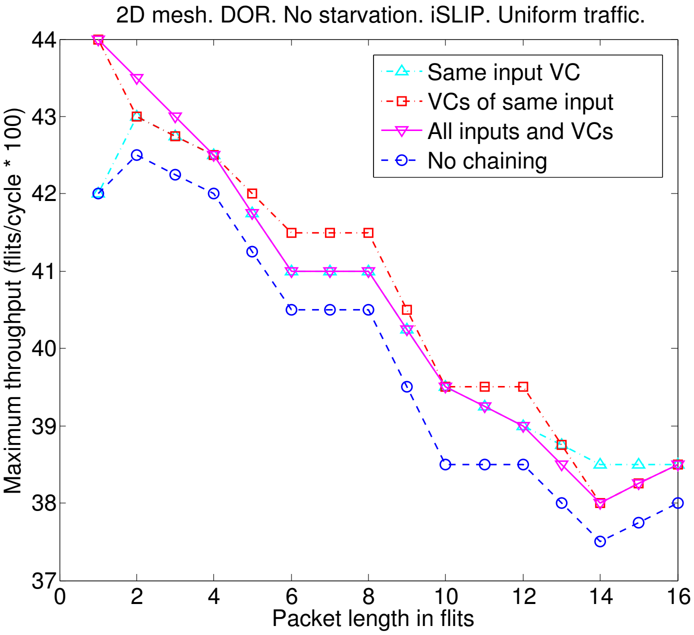
Figure 4.8 shows that the advantages of packet chaining remain largely the same across traffic patterns except for *bitcomp* (bit-complement) without starvation control because bitcomp creates continuous flows of traffic which starve other packets. By releasing connections after four cycles with bitcomp, packet chaining is comparable (offers 2% higher throughput) to iSLIP-1 without packet chaining. Results are similar for the FBFly, where packet chaining consistently offers higher throughput, with the exception of *transpose* for the same reason as bitcomp in the mesh. These traffic patterns illustrate the lack of global fairness in our simulations. For *shuffle, tornado* and *neighbor*, selecting among VCs of the same input provides a higher throughput than all inputs and VCs.

4.3.4 Packet Length

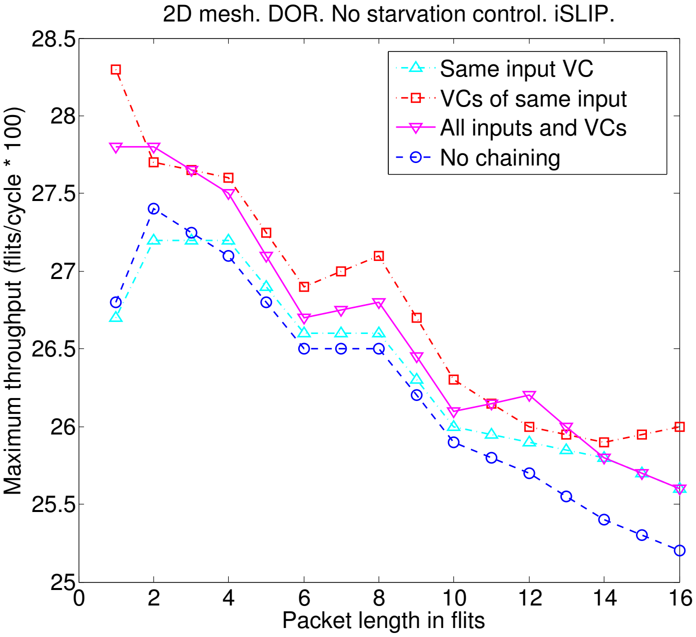
Packet chaining always provides performance benefits, but the benefits compared to incremental allocation decrease when increasing packet length because incremental allocation creates connections and thus improves switch allocation without chaining packets. Furthermore, longer packets translate into fewer packet boundaries and thus fewer packet chaining opportunities. This reduces the performance gains of packet chaining compared to wavefront and augmenting path. Also, with long packets the PC allocator has a lower activity factor. The effect of packet length in throughput is shown in Figure 4.9.

By average across traffic patterns and compared to iSLIP-1 (no chaining), throughput is comparable (2% gain for packet chaining) for eight-flit or longer packets. The marginal benefit for packet chaining shows that even though there are fewer opportunities to chain packets, doing so still provides benefits. With starvation control enabled, throughput for uniform random traffic with sixteen-flit packets is no lower than that of iSLIP-1. Disabling starvation control increases throughput slightly (1.5%) for some traffic patterns. In those traffic patterns, starvation control releases connections to prevent starvation, but starvation would not form or would not become an issue (because it would resolve quickly) otherwise. The FBFly displays similar behavior as the mesh.

Throughput drops for all test cases with the increase of packet length due to the constant buffer size. Packet chaining enables long packets to be subdivided to avoid this reduction in performance, without loss of allocation efficiency due to more short packets. The only exception is increasing from one-flit to two-flit packets with iSLIP-1, which clearly illustrates the gains when incremental allocation is able to form connections.



(a) Uniform random traffic.



(b) Average across traffic patterns.

Figure 4.9: Throughput by packet length in flits for the mesh.

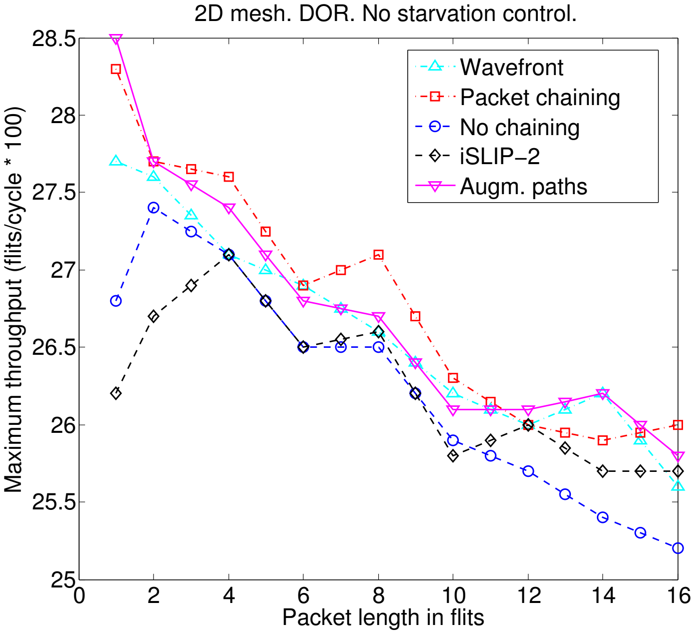


Figure 4.10: Throughput by packet length for the different allocators averaged across traffic patterns.

Figure 4.10 compares packet chaining to more complex allocators. For eight-flit packets, packet chaining is comparable to (outperforms by 2%) wavefront and iSLIP-2, as well as augmenting path (outperforms by 1.5%) by average across traffic patterns. For uniform random traffic, packet chaining is comparable to augmenting path, wavefront (outperforms by 2.5%) and iSLIP-2 (outperforms by 1%). Therefore, packet chaining provides comparable (and slightly increased) throughput to slower and more expensive allocators with long packets. The average throughput of iSLIP-2 with short packets is lower than that of iSLIP-1 because of bitcomp, where locally optimal decisions made possible by the second iteration are not globally optimal.

Traffic patterns that comprise equal amounts of short and long packets (bimodal) still benefit significantly from packet chaining, which increases overall throughput. For instance, when assuming a request-reply protocol with single-flit short and five-flit long packets, packet chaining provides a marginal (1%) throughput increase by average across traffic patterns and a 4% increase for uniform random traffic, when considering all inputs and VCs. These gains are compared to 2.5% for solely five-flit packets and 5% for solely single-flit packets under uniform random traffic. Performance under realistic bimodal patterns is shown in Section 4.3.8.

4.3.5 Optimal Packet Chaining Scheme

The optimal chaining scheme depends on the network configuration which affects what outputs packets are more likely to request at each hop. Selecting among all VCs of the same input is optimal for the mesh with DOR because it avoids requests from different inputs (outputs) to the same output (input), but still provides the opportunity to chain the majority of departing tail flits because with DOR flits are more likely to remain in the same dimension in each hop. Also, because one input can only be chained with only one specific output, selecting among all VCs of the same input does not require a complete PC allocator. Increasing the density of the request matrix by selecting among all inputs and VCs can decrease matching efficiency of our separable PC allocator, because requests from any input to any output can cause suboptimal decisions at the input and output arbiters, as explained in Section 4.1.1.

However, more complex routing algorithms are less predictable, which may necessitate considering all inputs and VCs. In the FBFly with UGAL, flits are less likely to request the same output as the one chained to their input, compared to the mesh with DOR. In our simulations, considering all inputs and VCs provides gains comparable to considering all VCs of the same input, because the latter still has an adequate number of chaining candidates due to the presence of four VCs per input in our network. With fewer input VCs, considering all inputs and VCs would provide higher performance. The predictability of the outputs requested by packets also depends on the traffic pattern, which may not be known at design time. Furthermore, simpler chaining schemes intensify fairness issues because more input VCs cannot be served before the starvation control mechanism releases the conflicting connection. Selecting among only the same input VC is too restrictive for packet chaining to be effective.

