Appears in the Proceedings of the 4th ACM/IEEE International Symposium on Networks-on-Chip (NOCS 2010)

Evaluating Bufferless Flow Control for On-Chip Networks

George Michelogiannakis, Daniel Sanchez, William J. Dally, Christos Kozyrakis
Electrical Engineering Department, Stanford University
E-mail: {mihelog, sanchezd, dally, kozyraki}@stanford.edu

***Abstract*—With the emergence of on-chip networks, the power consumed by router buffers has become a primary concern. Bufferless flow control addresses this issue by removing router buffers, and handles contention by dropping or deflecting flits. This work compares virtual-channel (buffered) and deflection (packet-switched bufferless) flow control. Our evaluation includes optimizations for both schemes: buffered networks use custom SRAM-based buffers and empty buffer bypassing for energy efficiency, while bufferless networks feature a novel routing scheme that reduces average latency by 5%. Results show that unless process constraints lead to excessively costly buffers, the performance, cost and increased complexity of deflection flow control outweigh its potential gains: bufferless designs are only marginally (up to 1.5%) more energy efficient at very light loads, and buffered networks provide lower latency and higher throughput per unit power under most conditions.**

I. Introduction

Continued improvements in VLSI technology enable integration of an increasing number of logic blocks on a single chip. Scalable packet-switched *networks-on-chip* (NoCs) [7] have been developed to serve the communication needs of such large systems. As system size increases, these interconnects become a crucial factor in the performance and cost of the chip.

Compared to off-chip networks, on-chip wires are cheaper and buffer cost is more significant [7]. Router buffers are used to queue packets or flits that cannot be routed immediately due to contention [6]. Several proposals eliminate router buffers to reduce NoC cost. In these *bufferless* schemes, contending packets or flits are either *dropped* and retransmitted by their source [9] or *deflected* [2], [18] to a free output port. Frequent retransmissions or deflections degrade network performance. However, under light load, dropping or deflecting may occur infrequently enough to have a small impact on performance.

Bufferless flow control proposals often report large area and power savings compared to conventional buffered networks (e.g. 60% area and up to 39% energy savings in a conventional CMP network [18]). However, previous work has aimed to reduce the cost of router buffers. For example, by using custom SRAM-based implementations, buffers can consume as little as 6% of total network area and 15.5% of total network power in a flattened butterfly (FBFly) network [14], [17]. Furthermore, flits can bypass empty buffers in the absence of contention [24], reducing dynamic power consumption in lightly-loaded networks. 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.

In this paper, we compare a state-of-the-art packet-switched bufferless network with deflecting flow control, BLESS [18], and the currently-dominant virtual channel (VC) buffered flow control [6]. To perform an equitable comparison, we optimize both networks. In particular, VC networks feature efficient custom SRAM buffers and empty buffer bypassing. We also propose 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. In an 8×8 2D mesh with 5×5 routers and uniform traffic, this routing scheme reduces latency by 5% over dimension-ordered routing (DOR).

We find that bufferless flow control provides a minimal advantage at best: in a lightly-loaded 8×8 2D mesh, bufferless flow control reduces power consumption by only 1.5%, mostly due to buffer leakage power, when using a high-performance, high-leakage process. However, at medium or high loads the buffered network offers significantly better performance and higher power efficiency with 21% more throughput per unit power, as well as a 17% lower average latency at a 20% flit injection rate. The buffered network becomes more energy efficiency at flit injection rates of 7% (11% with low-swing channels). Buffer optimizations play a crucial role: at a flit injection rate of 20%, buffers without bypassing consume 8.5× the dynamic power with bypassing. Finally, the age-based allocator required to prevent livelocks in BLESS is 81% slower than an input-first separable switch allocator used in VC flow control.

The rest of this paper is organized as follows: Section II provides the necessary background on bufferless interconnects. Section III discusses our evaluation methodology. Section IV presents our novel routing scheme for BLESS. Section V examines the implications for router microarchitecture. In Section VI we present our evaluation and results. Section VII discusses further design parameters that can affect our comparisons. Finally, Section VIII concludes this paper.

II. Background

Deflection flow control was first proposed as "hot-potato" routing in off-chip networks [2]. Recent work has found that network topology is the most important factor affecting performance, and that global or history-related deflection criteria are beneficial [16]. Furthermore, dynamic routing can be used to provide an upper bound for delivery time [4].

In this paper, we consider BLESS, a state-of-the-art bufferless deflection flow control proposal for NoCs [18]. In BLESS, flits bid for their preferred output. If the allocator is unable to grant that output, the flit is deflected to any free output. This requires that routers have at least as many outputs as inputs. Flits bid for a single output port, following deterministic DOR. To avoid livelock, older flits are given priority. Finally, injecting flits to a router requires a free output port to avoid deflecting flits to ejection ports.

Two BLESS variants were evaluated in [18], 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, imposing 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, worms may still have to be split under congestion, and WORM-BLESS still needs to be able to route all flits independently.

Due to the lack of VCs, traffic classes need to be separated with extra mechanisms or by duplicating physical channels. Additionally, BLESS requires extra logic and buffering at network destinations to reorder flits of the same packet arriving out of order. The number of packets that can arrive interleaved is practically unbounded, unlike VC networks where destinations just need a FIFO buffer per VC.

While most bufferless proposals either drop or deflect flits, both mechanisms can be combined [8]. Other techniques also eliminate buffers: circuit-switching relies on establishing end-to-end circuits in which flits never contend [15]. Finally, elastic buffer flow control [17] uses channels as distributed FIFOs in place of router buffers.

III. Methodology

We use a modified version of Booksim [5] for cycle-accurate microarchitecture-level network simulation. To estimate area and power we use ITRS predictions for a 32nm high-performance process [12], operating at 70°C. Modeling buffer costs accurately is fundamental in our study. Orion [23] is the standard modeling tool in NoC studies, but a recent study shows that it can lead to large errors [13], and the update fixing these issues was not available at the time of this work. Instead, we use the models from Balfour and Dally [1], which are derived from basic principles, and validate SRAM models using HSPICE.

We assume a clock frequency of 2GHz and 512-bit packets. We model channel wires as being 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. Our conservative low-swing model has 30% of the full-swing repeated wire traversal power and twice the channel area [11]. 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.

We choose FLIT-BLESS for our comparisons. FLIT-BLESS performs better than WORM-BLESS [18], but incurs extra overhead because all flits contain routing information. However, in our evaluation we do not model this overhead, giving BLESS a small advantage over buffered flow control.

We use two topologies 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 FBFly [14] 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, a typical die size in modern processes [22].

We assume a two-stage router design. The VC network features input-first separable round-robin allocators, speculative switch allocation [21] and input buffer bypassing [24]. No communication protocol is assumed. The VC network uses DOR for the mesh and FBFly. The deflection network uses multidimensional routing, explained in Section IV. 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 [5]. 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.

IV. Routing in Bufferless Networks

BLESS networks use DOR [18]. In VC networks, DOR prevents cyclic network dependencies without extra VCs. However, in bufferless networks flits never block waiting for buffers, so there can be no network deadlocks, making DOR unnecessary.

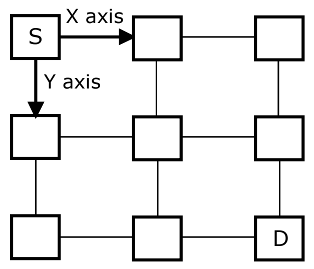
We propose two oblivious routing algorithms that decrease deflection probability. We observe that a flit often has several *productive* outputs (i.e. outputs that would get the flit closer to its destination). For example, in a 2D mesh, those are the two outputs shown in Figure 1(a), unless the flit is already at one of the axes of its final destination. Our first routing algorithm, multi-dimensional routing (MDR), *exploits choice* by having flits request all of their productive outputs. If both outputs are available, the switch allocator assigns one pseudorandomly.

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 MDR (PMDR). Figure 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.