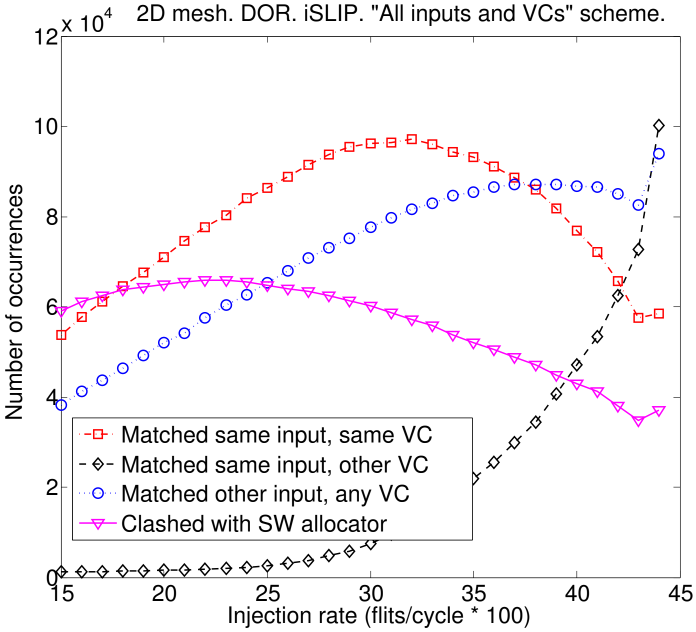
Therefore, given enough input VCs or a routing algorithm that makes the outputs packets request at every hop predictable, considering all VCs of the same input will most likely provide comparable performance to considering all VCs and inputs, but without the overhead for a complete PC allocator.

4.3.6 Packet Chaining Probability

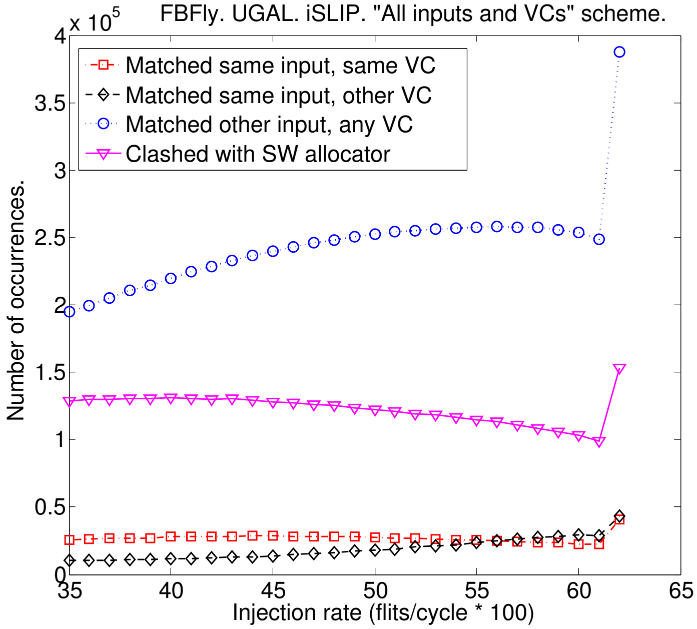
Figure 4.11 shows the input VCs that are connected with departing tail flits, when considering all inputs and VCs. Multiple chaining requests to the same output are caused by the same departing tail flit and thus regarded as one chaining opportunity. Also, multiple requests from the same input (but different input VCs) to different outputs are regarded as one, because only one can be granted. Failure to chain includes all requests which did not result in a chained connection, except for connections released because all output VCs became reserved or full during the PC stage, in parallel with the PC allocator. Connections released due to starvation timeout are also not counted in this figure. Connections released because of conflicting switch allocator decisions (grants for the same input or output) are also illustrated. However, failed chaining attempts are not shown.

At low loads, there is a significant number of clashes with the switch allocator because there are only a few chaining candidates and the switch allocator is able to provide efficient matchings. Thus, it is more likely to grant the same inputs and outputs as the PC allocator. The number of clashes initially increases and then decreases when the switch allocator's efficiency decreases due to the heavier load. At low loads, a significant percentage of the few packet chaining requests are successful, because there are only a few chaining requests. The number of successful chaining attempts increases with injection rate and remains constant after saturation (0.45 flits/cycle for the mesh and 0.65 flits/cycle for the FBFly).

For the mesh, above 0.32 flits/cycle the number of chains to the same input and VC decreases, and the number of chains to another VC of the same input increases. This is because our network assigns VCs in each traffic class in order starting from the lowest-numbered VC. Thus, at low loads almost all packets are in VC 0. The probability of chaining to another input remains smaller than chaining to the same input because with DOR the input that is already part of the connection is more



(a) 2D mesh.



(b) Flattened butterfly with UGAL.

Figure 4.11: An illustration of grants by the PC allocator.

likely to contain flits routed to the connected output. At saturation, 9% of requests chain to another VC of the same input, 5% chain to the same input and VC, 8% chain to another input, and 74% of the chaining requests fail. The low probability of chaining to the same input and VC illustrates the few chaining opportunities from the same input and VC, which is the reason for packet chaining's poor performance gains when considering only the same input VC. Results differ for routers at the edges and corners of the mesh because they have fewer inputs and outputs and are also essentially operating at a lighter load due to DOR and the asymmetry of the 2D mesh. Those routers exhibit statistics similar to the lighter loads shown in Figure 4.11.

In the FBFly, UGAL routes packets minimally using DOR with one hop per dimension to their intermediate and final destinations [100]. Thus, packets are routed less predictably than in the mesh. However, finding consecutive packets wanting to make the same turn is still likely, as shown by the probability to chain using the same input. Furthermore, due to the two traffic classes required by UGAL, traffic is spread over VCs more than in the mesh, and therefore it is less likely to chain with the same input VC. At saturation, 14.5% of the packets chain with a packet from another input, 2% chain with a packet from the same input and VC, and 2% chain using the same input but another VC. Because the FBFly is symmetric, chaining probabilities are comparable among all FBFly routers.

4.3.7 Starvation and Priorities

Section 4.1.5 describes two starvation control mechanisms. In this section, we evaluate the mechanism which releases connections after a predetermined number of cycles. This provides weaker fairness guarantees but also avoids increasing the number of priority classes the PC allocator needs to support. In our simulations, this mechanism was adequate to prevent starvation, and therefore there was no need for the more complex mechanism.

Starvation control has a minimal effect on throughput and latency. For single-flit packets, a starvation threshold of eight cycles provides a marginal (1.5%) throughput increase due to improving fairness, while for eight-flit packets it has no effect.

However, setting a starvation threshold smaller than the packet length reduces performance gains because starvation control releases connections before packets can be fully transferred. Therefore, packets wait for a switch allocator grant while having reserved an output VC. For instance, using a starvation threshold of four cycles with eight-flit packets drops maximum throughput by an average of 3%, compared to not using starvation control. This illustrates that starvation control can negate packet chaining gains if it releases connections too early. Similarly, simulations with sixteen-flit packets have a comparable (1.5% lower) maximum throughput with a starvation threshold of eight cycles.

As discussed in Section 4.3.3, starvation control increases throughput in traffic patterns in which packet chaining would cause starvation. In the cases where performance drops with starvation control, packet chaining never performs worse than iSLIP-1 (no packet chaining). Starvation control has a more significant effect with packet chaining schemes with very few chaining candidates, such as considering only the same input and VC. That is because more inputs and VCs risk being starved because they are not considered for chaining. When considering all inputs and VCs, the iSLIP-1 PC allocator already performs round-robin selection of inputs. Selecting a PC allocator with some inherent fairness properties assists with providing adequate fairness and starvation control.

Latency distributions are similar for networks with and without starvation control. Throughput results presented in this paper are the minimum throughput among all sources for each simulation (worst-case throughput). Therefore, worst-case throughput is also similar for networks with and without starvation control. This shows that in all our simulations, connections were released before noticeable starvation or unfairness arose. Under low loads, connections were usually released due to input VCs becoming empty. Under high loads, connections were usually released due to output VCs without credits. Therefore, the starvation mechanisms we proposed should have a threshold which does not degrade performance in the common case, but also adequately prevents fairness issues and starvation under adversarial traffic.

Disabling priority-handling in the PC allocator reduces throughput by 6.5% for uniform random traffic and 4.5% by average across traffic patterns and with single-flit

Table 4.1: Packet chaining versus iSLIP-1 using application benchmarks.

Benchmark	IPC increase	Benchmark	IPC increase
Blackscholes	46%	Canneal	1%
Dedup	6%	FFT	9%
Fluidanimate	3%	Swaptions	29%
Average	16%		

packets. That is because PC allocator requests which are probable to be cancelled due to unfavorable switch allocator decisions may no longer be placed in the lower priority class, as explained in Section 4.1.4.

4.3.8 Application Performance

Table 4.1 presents our application results. In our simulation infrastructure, results for IPC correspond to those for execution speedup. Packet chaining increases IPC, but the gains depend on the load and traffic pattern created by each application. Applications with an increased network load, bursty traffic or shorter packets receive higher benefits from packet chaining. Applications with working sets larger than L1 caches create a high load on the network. Under high load, the network may operate past saturation and thus benefit from reduced tree saturation due to packet chaining. For instance, Blackscholes has the largest IPC reduction because it creates more network traffic both in small periods of time (bursty traffic) and by average. The same is true for Swaptions, but to a smaller degree. Blackscholes and Swaptions both create fairly uniform traffic and have little synchronization, so they are more capable of loading the network. Other applications either create less traffic or are more latency-insensitive. For example, Canneal and Dedup create hotspots at the memory controllers and the network is not loaded because processing cores wait for those memory accesses. Increased load and short packets may be caused by the interaction of the application with the cache system. This interaction determines the number of distant read or write requests, as well as the number and type of control messages, such as for synchronization or data invalidation. This is particularly true for systems with small cache lines and more communication-heavy coherence protocols.