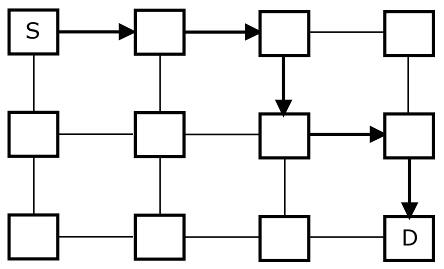
Figure 2 compares DOR, MDR and PMDR in mesh and FBFly bufferless networks. In the mesh, MDR offers 5% lower average latency than DOR and equal maximum throughput. In the FBFly, MDR achieves 2% lower average latency and 3% higher maximum throughput. Under a sample 20% flit injection rate, 13% more flits 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.

V. Router Microarchitecture

This section explores router microarchitecture issues pertinent to our study of bufferless networks.



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



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

Figure 1. MDR and PMDR routing algorithms for deflection bufferless networks.

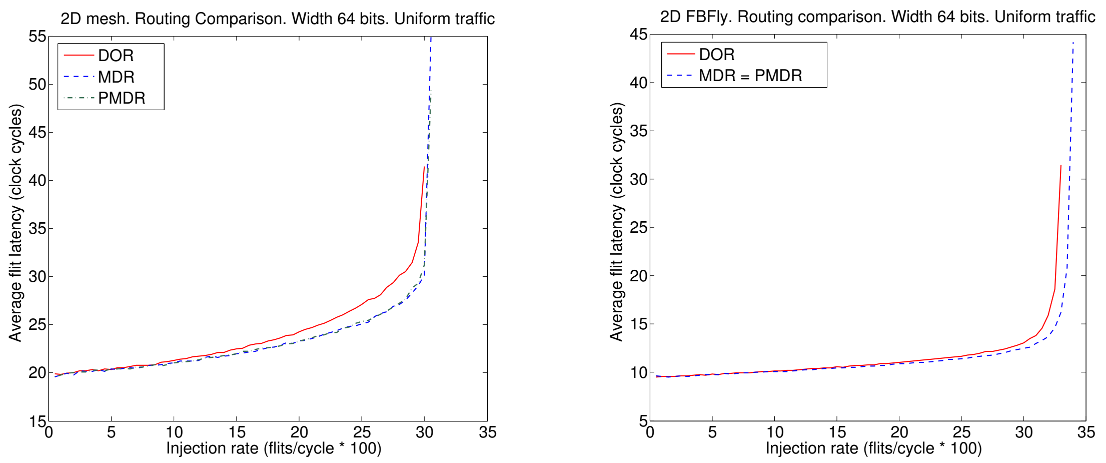


Figure 2. Comparison of DOR, MDR and PMDR routing algorithms.

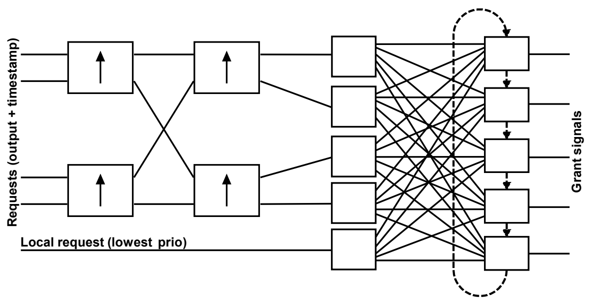


Figure 3. BLESS allocator implementation.

A. Allocator Complexity

We design and implement an age-based BLESS allocator, shown in Figure 3. Requests are partially ordered according to their age by the sorting blocks shown on the left, each of which sorts two requests. The output arbiters satisfy the oldest request first, and forward conflicting requests to the next arbiter. The oldest incoming flit is guaranteed to not be deflected, thus preventing livelocks. Although the logic of each arbiter is simple, all requests need to be granted [18] because all flits need to be routed. This creates a long critical path that passes through each 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, as it trades off accuracy for cycle time.

Table I compares our BLESS allocator and an input-first separable speculative switch allocator with round-robin arbiters for a VC network with 2 VCs [3]. Dynamic power is estimated by applying Synopsys Design Compiler's default activity factor

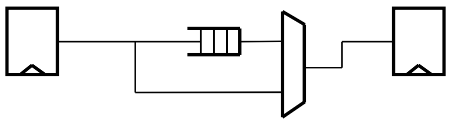


Figure 4. Empty buffer bypassing.

Table I
ALLOCATOR SYNTHESIS COMPARISON.

Aspect	Sep. I.-F.	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%
Dyn. power (mW)	0.48	0.27	-44%

to all inputs. The VC network switch allocator represents a reasonable design and has comparable cycle time to other separable and wavefront allocators [3], [5]. We compare against the switch allocator because a similar VC allocator has a shorter critical path and speculation parallelizes VC and switch allocation [3].

As Table I 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 has a larger area than the age-based allocator, and consumes clock tree and flip-flop power because it needs to maintain state for the round-robin arbitration. However, the difference is a small fraction of the overall router power.

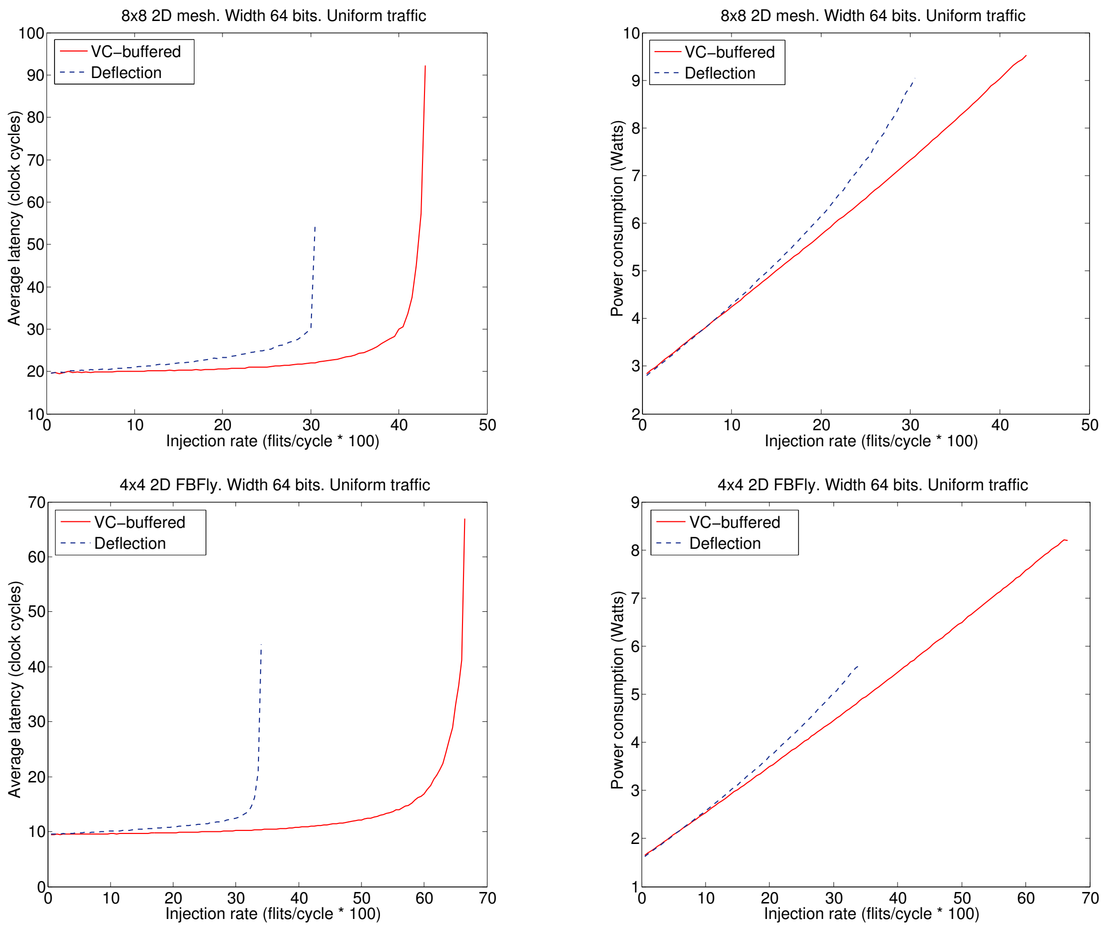


Figure 5. Latency and power comparison.