Short packets are critical in a typical cache-coherent CMP. They can affect execution time significantly because they transfer time-critical control messages that are often on the application's critical path. Processing cores (network endpoints) often wait for control messages in order to make progress or generate further traffic which other processing cores are waiting for. Of note, 53% of the packets are single-flit by average across applications in our simulations. This shows the significance of optimizing single-flit packet traffic in CMPs. In addition to increased throughput, a crucial factor for the IPC increase is reducing packet latency due to packet chaining, especially at times of heavy network load. Maximum packet latency is reduced by an average of 20% with packet chaining. Average latency is only 7% less with packet chaining, because many packets are sent under low network load, and thus have low latency with and without packet chaining. Reducing maximum latency is important under high network load.

Most applications are not affected by the network for most of their execution time, because many parallel algorithms consist of computation phases with no barriers and working sets that fit into L1 caches. In these cases, gains from network optimizations are limited. Finally, applications may or may not benefit from starvation control, depending on their traffic pattern. For instance, performance for Canneal and FFT degrades without starvation control.

In addition, packet chaining is comparable (provides a 0.5% lower IPC) by average across applications compared to wavefront, which has higher timing and cost overheads as explained in Section 4.3.9. This clearly illustrates that packet chaining offers performance comparable to more complex allocators without the associated delay and cost.

Previous work has observed that many applications in certain CMP configurations make light use of the network and thus are not affected by techniques improving throughput, like packet chaining [98]. In these cases, networks can reduce their cost, for example by narrowing their datapath, such that their average load increases and thus they become throughput-limited. Packet chaining makes this option more attractive by increasing maximum throughput and reducing latency past very low injection rates. To illustrate this point, packet chaining increases IPC by an average of 16% compared to iSLIP-1 when both networks have a datapath width of 32 bits, which is half compared to the results presented above. While the average IPC increase across applications remains the same as with a 64-bit datapath, the maximum IPC increase is reduced to 37% for a 32-bit datapath and occurs for Swaptions. These results also show that packet chaining does not increase application performance only if the shortest packets are single-flit because with a 32-bit datapath the minimum packet length is two flits.

4.3.9 Packet Chaining Cost

Packet chaining, when considering all inputs and VCs, requires an extra PC allocator similar to the switch allocator. The PC allocator is placed in parallel to the switch allocator as illustrated in Figure 4.12. Flits in the PC and SA stages described in Section 4.1.4 are physically located in the input buffers. Advancing to the SA stage from the PC stage is accomplished by updating the active connections instead of transporting flits. Flits depart the buffers when ready to traverse the switch. Similar to the combined switch allocator [63], requests for the PC allocator are OR-reduced to a P×P set of requests. Each input and output port maintains a register to store which other input or output port it is connected to.

Allocators only consider eligible requests as described in Section 4.1.4. Moreover, the PC allocator must have knowledge of tail flits in the SA stage. That check is part of eligibility checking and it simply requires an extra input to the AND gate responsible for deasserting ineligible PC allocator requests. All data for eligibility checking resides in the input buffers and state registers, and therefore is available at the beginning of the cycle. The eligibility checking logic for the PC and switch allocators are practically identical (and in parallel) because both allocators consider non-empty input VCs and output VCs with remaining credits, with the exception that the PC allocator considers connected inputs and outputs, whereas the switch

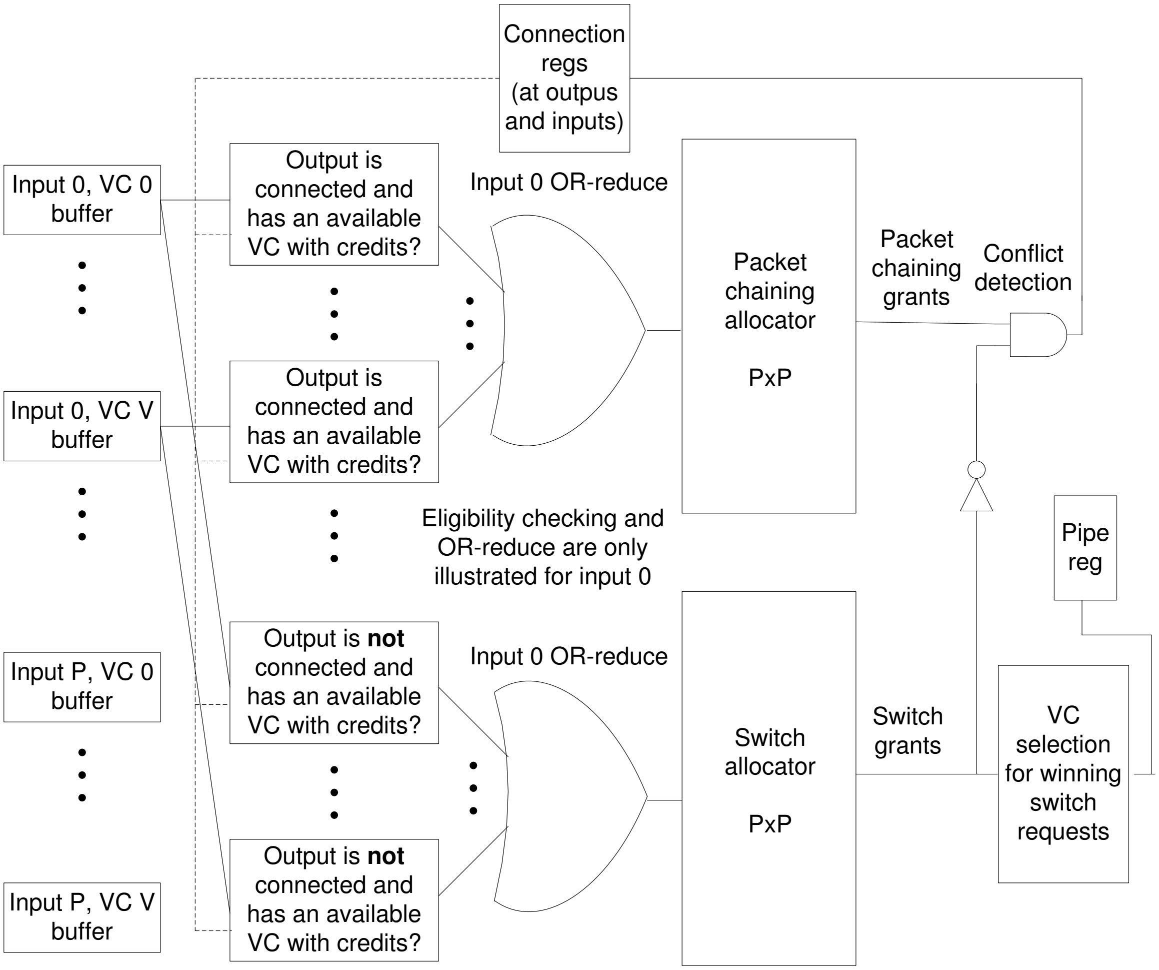

Connection regs (at outpus and inputs)
Input 0, VC 0 buffer
Input 0, VC V buffer
Input P, VC 0 buffer
Input P, VC V buffer
Output is connected and has an available VC with credits?
Output is connected and has an available VC with credits?
Output is **not** connected and has an available VC with credits?
Output is **not** connected and has an available VC with credits?
Input 0 OR-reduce
Input 0 OR-reduce
Eligibility checking and OR-reduce are only illustrated for input 0
Packet chaining allocator
PxP
Switch allocator
PxP
Packet chaining grants
Conflict detection
Switch grants
Pipe reg
VC selection for winning switch requests

Figure 4.12: Block diagram of the PC and SA stages with the PC and switch allocators in parallel.

allocator considers unconnected inputs and outputs. Therefore, adding the eligibility checking logic for PC allocation does not prolong the allocation timing path.

At the end of the allocation pipeline stage, AND gates perform conflict detection by deasserting any PC allocator grants that conflict with switch allocator grants. Then, the state registers that keep a record of the active connections, are updated, which is part of the allocation timing path with incremental allocation as well. The results of the PC allocator affect switch allocator request eligibility in the next cycle by setting the connection registers. Note that the logic at the end of the pipeline stage requires a few more logic gates if lower-priority requests to the PC allocator are generated for chaining requests which depend on switch allocator grants, as explained in Section 4.1.4. However, this check can be performed by a single logic gate at each output which takes as input the switch allocator grant for that output and the desired switch allocator result to make the PC allocator grant for that output valid. This desired result can be computed early in the cycle.

The conflict detection and lower-priority PC request handling operate in parallel with assigning VCs to winning switch requests which is more complex and also occurs at the end of the pipeline stage for the combined switch allocator [63]. Therefore, the necessary logic after PC allocation does not prolong the allocation timing path. If the switch allocator is not combined but there is a separate VC allocator, the one or two gates per PC allocator output described above prolong the allocation timing path only marginally compared to the rest of the timing path. If speculative VC-switch allocation is used, the logic after the switch allocator to handle speculative requests is similarly complex as, and in parallel with, PC conflict detection.