B. *Buffer Cost*

Our models assume efficient custom SRAM blocks for buffers [1], 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 flip-flop arrays will likely make buffers more costly, increasing the motivation for eliminating them or reducing their size.

Additionally, we use empty buffer bypassing [24], shown in Figure 4. This allows flits to bypass empty buffers in the absence of contention, reducing dynamic power under light load.

We find that these schemes suffice to implement efficient buffers, but additional techniques could be applied. For instance, leakage-aware buffers [20] 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 [19].

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 flip-flop arrays for buffers. With this implementation, the flip-flop 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.

VI. Evaluation

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

A. *Latency and Throughput*

Figure 5 presents 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, 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. Deflecting flits increases the average channel activity factor for a given throughput. As shown, 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.

Due to the larger activity factor, 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.

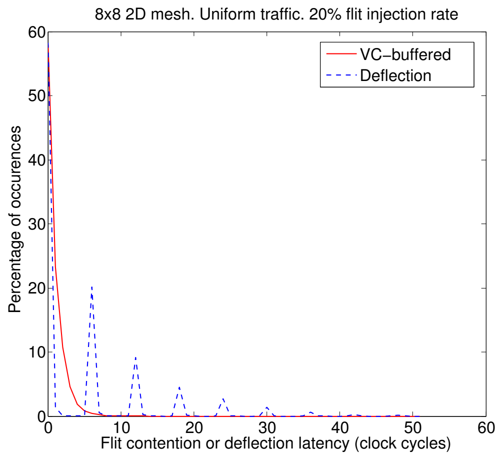


Figure 6. Blocking or deflection latency distribution.

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.

Without empty buffer bypassing, the dynamic buffer power increases significantly, as detailed in Section VI-B. 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. This emphasizes the importance of buffer bypassing.

Figure 6 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 7 shows throughput versus area and power Pareto-optimal 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 VI-B. 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 VI-B. 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.

B. Power and Area Breakdowns

Figure 8 shows the power and area breakdowns for the 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 flip-flops 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 flip-flops. Leakage power is included for buffers and channels.

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.

C. Low-swing Channels

Low-swing channels favor the deflection network because they reduce deflection energy. 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

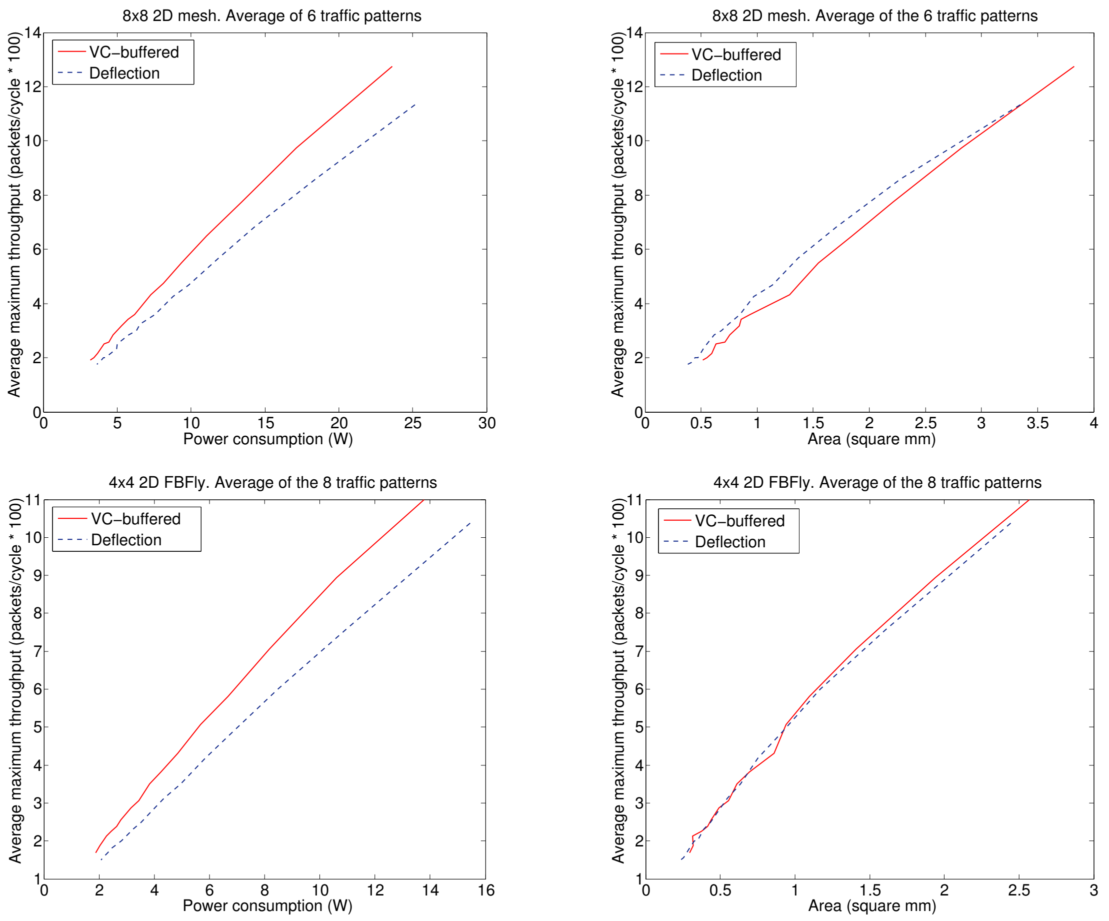


Figure 7. Throughput versus power and area Pareto-optimal curves.

(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 9 illustrates the results for the mesh.

D. *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 [10]. 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 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 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. 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.

E. *Flit Injection*

Injection in deflecting flow control requires feedback from the router because at least one output port must be free [18]. However, acquiring this information is problematic, specially if the round-trip distance between the router and the source is more than one clock cycle. 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.

F. *Process Technology*

Our evaluation uses a 32nm high-performance ITRS-based process as a worst case due to its high leakage current. To illustrate the other extreme, we use the commercial 45nm low-power library used for synthesis in Section V-A. Its leakage current is negligible. With empty buffer bypassing, the deflection network never consumes less energy than the VC network. Both

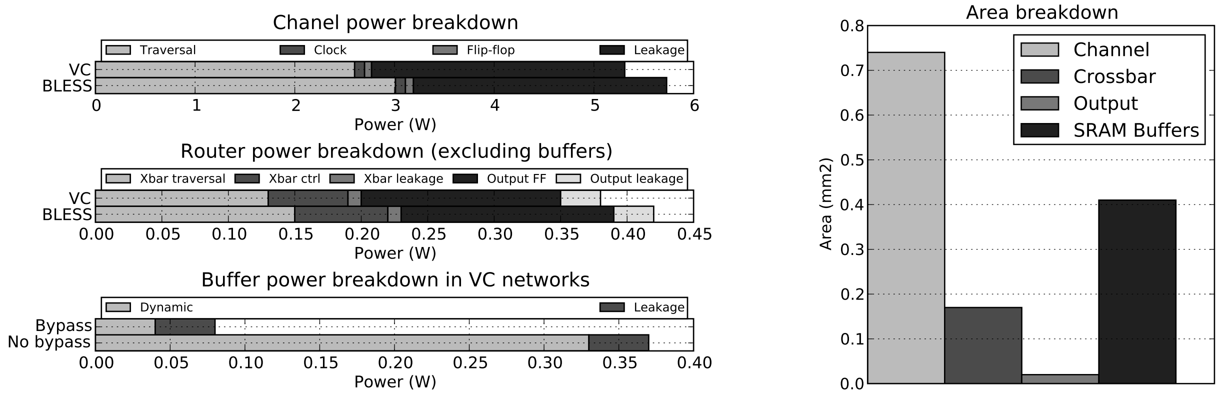


Figure 8. Power and area breakdowns for the 2D mesh under a 20% flit injection rate with full-swing channels.