The cost and timing overhead of packet chaining described above should be compared to wavefront because wavefront provides performance comparable to or lower than packet chaining. In a mesh, wavefront requires up to 3× the power, 2.5× the area and 20% more delay than separable allocators [10]. In high-radix routers such as the FBFly, wavefront occupies 2.7× the area, consumes 6× the power and has an increased delay by 36% [10]. In contrast, adding the PC allocator doubles the area for allocation. Power doubles in the worst case, which we assume for our calculations, but in the average case the switch allocator's activity factor will be reduced, reducing its dynamic power. Furthermore, as explained above, PC allocation does not prolong the allocation timing path with a combined separable switch allocator. Therefore, compared to packet chaining, wavefront requires 1.5× more power, 1.25× more area and 20% more delay in the mesh, as well as 3× more power, 1.35× more area and 36% more delay in the FBFly. Also, compared to packet chaining, a two-iteration separable switch allocator has the same area but twice the delay and worst-case power because it performs two iterations in a single cycle. Finally, augmenting path allocators are even more complex than wavefront and thus are too costly for single-cycle implementations [46].

Considering only VCs from the same input significantly simplifies packet chaining because an arbiter per input is required instead of a complete allocator. As shown by our results, this scheme still offers comparable or superior performance to wavefront and augmenting path in numerous cases with only a small fraction of the cost for the PC allocator and no delay overhead. Therefore, considering only VCs of the same input further increases the cost gains in favor of packet chaining in the cases where considering all inputs and VCs does not provide any additional performance.

4.4 Summary

Packet chaining is a simple and effective method for increasing allocator matching efficiency without extending allocation time, focusing on short packets. It extends the benefits of incremental allocation to packets of any length. Compared to iSLIP-1 with incremental allocation, which has comparable allocation delay, packet chaining offers a 15% increased throughput at maximum injection rate. Packet chaining increases throughput compared to multi-iteration iSLIP allocators and wavefront allocators by 10% and 6% respectively under maximum injection rate, and gives comparable (1% higher) throughput to an augmenting path allocator for single-flit packets. For long packets, packet chaining still offers comparable or slightly increased throughput compared to these allocators. Packet chaining achieves this without the delay or cost of these more complex allocators, especially in high-radix routers where the overhead of these allocators increases and can reach up to 6× more power and 37% more delay for a wavefront allocator compared to a separable allocator in a FBFly [10]. Cache-coherent CMPs benefit from packet chaining because short messages are critical and often dominate traffic. In our simulations using application benchmarks, packet chaining increases IPC by up to 46% (16% average).

Packet chaining is beneficial to a wide range of systems and provides a simple way to increase allocation efficiency with minimal impact on the allocation timing path and without the area and power overheads of more complex allocators. It provides an elegant solution to the pressing tradeoff between allocation quality, delay and cost present in on-chip networks. By increasing allocation quality, the network can improve performance for both throughput- and latency-limited CMPs, or can simply match the maximum performance required by a specific system but at a lower cost. Finally, our work shows the importance of optimizing the network for short packets, because short packets are critical for CMP traffic.

Chapter 5

Related Work

Due to the pressing need to reduce communication energy, there has been a considerable amount of research on bufferless flow control. Deflection flow control was first proposed as “hot-potato" routing in off-chip networks [9]. Deterministic routing algorithms have been further investigated with deflection flow control in various topologies to provide near-optimal performance by assigning deflection priorities to packets depending on their location relative to their destination's row and column, and if they are travelling on their destination's row or column [14]. Recently, deflection flow control has been evaluated in the context of modern on-chip networks. That work concluded that the most important factor affecting performance is the network topology, and that global or history-related deflection criteria are beneficial [71]. Dynamic routing has also been developed which prioritizes packets based on factors such as the number of deflections they have suffered [16]; this way, the network can guarantee an upper bound for delivery time, which has also been proposed in [15]. Further research has proposed using load information from neighboring routers to make switching decisions at every hop, thus avoiding transmitting or deflecting packets to congested regions [90]. Furthermore, analytical models have been proposed to estimate throughput and latency in mesh networks with deflection flow control [38].

Dropping flow control has also been studied. In the context of on-chip networks, dropping flow control has not been researched intesively, because it requires large buffers at traffic sources to buffer packets until they are successfully received; since buffering resources are expensive in on-chip networks, this is a major drawback. Past research has focused on reducing the dropping probability to mitigate some of dropping flow control's energy drawbacks by using loopback channels or deflecting packets instead of dropping them [35]. Further research has also combined deflection and dropping flow control and chooses between the two options for packets under congestion depending on the availability of output ports for deflection [34]. The same work also investigated the effect of duplicating channels between routers in order to provide more outputs and thus reduce dropping probability. Dropping flow control is more appropriate in off-chip networks, where buffering resources and complexity at traffic sources are less significant. Dropping packets is used by TCP to detect congestion, and adjust the transmission window appropriately to avoid congestion (and thus more dropped packets) in the near future. Dropping flow control, and specifically TCP, is predominantly used in datacenter networks because of the long round-trip between routers, which inserts a lag in detecting congestion in nearby routers [2, 3].

Recent work has evaluated and tuned bufferless networks for chip multiprocessors (CMPs). BLESS implements deflection flow control with deterministic dimension-order routing (DOR) in a mesh [83]. BLESS requires that routers have at least as many outputs as inputs. Injecting flits to a router requires a free output port to avoid deflecting flits to ejection ports. To avoid livelock, older flits are given priority. BLESS also evaluates two deflection policies: FLIT-BLESS and WORM-BLESS. In FLIT-BLESS, every flit of a packet can be routed independently. Thus, all flits need to contain routing information, but not extra information found only at the header flit, such as packet type and error detection checksums. This imposes overhead compared to buffered networks, where only head flits contain routing information. To reduce this overhead, WORM-BLESS tries to avoid splitting worms by providing subsequent flits in a packet with higher priority for allocating the same output as the previous flit. However, packets may still have to be split under congestion, and WORM-BLESS still needs to be able to route all flits independently. A third variant is also proposed but not evaluated in [83] which combines deflection and buffered flow control by providing input buffers but no backpressure. With this scheme, flits get buffered but flits arriving to a full buffer cause the flit at the head of the buffer to be deflected. As explained in Chapter 2.1, our bufferless evaluation uses FLIT-BLESS because FLIT-BLESS performs better than WORM-BLESS [83], but incurs extra overhead which, however, is not modelled in our evaluations.

More recent work used the observations on deflection flow control in this thesis to mitigate some of deflection flow control's shortcomings [30]. That work proposed replacing the router pipeline with a partial permutation network composed of 2×2 switches. This way, the long allocation timing path described in Section 2.3.1 is broken, but the deflection probability is also increased and routers become inflexible for the number of inputs and outputs they can support. Other work has also proposed an allocator for deflection flow control without changing the router architecture, but the long timing paths remain [28]. Furthermore, this work proposes a scheme to avoid livelocks by promoting one packet at a time to the highest-priority level for enough time to guarantee that this packet will be delivered before its priority expires. While this scheme simplifies allocation, it further increases the maximum, median and average packet latencies because every other packet is of the same priority and thus can get deflected, and the number of cycles before any specific packet receives highest priority is large. Finally, that work proposes a buffer reservation protocol where packets are speculatively sent to a destination, but upon encountering a full ejection buffer, they are dropped and not retransmitted until buffer space at the destination's ejection buffers has been reserved. This scheme prevents deadlocks, but increases latency and energy if packets have to be retransmitted, may underutilize ejection buffers due to potentially long round-trip delays of positive or negative acknowledgments, and assumes a CMP system because in CMPs miss status handling registers (MSHRs) can be used as retransmission buffers instead of explicitly adding buffers.

Dropping flow control has also been proposed for CMP on-chip networks [44]. That work uses a separate circuit-switched network to deliver negative acknowledgments (NACKS) in order to speed up retransmission. Furthermore, that work designed a minimally-adaptive switch allocator and also proposed buffering flits in the MSHRs of intermediate nodes instead of dropping them. This technique reduces retransmission overhead because future retransmissions of flits buffered this way originate from the intermediate nodes.

Our bufferless network evaluation of Chapter 2 extends past work by performing a thorough evaluation of deflection flow control and providing insight on the various complications bufferless flow control faces. These complications were not previously well-known, and are now the basis of research on bufferless networks [30]. One of our primary contributions is also our comparison with a highly-tuned buffered virtual channel (VC) network, which shows that buffered networks are superior except in very low loads, where they are marginally more costly in power. However, our analysis shows that the marginal energy gain is outweighed by the numerous complications of bufferless flow control. Finally, to perform an equitable comparison, we propose multi-dimensional routing (MDR), a routing algorithm to reduce deflection or dropping probability. MDR is similar to the adaptive routing for bufferless networks described in [90], but that work uses congestion sensing to prioritize among productive outputs and therefore is more complex. Furthermore, a similar routing algorithm is proposed in [34] but with a different allocation scheme which therefore chooses among productive outputs in a different manner.

Past research has proposed other bufferless flow control schemes, such as compressionless routing which relies on feedback from the network sent back to the network interfaces [58]. Bus-based networks have had limited application to on-chip networks due to the large scale of such networks. However, past work has proposed segmenting buses to mitigate the scalability issues and reduce the router traversal overhead [73].

Past work has also attempted to mitigate buffer costs by proposing novel implementations. Leakage-aware buffers [88] reduce leakage by directing incoming flits to the least leaky buffer slots and supply-gating unused slots. Also, to increase buffer utilization and reduce buffer size, researchers have proposed dynamic buffer allocation which makes more efficient use of buffer space, thus enabling a reduction in buffer size while maintaining performance constant [87].

The most widely-known bufferless flow control in off-chip networks is circuit-switching [24]. Circuit-switching establishes end-to-end connections (circuits) prior to packet transmission in order to prevent contention in the bufferless network. In the context of on-chip networks, the latency that short control packets face to establish a circuit is significant. That latency can be reduced in the absence of contention by speculatively transmitting packets without an established circuit. If speculative packets contend, they get dropped and become circuit requests [24]. Time-division multiplexing can also be used to share channel bandwidth between multiple circuits [70]. Past work proposes to use a reconfigurable network combined with circuit-switching to separate concurrent data flows [110]. Further work combines packet-switching with circuit-switching by requiring routers to provide buffer space only for short control packets (such as read request packets), while longer packets have to establish end-to-end circuits [109].

The concept of circuits has lead to the development of virtual circuits which can be combined with packet-switched networks to reserve resources and reduce packet overhead for dominant flows or flows meeting other criteria, such as distance to destination [26, 104]. In some schemes, intermediate nodes can quickly tear down and modify end-to-end virtual circuits [105]. The same end can be achieved by express virtual channels [65], low-latency pre-configured paths [80] and skip-links [51] which also establish dynamically-reconfigurable paths to bypass intermediate routers for flows matching certain criteria.

While Section 3.1 proposes an implementation for the elastic buffer (EB) control logic and the ready-valid handshake [75], there have been numerous other proposals in different and similar contexts [21, 52]. Those proposals focus on elastic pipelines in processors, but can be easily used in EB networks as well. In [43], *valid* is not required as part of the handshake because invalid data is never present in the pipeline. That work also discusses how to make EBs scannable. In [52], the ready-valid handshake and EB control logic are essentially the same with some optimizations from custom circuit design. That work also discusses how to connect one EB to multiple EBs [52]. Past work has also examined different backpressure and handshake techniques applicable to EB networks [97]. Alternative and slightly more costly EB implementations add an auxiliary flip-flop (FF) to every pipeline FF [17], therefore doubling the amount of latches in the channels. The auxiliary FF is used to store incoming data when the pipeline is stalled. Finally, another EB design uses repeaters to store data by adding transistors to disconnect the voltage supply and ground when the pipeline is stalled [82]. However, this design requires very careful implementation and faces charge sharing and leakage current issues because data is stored in parasitic capacitance. Due to the inevitable leakage current, data cannot be stored indefinitely without being corrupted.

Past on-chip networks have used the EB design with the main and auxiliary FFs [20] as well as the EBs based on repeaters [61]. However, both of these proposals make full use of router input buffers in all cases, and therefore retain the associated complexity and costs. On the contrary, EB flow control as described in Chapter 3 uses EBs instead of input buffers, and significantly simplifies router design by removing credits and VCs. If a large number of traffic classes are required, input buffers and VCs are reintroduced but the buffers are still not used in the common case; thus the majority of energy savings are retained.

While universal globally adaptive load-balancing (UGAL) was used in the EB flattened butterfly (FBFly) network of Section 3.5.2 as an example of applying adaptive routing to EB networks with duplicate physical channels, other adaptive routing algorithms are equally applicable. Furthermore, past research on congestion-sensing and adaptive routing is applicable to EB networks. EB networks can use regional congestion awareness [36]. They can also propagate control information in advance of the packets to create reservations at the output ports [94] and pre-compute arbiter decisions [85]. Finally, other proposals such as crossbar decomposition [60] or slicing [24] are applicable. Applying these and other orthogonal optimizations to EB networks is beyond the scope of this study.

Research related to packet chaining has also been conducted. Pseudo-circuits [1] operate on the same principle as packet chaining but only consider consecutive packets in the same input VC. Also, flits in pseudo-circuits skip router pipeline stages. Pseudo-circuits are released when another input VC requests the connected output to prevent starvation. Therefore, the benefits from pseudo-circuits come from enabling flits to skip pipeline stages, only in the absence of contention. With packet chaining, flits using a connection do not skip the switch allocation stage because in our latency-optimized two-cycle router, doing so would place look-ahead routing [32] in the critical path. It would also require a separate VC allocator which would reduce the number of free VCs available for chaining compared to our combined allocator. Therefore, since we base our work on a latency-optimized router pipeline, packet chaining prioritizes throughput by maintaining connections that can be productively used in the presence of other requests in order to improve allocation efficiency (maximize the number of packets that receive a grant every cycle) under load, as discussed in Section 4.1.2. In addition, packet chaining considers packets from any input and VC, thus vastly increasing the eligible packets and therefore the probability for chaining compared to considering only the same input and VC.

There is much research on allocation which is orthogonal to packet chaining. For instance, speculative VC allocation parallelizes VC and switch allocation [85]. Moreover, requests can be propagated in advance of flits travelling in frequently-used paths [93] or decisions can be precomputed [85]. Allocation can also be performed for packet flows instead of single packets in order to reduce the allocation overhead per packet, and to provide more predictable performance [7]. Packet priorities can be set according to various criteria, such as packet age in order to minimize maximum packet latency [68]. Finally, express VCs [65] and token flow control [64] allow flits to bypass the complete router pipeline based on prior knowledge or established paths. Packet chaining does not rely on pre-established paths or any a priori knowledge and is applicable to such techniques.

Chapter 6

Conclusions

This thesis focuses on energy-efficient flow control for on-chip networks. Specifically, we analyze the effect of router buffers in the cost efficiency of the network, as well as how to mitigate that cost. This question has received much focus recently because high power and area costs have been attributed to the buffers [83, 37]. Therefore, past work has proposed removing router buffers and resolving contention by deflecting [9] or dropping [35] packets. In this thesis, we improve and extensively evaluate bufferless flow control in order to conclude whether it sufficiently addresses the problem of mitigating buffer costs. Then, we propose elastic buffer (EB) flow control which uses pipeline flip-flops (FFs) in the channels for buffering instead of adding input buffers. Finally, we propose a simple technique to increase switch allocation quality to match or exceed more complex allocators, but without extending cycle time.

Our bufferless study clearly shows that unless technology process constraints lead to excessively costly buffers, bufferless networks offer marginal gains at best and only under very low loads. This comes at the price of numerous complications which increase the complexity and cost of the overall system, and makes it challenging to provide performance or low latency guarantees. Except for cycle time, area and power, other important factors such as design and verification time also increase with design complexity. To avoid the complications of bufferless flow control our work describes, on-chip networks require buffering.

EB flow control provides buffering in the network without the cost of input buffers.

It is an example of removing overhead which is not inherently necessary in the network, and instead making full use of resources (FFs) which are required anyway, but are not utilized to their full extend. Therefore, it is preferable to bufferless flow control because it avoids the negative side effects of bufferless flow control, but without the cost for buffers. EB networks also highlight the benefits of simple design. Single-stage EB routers are able to be clocked at the same or higher frequency than two-stage virtual channel (VC) networks. EB routers also lack the overhead for VC allocation, switch allocation (since the switch allocator is replaced by an arbiter for each output port), VC stage managements and credits. Except for cycle time, design simplicity also reduces area and power. Design simplicity also provides flexibility to the designer to tradeoff gains in one aspect for higher gains in another, such that the optimal design for a specific system is reached.

Finally, packet chaining is an example of a technique which optimizes an aspect important to chip multiprocessors (CMPs). By increasing the allocation efficiency for short packets, control and request packets have a reduced latency and the network can sustain a higher throughput. Since control and request packets are often time-critical in CMPs, execution time is decreased, which is the metric that end users value. To achieve this, packet chaining identifies and addresses the shortcomings of incremental flow control when dealing with single-flit packets. Packet chaining increases allocation quality without extending cycle time, and thus offers an alternative to the traditional tradeoff between allocation quality and cycle time.

The impact of the work described in this thesis in real-world systems can be important. As an example, lets assume a CMP with 64 cores which can only afford a certain power budget (e.g. 10 Watts) for its on-chip network. By deploying an EB network instead of the typical VC flow control with input buffers, the CMP can receive 21% higher maximum throughput for the same power budget. Alternatively, the CMP can receive 22% higher throughput for the same area budget, or 45% lower network cycle time. These numbers may be affected by different traffic patterns, but the relative trends between EB and VC networks remain constant. Packet chaining provides similar throughput gains as well as performance gains at maximum injection rate, without affecting cycle time and with minimal cost impact. Therefore, applications in that CMPs that load the network heavily may have a significantly lower execution time by applying these techniques. The percentage gains depend on a number of factors, such as the processors in the CMP, the number of threads, the nature of the traffic patterns, and many others.

In contrast, many systems and applications do not load the network heavily, and thus are not constrained by maximum throughput. However, they are forced to operate the network with a datapath wide enough to keep serialization latency low. In those cases, the network operates well before saturation. EB flow control benefits such systems in multiple ways. Firstly, for equal cost the EB network has a wider datapath than the VC network. Thus, serialization latency is reduced. Furthermore, EB networks can operate at an up to 45% higher clock frequency which reduces latency in absolute time. Moreover, EB networks can use the single-stage router which removes a clock cycle of latency per router. Finally, systems which load the network up to a certain load which is known at design time and does not saturate the network can use EB flow control to get the same throughput with a lower cost. Packet chaining also reduces packet latency before saturation by up to 20% in a CMP due to better allocator performance. Therefore, by using the techniques described in this thesis, latency and cost and be improved even for systems which do not stress the network.

When evaluating techniques for on-chip networks, it is crucial that performance per unit cost is evaluated. It is also important for performance and cost to be defined according to the design priorities and expected traffic patterns of the system using the on-chip network. Since any network can increase its performance and cost by widening its datapath, proposals which do not clearly increase performance per cost may result in a less efficient network compared to the baseline network with a wider datapath. Moreover, any cost savings must be viewed in the context of the overall system. Past research has shown that on-chip networks are a small fraction of the overall power and area cost [98]. Therefore, depending on that fraction and other system parameters, increasing clock frequency may outweigh reducing area or energy costs. Clock frequency is an important factor because, depending on the design, the network clock frequency may constrain the clock frequency of the rest of the system.