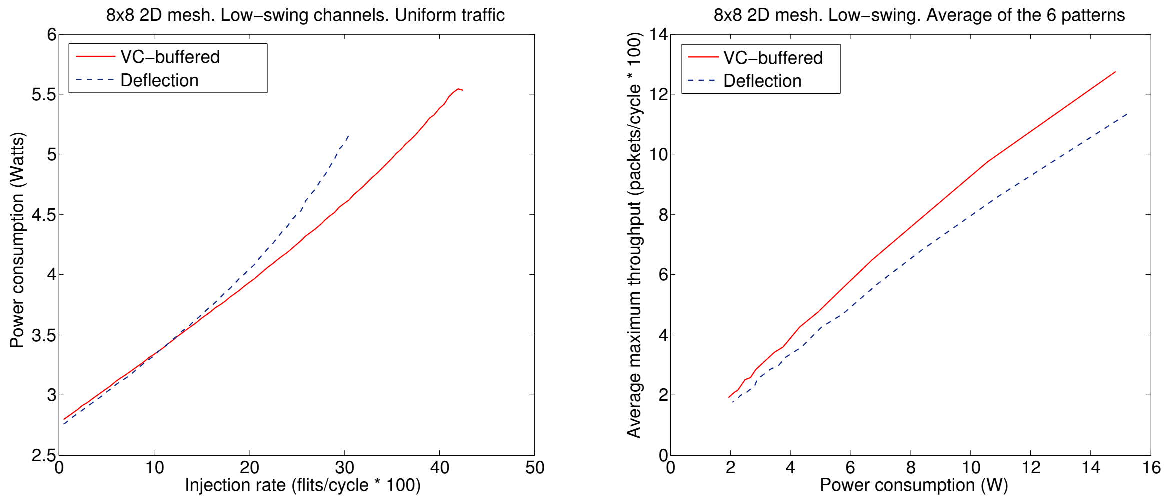


Figure 9. Power consumption with varying injection rate and throughput-power Pareto-optimal curves with low-swing channels.

consume approximately the same amount of power even for very low injection rates. Furthermore, the VC mesh described in Section VI-A 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 process.

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.

VII. Discussion

Our quantitative evaluation tries to cover 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 V-A. However, more traffic classes also increase the demand for endpoint buffering discussed in Section VI-D. For a single traffic class, we have shown that at least a buffered network with 2 VCs is more efficient than a deflection network.

Network size: While network size affects the relevant tradeoffs, smaller networks provide fewer deflection paths. The deflection and buffering probabilities are similarly affected by size. 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. Dropping, as 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 VI-A. 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. Complete 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.

VIII. Conclusions

We have 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. The deflection network with MDR consumes less power up to a flit injection rate of 7%. However, it 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. Furthermore, the BLESS allocator has an 81% larger cycle time than a separable input-first round-robin speculative switch allocator. 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. 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, we believe that design effort should be spent on more efficient buffers before considering bufferless flow control.

Acknowledgements

We sincerely thank Daniel Becker, Nathan Binkert, Jung Ho Ahn and the anonymous reviewers for their useful comments. This work was supported by the National Science Foundation under Grant CCF-0702341, the National Security Agency under Contract H98230-08-C-0272, the Stanford Pervasive Parallelism Lab, the Gigascale Systems Research Center (FCRP/GSRC). George Michelogiannakis was supported by a Robert Bosch Stanford Graduate Fellowship. Daniel Sanchez was supported by a Fundacion Caja Madrid Fellowship and a Hewlett-Packard Stanford School of Engineering Fellowship.