Even if it does not, the network may operate at a different clock frequency than other system components, which adds synchronization overhead. Therefore, research to simplify on-chip networks has higher potential for impact.

6.1 Future Work

On-chip networks have been researched extensively in recent years. Recently, a significant part of on-chip network research has focused on the context of CMPs [66, 98, 83]. However, past research fails to gain a deep and solid understanding of how the network affects system performance. That is because new ideas and proposals are simulated with benchmarks and the average statistics, such as utilization factors, latency and execution speedup, are extracted. However, a lot of information and potential may be lost when looking at the averages. Applications go through various communication phases. If we optimize the network for the average traffic pattern, we may not actually be optimizing for any communication phase, but instead we may be optimizing for a pattern that never actually happens. Therefore, it is critical to identify the various communication phases, determine what is the primary constraint for their performance, and optimize the network to satisfy each communication phase as much as possible, taking into account each phase's importance.

Moreover, while performance per unit cost efficiency may be increased with various techniques such as the ones presented in this thesis, the cost budget for an on-chip network in a system, such as a CMP, can be reduced by making high-load operation more attractive. Current CMPs often under-utilize the network [83, 98]. This has led researchers to prioritize latency instead of throughput. For this, it is important to remember that there are numerous throughput-limited systems [49], including graphical processor units (GPUs) with stream multiprocessors. However, it is more important to remember that in any system, operating under low load is not an inherent property of on-chip networks, but a design choice. Any network can operate at high-loads in any given system by narrowing the datapath accordingly. However, operating under high loads increases average latency, but also latency variance. High loads also make it challenging to provide quality of service (QoS) guarantees. Therefore, if on-chip networks provide more predictable performance, designers can reduce the cost of their network by using a cheaper network, and affect system performance only marginally because the system can be designed to tolerate the predicted latency and throughput. For instance, if the network can provide maximum latency guarantees (even if 95% of the time), the processing cores can have just enough latency tolerance such that they are not adversely affected. Therefore, making high-load operation more attractive in on-chip networks by providing throughput and latency QoS guarantees for all traffic sources may enable the use of narrow and cheap on-chip networks.

Finally, there is a potential to co-design on-chip networks with the rest of the CMP components. This may reduce overhead that neither the CMP nor the network can reduce in isolation. For instance, when a processing core performs an addition, it requires only the results and none of the operands. Therefore, if the network can perform the addition in a router that is closer to the location of the two operands, it can generate a reply packet with only the result. The addition could be performed at a router in the network that minimizes packet hop count. This way, the activity factor is reduced because fewer bits travel and in shorter distances to transfer the result to the requesting core. Reducing data propagation reduces energy costs, but also reduces contention inside the network. This concept might also apply in coherence protocols where the network can assist in locating data and relieve directories and caches from having to know what node to query. By optimizing the network and CMP together and not independently, a more optimal global design point may be achieved.

On-chip networks serve a variety of roles. In throughput-limited systems they must be engineered for maximum throughput, whereas for latency-limited systems latency is critical. Therefore, the goal in each case is to approach as much as possible the *ideal* on-chip network, in the context that is relevant in each system. To achieve this, designers can benefit from gaining a broad understanding of the system, and striving to increase performance per unit cost. However, design complexity and cycle time may also be crucial and often are at ends with increasing performance per unit cost.

In conclusion, future research on on-chip networks may benefit from viewing the overall system perspective and taking advantage of the network to optimize system-wide performance and cost measures. For example, functionality could be moved to the network in order to reduce unnecessary overhead such as data lookup as well as data movement by optimizing data placement and operating on data inside the network in order to minimize propagation. This is an example of breaking down the design boundaries between the network and the rest of the system.

Bibliography

[1] Minseon Ahn and Eun Jung Kim. Pseudo-circuit: Accelerating communication for on-chip interconnection networks. In *Proceedings of the 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, 2010.

[2] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture. In *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, SIGCOMM, 2008.

[3] Mohammad Al-Fares, Sivasankar Radhakrishnan, Barath Raghavan, Nelson Huang, and Amin Vahdat. Hedera: dynamic flow scheduling for data center networks. In *Proceedings of the 7th USENIX conference on Networked systems design and implementation*, NSDI, 2010.

[4] Krste Asanovic, Ras Bodik, Bryan Christopher Catanzaro, Joseph James Gebis, Parry Husbands, Kurt Keutzer, David A. Patterson, William Lester Plishker, John Shalf, Samuel Webb Williams, and Katherine A. Yelick. The landscape of parallel computing research: A view from berkeley. Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, Dec 2006.

[5] James Balfour, William Dally, David Black-Schaffer, Vishal Parikh, and Jong-Soo Park. An energy-efficient processor architecture for embedded systems. *IEEE Computer Architecture Letters*, 7:29–32, January 2008.

[6] James Balfour and William J. Dally. Design tradeoffs for tiled CMP on-chip networks. In *Proc. of the 20th annual International Conference on Supercomputing*, 2006.

[7] A. Banerjee and S.W. Moore. Flow-aware allocation for on-chip networks. In *Networks-on-Chip, 2009. NoCS 2009. 3rd ACM/IEEE International Symposium on*, 2009.

[8] Arnab Banerjee, Robert Mullins, and Simon Moore. A power and energy exploration of network-on-chip architectures. In *NOCS '07: Proceedings of the First International Symposium on Networks-on-Chip*, pages 163–172, 2007.

[9] P. Baran. On distributed communication networks. In *IEEE Transactions on communication systems*, 1964.

[10] Daniel U. Becker and William J. Dally. Allocator implementations for network-on-chip routers. In *Proceedings of the 2009 ACM/IEEE Conference on Supercomputing*, 2009.

[11] S. Bell, B. Edwards, J. Amann, R. Conlin, K. Joyce, V. Leung, J. MacKay, M. Reif, Liewei Bao, J. Brown, M. Mattina, Chyi-Chang Miao, C. Ramey, D. Wentzlaff, W. Anderson, E. Berger, N. Fairbanks, D. Khan, F. Montenegro, J. Stickney, and J. Zook. TILE64 processor: A 64-core SoC with mesh interconnect. In *International Solid-State Circuits Conference*, page 88, 2008.

[12] Christian Bienia. *Benchmarking Modern Multiprocessors.* PhD thesis, Princeton University, January 2011.

[13] Tobias Bjerregaard and Shankar Mahadevan. A survey of research and practices of network-on-chip. *ACM Computing Surveys*, 38(1):1, 2006.

[14] A. Borodin, Y. Rabani, and B. Schieber. Deterministic many-to-many hot potato routing. *IEEE Transactions on Parallel and Distributed Systems*, 8(6):587 –596, jun 1997.

[15] J.T. Brassil and R.L. Cruz. Bounds on maximum delay in networks with deflection routing. *IEEE Transactions on Parallel and Distributed Systems*, 6(7):724 –732, jul 1995.

[16] Costas Busch, Maurice Herlihy, and Roger Wattenhofer. Routing without flow control. In *Proceedings of the 13th annual ACM Symposium on Parallel Algorithms and Architectures*, 2001.

[17] Luca P. Carloni. The role of back-pressure in implementing latency-insensitive systems. *Electron. Notes Theor. Comput. Sci.*, 146:61–80, 2006.

[18] A.P. Chandrakasan, S. Sheng, and R.W. Brodersen. Low-power cmos digital design. *Solid-State Circuits, IEEE Journal of*, 27(4):473 –484, apr 1992.

[19] David Graeme Chinnery. *Low Power Design Automation*. PhD thesis, 2006.

[20] Nicola Concer, Michele Petracca, and Luca P. Carloni. Distributed flit-buffer flow control for networks-on-chip. In *Proceedings of the 6th IEEE/ACM/IFIP international conference on hardware/software codesign and system synthesis*, CODES+ISSS '08, pages 215–220, 2008.

[21] Jordi Cortadella, Mike Kishinevsky, and Bill Grundmann. Synthesis of synchronous elastic architectures. In *Proceedings of the 43rd annual Design Automation Conference*, pages 657–662, 2006.

[22] William J. Dally. Virtual-channel flow control. *IEEE Transactions on Parallel and Distributed Systems*, 3(2), 1992.

[23] William J. Dally and Brian Towles. Route packets, not wires: On-chip interconnection networks. In *Proceedings of the 38th annual Design Automation Conference*, 2001.

[24] William J. Dally and Brian Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., 2003.

[25] W.J. Dally. Express cubes: improving the performance of k-ary n-cube interconnection networks. In *IEEE Transactions on Computers*, volume 40, pages 1016–1023, Sep 1991.

[26] Binh Vien Dao, Sudhakar Yalamanchili, and Jose Duato. Architectural support for reducing communication overhead in multiprocessor interconnection networks. In *Proceedings of the 3rd IEEE Symposium on High-Performance Computer Architecture*, HPCA '97, 1997.

[27] Giovanni de Micheli and Luca Benini. Networks on chip: A new paradigm for systems on chip design. In *DATE '02: Proceedings of the conference on Design, Automation and Test in Europe*, page 418, 2002.

[28] Giorgos Dimitrakopoulos and Kostas Galanopoulos. Switch allocator for bufferless network-on-chip routers. In *Proceedings of the Fifth International Workshop on Interconnection Network Architecture: On-Chip, Multi-Chip*, INA-OCMC '11, 2011.

[29] J. Duato. A new theory of deadlock-free adaptive multicast routing in wormhole networks. In *Proceedings of the Fifth IEEE Symposium on Parallel and Distributed Processing*, pages 64–71, 1993.

[30] Chris Fallin, Chris Craik, and Onur Mutlu. CHIPPER: A low-complexity bufferless deflection router. In *HPCA: 17th IEEE International Symposium on High Performance Computer Architecture*, February 2011.

[31] L R Ford and D R Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8(3), 1956.

[32] Mike Galles. Spider: A high-speed network interconnect. *IEEE Micro*, 17(1):34–39, 1997.

[33] F. Gilabert, M. E. Gomez, S. Medardoni, and D. Bertozzi. Improved utilization of noc channel bandwidth by switch replication for cost-effective multi-processor

systems-on-chip. In *NOCS '10: Proceedings of the Fourth International Symposium on Networks-on-Chip*, pages 165–172, 2010.

[34] C. Gomez, M.E. Gomez, P. Lopez, and J. Duato. BPS: A bufferless switching technique for NoCs. In *Workshop on Interconnection Network Architectures*, 2008.

[35] Crispín Gómez, María E. Gómez, Pedro López, and José Duato. Reducing packet dropping in a bufferless NoC. In *Proceedings of the 14th international Euro-Par conference on Parallel Processing*, 2008.

[36] P. Gratz, B. Grot, and S.W. Keckler. Regional congestion awareness for load balance in networks-on-chip. In *HPCA '08: Proceeding of the 14th international symposium on high-performance computer architecture*, pages 203 –214, 2008.

[37] P. Gratz, Changkyu Kim, R. McDonald, S.W. Keckler, and D. Burger. Implementation and evaluation of on-chip network architectures. In *Proceedings of the 24th International Conference on Computer Design*, 2006.

[38] A.G. Greenberg and J. Goodman. Sharp approximate models of deflection routing in mesh networks. *IEEE Transactions on Communications*, 41(1):210 –223, jan 1993.

[39] Boris Grot, Joal Hestness, and Stephen W. Kecklerand Onur Mutlu. Express cube topologies for on-chip interconnects. pages 163–174, 2009.

[40] Boris Grot and Stephen W. Keckler. Scalable on-chip interconnect topologies. In *2nd Workshop on Chip Multiprocessor Memory Systems and Interconnects*, 2008.

[41] Pankaj Gupta and Nick McKeown. Designing and implementing a fast crossbar scheduler. *IEEE Micro*, 19:20–28, 1999.

[42] Andreas Hansson, Kees Goossens, and Andrei Rădulescu. Avoiding message-dependent deadlock in network-based systems on chip. *VLSI Design*, 2007.

[43] David Harris and The Pennsylvania State University CiteSeer Archives. Local stall propagation. 2000.

[44] Mitchell Hayenga, Natalie Enright Jerger, and Mikko Lipasti. SCARAB: a single cycle adaptive routing and bufferless network. In *MICRO 42: Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 244–254, New York, NY, USA, 2009. ACM.

[45] Ron Ho, Ken Mai, and Mark Horowitz. Efficient on-chip global interconnects. In *Symposium on VLSI Circuits*, pages 271–274, 2003.

[46] Raymond R Hoare, Zhu Ding, and Alex K Jones. A near-optimal real-time hardware scheduler for large cardinality crossbar switches. In *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, 2006.

[47] Adolfy Hoisie and Vladimir Getov. Extreme-scale computing-where 'just more of the same' does not work. *IEEE Computer*, 42(11):24 –26, 2009.

[48] Yatin Hoskote, Sriram Vangal, Arvind Singh, Nitin Borkar, and Shekhar Borkar. A 5-ghz mesh interconnect for a teraflops processor. *IEEE Micro*, 27(5):51–61, 2007.

[49] C.J. Hughes, Changkyu Kim, and Yen-Kuang Chen. Performance and energy implications of many-core caches for throughput computing. *Micro, IEEE*, 30(6):25 –35, November-Decembe 2010.

[50] International Technology Roadmap for Semiconductors.

[51] Chris. Jackson and Simon J.. Hollis. Skip-links: A dynamically reconfiguring topology for energy-efficient nocs. In *System on Chip (SoC), 2010 International Symposium on*, 2010.

[52] H.M. Jacobson, P.N. Kudva, P. Bose, P.W. Cook, S.E. Schuster, E.G. Mercer, and C.J. Myers. Synchronous interlocked pipelines. In *Proceedings of the 8th international symposium on asynchronous circuits and systems*, 2002.

[53] Eric H. Jensen, Gary W. Hagensen, and Jeffrey M. Broughton. A new approach to exclusive data access in shared memory multiprocessors. (Technical Report UCRL-97663), Nov 1987.

[54] Andrew B. Kahng, Bin Li, Li-Shiuan Peh, and Kambiz Samadi. Orion 2.0: A fast and accurate NoC power and area model for early-stage design space exploration. In *Proceedings of the 46th annual conference on Design automation*, pages 423–428, 2009.

[55] Andrew B. Kahng, Bill Lin, and Kambiz Samadi. Improved on-chip router analytical power and area modeling. In *Proceedings of the 2010 Asia and South Pacific Design Automation Conference*, ASPDAC '10, pages 241–246, 2010.

[56] John Kim, Wiliam J. Dally, Steve Scott, and Dennis Abts. Technology-driven, highly-scalable dragonfly topology. In *ISCA '08: Proceedings of the 35th Annual International Symposium on Computer Architecture*, pages 77–88, 2008.

[57] John Kim, William J. Dally, and Dennis Abts. Flattened butterfly: a cost-efficient topology for high-radix networks. In *Proceedings of the 34th annual International Symposium on Computer Architecture*, 2007.

[58] Jong H. Kim, Ziqiang Liu, and Andrew A. Chien. Compressionless routing: a framework for adaptive and fault-tolerant routing. *SIGARCH Computer Architecture News*, 22(2):289–300, 1994.

[59] Jongman Kim, Dongkook Park, T. Theocharides, N. Vijaykrishnan, and Chita R. Das. A low latency router supporting adaptivity for on-chip interconnects. In *DAC '05: Proceedings of the 42nd annual conference on Design automation*, pages 559–564, 2005.

[60] Jongman Kim, Dongkook Park, T. Theocharides, N. Vijaykrishnan, and Chita R. Das. A low latency router supporting adaptivity for on-chip interconnects. In *Proceedings of the 42nd annual Design Automation Conference*, DAC '05, pages 559–564, 2005.

[61] Avinash Kodi, Ashwini Sarathy, and Ahmed Louri. Design of adaptive communication channel buffers for low-power area-efficient network-on-chip architecture. In *ANCS '07: Proceedings of the 3rd ACM/IEEE Symposium on Architecture for Networking and Communications Systems*, pages 47–56, 2007.

[62] C. P. Kruskal and M. Snir. The performance of multistage interconnection networks for multiprocessors. *IEEE Transanctions on Computers*, pages 1091–1098, December 1983.

[63] A. Kumar, P. Kundu, A.P. Singh, L.-S. Peh, and N.K. Jhay. A 4.6Tbits/s 3.6GHz single-cycle NoC router with a novel switch allocator in 65nm CMOS. In *Proceedings of the 25th International Conference on Computer Design*, 2007.

[64] Amit Kumar, Li-Shiuan Peh, and Niraj K. Jha. Token flow control. In *Proceedings of the 41st annual IEEE/ACM International Symposium on Microarchitecture*, 2008.

[65] Amit Kumar, Li-Shiuan Peh, Partha Kundu, and Niraj K. Jha. Express virtual channels: towards the ideal interconnection fabric. In *Proceedings of the 34th annual international symposium on Computer architecture*, 2007.

[66] Rakesh Kumar, Victor Zyuban, and Dean M. Tullsen. Interconnections in Multi-Core architectures: Understanding mechanisms, overheads and scaling. In *Proceedings of the 32nd annual International Symposium on Computer Architecture*, pages 408–419, 2005.

[67] Dara Kusic, Raymond Hoare, Alex K. Jones, Joshua Fazekas, and John Foster. Extracting speedup from c-code with poor instruction-level parallelism. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 14 - Volume 15*, IPDPS '05, 2005.

[68] Michael M. Lee, John Kim, Dennis Abts, Michael Marty, and Jae W. Lee. Approximating age-based arbitration in on-chip networks. In *Proceedings of the 19th international conference on Parallel architectures and compilation techniques*, PACT '10, 2010.

[69] Charles E. Leiserson. Fat-trees: universal networks for hardware-efficient supercomputing. In *IEEE Transactions on Computers*, volume 34, pages 892–901, 1985.

[70] Jian Liu, Li-Rong Zheng, and Hannu Tenhunen. A guaranteed-throughput switch for network-on-chip. In *SOC '03: Proceedings of the International Symposium on System-on-Chip*, pages 31–34, 2003.

[71] Zhonghai Lu, Mingchen Zhong, and Axel Jantsch. Evaluation of on-chip networks using deflection routing. In *Proceedings of the 16th ACM Great Lakes symposium on VLSI*, 2006.

[72] Chi-Keung Luk, Robert Cohn, Robert Muth, Harish Patil, Artur Klauser, Geoff Lowney, Steven Wallace, Vijay Janapa Reddi, and Kim Hazelwood. Pin: building customized program analysis tools with dynamic instrumentation. In *Proceedings of the 2005 ACM SIGPLAN conference on Programming language design and implementation*, 2005.

[73] R. Manevich, I. Walter, I. Cidon, and A. Kolodny. Best of both worlds: A bus enhanced noc (BENoC). In *2010 IEEE 26th Conventiong of Electrical and Electronics Engineers in Israel (IEEEI)*, 2010.