References

[1] J. Balfour and W. J. Dally, "Design tradeoffs for tiled CMP on-chip networks," in *Proc. of the 20th annual Intl. Conf. on Supercomputing*, 2006.
[2] P. Baran, "On distributed communication networks," in *IEEE Trans. on communication systems*, 1964.
[3] D. U. Becker and W. J. Dally, "Allocator implementations for network-on-chip routers," in *Proc. of the Conf. on High Performance Computing Networking, Storage and Analysis*, 2009.
[4] C. Busch, M. Herlihy, and R. Wattenhofer, "Routing without flow control," in *Proc. of the 13th annual ACM Symp. on Parallel Algorithms and Architectures*, 2001.
[5] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers, 2003.
[6] W. J. Dally, "Virtual-channel flow control," *IEEE Trans. on Parallel and Distributed Systems*, vol. 3, no. 2, 1992.
[7] W. J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in *Proc. of the 38th annual Design Automation Conf.*, 2001.
[8] C. Gomez, M. Gomez, P. Lopez, and J. Duato, "BPS: A bufferless switching technique for NoCs," in *Workshop on Interconnection Network Architectures*, 2008.
[9] C. Gómez, M. E. Gómez, P. López, and J. Duato, "Reducing packet dropping in a bufferless NoC," in *Proc. of the 14th intl. Euro-Par conf. on Parallel Processing*, 2008.
[10] A. Hansson, K. Goossens, and A. Rădulescu, "Avoiding message-dependent deadlock in network-based systems on chip," *VLSI Design*, 2007.
[11] R. Ho, K. Mai, and M. Horowitz, "Efficient on-chip global interconnects," in *Symp. on VLSI Circuits*, 2003.
[12] International Technology Roadmap for Semiconductors, 2007 Edition, www.itrs.net.
[13] A. Kahng, B. Li, L.-S. Peh, and K. Samadi, "Orion 2.0: A fast and accurate noc power and area model for early-stage design space exploration," in *Proc. of the conf. on Design, Automation and Test in Europe*, 2009.
[14] J. Kim, W. J. Dally, and D. Abts, "Flattened butterfly: a cost-efficient topology for high-radix networks," in *Proc. of the 34th annual Intl. Symp. on Computer Architecture*, 2007.
[15] J. Liu, L.-R. Zheng, and H. Tenhunen, "A guaranteed-throughput switch for network-on-chip," in *Proc. of the Intl. Symp. on System-on-Chip*, 2003.
[16] Z. Lu, M. Zhong, and A. Jantsch, "Evaluation of on-chip networks using deflection routing," in *Proc. of the 16th ACM Great Lakes symp. on VLSI*, 2006.
[17] G. Michelogiannakis, J. Balfour, and W. J. Dally, "Elastic buffer flow control for on-chip networks," in *Proc. of the 15th Intl. Symp. on High-Performance Computer Architecture*, 2009.
[18] T. Moscibroda and O. Mutlu, "A case for bufferless routing in on-chip networks," in *Proc. of the 36th annual Intl. Symp. on Computer Architecture*, 2009.
[19] C. Nicopoulos, D. Park, J. Kim, N. Vijaykrishnan, M. S. Yousif, and C. R. Das, "ViChaR: A dynamic virtual channel regulator for network-on-chip routers," in *Proc. of the 39th annual Intl. Symp. on Microarchitecture*, 2006.
[20] C. Nicopoulos, A. Yanamandra, S. Srinivasan, V. Narayanan, and M. J. Irwin, "Variation-aware low-power buffer design," in *Proc. of The Asilomar Conf. on Signals, Systems, and Computers*, 2007.
[21] L.-S. Peh and W. J. Dally, "A delay model and speculative architecture for pipelined routers," in *Proc. of the 7th Intl. Symp. on High-Performance Computer Architecture*, 2001.
[22] D. Sanchez, G. Michelogiannakis, and C. Kozyrakis, "An analysis of interconnection networks for large scale chip-multiprocessors," *ACM Trans. on Arch. and Code Opt.*, vol. 7, no. 1, 2010.
[23] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik, "Orion: a power-performance simulator for interconnection networks," in *Proc. of the 35th annual ACM/IEEE Intl. Symp. on Microarchitecture*, 2002.
[24] H. Wang, L.-S. Peh, and S. Malik, "Power-driven design of router microarchitectures in on-chip networks," in *Proc. of the 36th annual IEEE/ACM Intl. Symp. on Microarchitecture*, 2003.