[74] Nick McKeown. The iSLIP scheduling algorithm for input-queued switches. *IEEE/ACM Transanctions on Networking*, 7:188–201, 1999.

[75] George Michelogiannakis, James Balfour, and William J. Dally. Elastic buffer flow control for on-chip networks. In *HPCA '09: Proceeding of the 15th International Symposium on High-Performance Computer Architecture*, pages 151–162, 2009.

[76] George Michelogiannakis, Daniel U. Becker, and William J. Dally. Evaluating elastic buffer and wormhole flow control. *IEEE Transanctions on Computers.*

[77] George Michelogiannakis and William J. Dally. Router designs for elastic buffer on-chip networks. In *SC '09: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, pages 1–10, 2009.

[78] George Michelogiannakis, Nan Jiang, Daniel U. Becker, and William J. Dally. Packet chaining: Efficient single-cycle allocation for on-chip networks. *IEEE Computer Architecture Letters*, 2011.

[79] George Michelogiannakis, Nan Jiang, Daniel U. Becker, and William J. Dally. Packet chaining: Efficient single-cycle allocation for on-chip networks. In *Proceedings of the 44st annual IEEE/ACM International Symposium on Microarchitecture*, 2011.

[80] George Michelogiannakis, Dionisios Pnevmatikatos, and Manolis Katevenis. Approaching ideal NoC latency with pre-configured routes. In *NOCS '07: Proceedings of the First International Symposium on Networks-on-Chip*, pages 153–162, 2007.

[81] George Michelogiannakis, Daniel Sanchez, William J. Dally, and Christos Kozyrakis. Evaluating bufferless flow control for on-chip networks. In *NOCS '10: Proceedings of the Fourth International Symposium on Networks-on-Chip*, pages 9–16, 2010.

[82] Masayuki Mizuno, , William J. Dally, and Hideaki Onishi. Elastic interconnects: repeater-inserted long wiring capable of compressing and decompressing data. In *ISSCC '01: Proceedings of IEEE International Solid-State Circuits Conference*, pages 346–347, 464, 2001.

[83] Thomas Moscibroda and Onur Mutlu. A case for bufferless routing in on-chip networks. *SIGARCH Computer Architecture News*, 37(3):196–207, 2009.

[84] Shubhendu S. Mukherjee, Federico Silla, Peter Bannon, Joel Emer, Steve Lang, and David Webb. A comparative study of arbitration algorithms for the alpha 21364 pipelined router. *SIGARCH Computer Architecture News*, 30:223–234, 2002.

[85] Robert Mullins, Andrew West, and Simon Moore. Low-latency virtual-channel routers for on-chip networks. In *Proceedings of the 31st annual International Symposium on Computer Architecture*, 2004.

[86] L. Ni and P McKinley. A survey of wormhole routing techniques in direct networks. pages 492–506, 2000.

[87] Chrysostomos Nicopoulos, Dongkook Park, Jongman Kim, Narayanan Vijaykrishnan, Masin S. Yousif, and Chita R. Das. ViChaR: A dynamic virtual channel regulator for network-on-chip routers. In *MICRO '06: Proceedings of the 39th annual International Symposium on Microarchitecture*, 2006.

[88] Chrysostomos Nicopoulos, Aditya Yanamandra, Suresh Srinivasan, Vijaykrishnan Narayanan, and Mary Jane Irwin. Variation-aware low-power buffer design. In *Proceedings of The Asilomar Conference on Signals, Systems, and Computers*, 2007.

[89] Erland Nilsson, Mikael Millberg, Johnny Oberg, and Axel Jantsch. Load distribution with the proximity congestion awareness in a network on chip. In *DATE '03: Proceedings of the Conference on Design, Automation and Test in Europe*, 2003.

[90] Erland Nilsson, Mikael Millberg, Johnny Oberg, and Axel Jantsch. Load distribution with the proximity congestion awareness in a network on chip. In *Proceedings of the conference on Design, Automation and Test in Europe*, DATE '03, 2003.

[91] Kunle Olukotun. *Chip Multiprocessor Architecture: Techniques to Improve Throughput and Latency*. Morgan and Claypool Publishers, 1st edition, 2007.

[92] John D. Owens, William J. Dally, Ron Ho, D. N. Jayasimha, Stephen W. Keckler, and Li-Shiuan Peh. Research challenges for on-chip interconnection networks. 27(5):96–108, 2007.

[93] Dongkook Park, Reetuparna Das, Chrysostomos Nicopoulos, Jongman Kim, Narayanan Vijaykrishnan, Ravishankar K Iyer, and Chita R Das. Design of a dynamic priority-based fast path architecture for on-chip interconnects. In *Proceedings of the 15th Symposium on High Performance Interconnects*, 2007.

[94] Li-Shiuan Peh and W. J. Dally. Flit-reservation flow control. In *Proceeding of the 6th International Symposium on High-Performance Computer Architecture (HPCA)*, 2000.

[95] Li-Shiuan Peh and William J. Dally. A delay model and speculative architecture for pipelined routers. In *Proceedings of the 7th International Symposium on High-Performance Computer Architecture*, 2001.

[96] Matthew A. Postiff, David A. Greene, Gary S. Tyson, and Trevor N. Mudge. The limits of instruction level parallelism in spec95 applications. *SIGARCH Computer Architecture News*, 27(1):31–34, 1999.

[97] Antonio Pullini, Federico Angiolini, Davide Bertozzi, and Luca Benini. Fault tolerance overhead in network-on-chip flow control schemes. In *SBCCI '05: Proceedings of the 18th annual symposium on Integrated circuits and system design*, pages 224–229, 2005.

[98] Daniel Sanchez, George Michelogiannakis, and Christos Kozyrakis. An analysis of interconnection networks for large scale chip-multiprocessors. *ACM Transactions on Architecture and Code Optimization*, 7(1):4:1–4:28, 2010.

[99] S. L. Scott and G. S. Sohi. Using feedback to control tree saturation in multistage interconnection networks. In *ISCA '89: Proceedings of the 16th annual international symposium on Computer architecture*, pages 167–176, 1989.

[100] Arjun Singh. *Load-Balanced Routing in Interconnection Networks.* PhD in electrical engineering, Stanford University, 2005.

[101] Marc Snir, Steve Otto, Steven Huss-Lederman, David Walker, and Jack Dongarra. *MPI-The Complete Reference, Volume 1: The MPI Core.* MIT Press, Cambridge, MA, USA, 2nd. (revised) edition, 1998.

[102] David Tam, Reza Azimi, and Michael Stumm. Thread clustering: sharing-aware scheduling on smp-cmp-smt multiprocessors. In *Proceedings of the 2nd ACM*

SIGOPS/EuroSys European Conference on Computer Systems 2007, EuroSys '07, pages 47–58, 2007.

[103] Y. Tamir and H. C. Chi. Symmetric crossbar arbiters for VLSI communication switches. *IEEE Transactions on Parallel and Distributed Systems*, 4:13–27, 1993.

[104] La Roy Tumes. Routing and flow control in tymnet. *IEEE Transactions on Communications*, 29(4):392 – 398, 1981.

[105] Yoshio Turner and Yuval Tamir. Deadlock-free connection-based adaptive routing with dynamic virtual circuits. *Journal of Parallel and Distributed Computing*, 67(1):13–32, 2007.

[106] Hang-Sheng Wang, Xinping Zhu, Li-Shiuan Peh, and Sharad Malik. Orion: a power-performance simulator for interconnection networks. In *Proceedings of the 35th annual ACM/IEEE International Symposium on Microarchitecture*, 2002.

[107] Hangsheng Wang, Li-Shiuan Peh, and Sharad Malik. Power-driven design of router microarchitectures in on-chip networks. In *Proc. of the 36th annual IEEE/ACM Intl. Symp. on Microarchitecture*, 2003.

[108] Yu Wang, Jiang Xu, Yan Xu, Weichen Liu, and Huazhong Yang. Power gating aware task scheduling in mpsoc. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 19(10):1801 –1812, October 2011.

[109] Daniel Wiklund and Dake Liu. SoCBUS: Switched network-on-chip for hard real time embedded systems. In *IPDPS '03: Proceedings of the 17th International Symposium on Parallel and Distributed Processing*, page 78.1, 2003.

[110] P.T. Wolkotte, G.J.M. Smit, G.K. Rauwerda, and L.T. Smit. An energy-efficient reconfigurable circuit-switched network-on-chip. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium*, april 2005.

[111] Steven Cameron Woo, Moriyoshi Ohara, Evan Torrie, Jaswinder Pal Singh, and Anoop Gupta. The splash-2 programs: characterization and methodological considerations. In *Proceedings of the 22nd annual International Symposium on Computer architecture*, ISCA '95, 1995.

[112] Thomas Canhao Xu, Pasi Liljeberg, and Hannu Tenhunen. Process scheduling for future multicore processors. In *Proceedings of the Fifth International Workshop on Interconnection Network Architecture: On-Chip, Multi-Chip*, INA-OCMC '11, pages 15–18, 2011.

[113] Yi Xu, Yu Du, Bo Zhao, Xiuyi Zhou, Youtao Zhang, and Yang Jun. A low-radix and low-diameter 3d interconnection network design. pages 30–42, 2009.

[114] L. Zhao, R. Iyer, S. Makineni, R. Illikkal, J. Moses, and D. Newell. Constraint-aware large-scale cmp cache design. In *Proceedings of the 14th international conference on High performance computing*, HiPC'07, pages 161–171, 2007.