Chapter 1
Bufferless and Minimally-Buffered Deflection Routing

Chris Fallin, Greg Nazario, Xiangyao Yu, Kevin Chang, Rachata Ausavarungnirun, Onur Mutlu

Abstract A conventional Network-on-Chip (NoC) router uses input buffers to store in-flight packets. These buffers improve performance, but consume significant power. It is possible to bypass these buffers when they are empty, reducing dynamic power, but static buffer power remains, and when buffers are utilized, dynamic buffer power remains as well. To improve energy efficiency, *bufferless deflection routing* removes input buffers, and instead uses deflection (misrouting) to resolve contention. Bufferless deflection routing is able to provide similar network performance to conventional buffered routing when the network carries light to moderate traffic, because deflections are relatively rare. However, at high network load, deflections cause unnecessary network hops, wasting power and reducing performance. In order to avoid some deflections and recover some performance, recent work has proposed to add a small buffer which holds only flits that contend with others and would have been deflected. This minimally-buffered deflection (MinBD) router improves performance relative to bufferless deflection routing without incurring the cost of a large buffer, because it can make more efficient use of a small buffer. The result is a router design which is more energy-efficient than prior buffered, bufferless, and hybrid router designs.

Chris Fallin, Greg Nazario, Kevin Chang, Rachata Ausavarungnirun, and Onur Mutlu
Carnegie Mellon University
`{cfallin,gnazario,kevincha,rausavar,onur}@cmu.edu`

Xiangyao Yu
Massachusetts Institute of Technology
`yxy@mit.edu`

1.1 Introduction

A network-on-chip is a first-order component of current and future multicore and manycore CMPs (Chip Multiprocessors) [12], and its design can be critical for system performance. As core counts continue to rise, NoCs with designs such as 2D-mesh (e.g., Tilera [51] and Intel Terascale [28]) are expected to become more common to provide adequate performance scaling. Unfortunately, packet-switched NoCs are projected to consume significant power. In the Intel Terascale 80-core chip, 28% of chip power is consumed by the NoC [28]; for MIT RAW, 36% [46]; for the Intel 48-core SCC, 10% [6]. NoC energy efficiency is thus an important design goal [4, 5].

Mechanisms have been proposed to make conventional input-buffered NoC routers more energy-efficient (i.e., use less energy per unit of performance). For example, bypassing empty input buffers [37, 50] reduces some dynamic buffer power, but static power remains.[1] Such bypassing is also less effective when buffers are not frequently empty. *Bufferless deflection routers* [17, 38] remove router input buffers completely (hence eliminating their static and dynamic power) to reduce router power. When two flits[2] contend for a single router output, one must be deflected to another output. Thus, a flit never requires a buffer in a router. By controlling which flits are deflected, a bufferless deflection router can ensure that all traffic is eventually delivered. Removing buffers yields simpler and more energy-efficient NoC designs: e.g., CHIPPER [17] reduces average network power by 54.9% in a 64-node system compared to a conventional buffered router.

Unfortunately, at high network load, deflection routing reduces performance and efficiency. This is because deflections occur more frequently when many flits contend in the network. Each deflection sends a flit further from its destination, causing unnecessary link and router traversals. Relative to a buffered network, a bufferless network with a high deflection rate wastes energy, and suffers worse congestion, because of these unproductive network hops. In contrast, a buffered router is able to hold flits (or packets) in its input buffers until the required output port is available, incurring no unnecessary hops. Thus, a buffered network can sustain higher performance at peak load, but at the cost of large buffers, which can consume significant power and die area.

The best interconnect design would obtain the energy efficiency of the bufferless approach with the high performance of the buffered approach. Neither purely bufferless deflection routing nor conventional input-buffered routing satisfy this goal. Ideally, a router would contain only a small amount of

[1] One recent estimate indicates that static power (of buffers and links) could constitute 80–90% of interconnect power in future systems [7].

[2] In a conventional bufferless deflection network, *flits* (several of which make up one packet) are independently routed, unlike most buffered networks, where a packet is the smallest independently-routed unit of traffic.

buffering, and would use this buffer space only for those flits that actually require it, rather than all flits that arrive.

In this chapter, we discuss *minimally-buffered deflection* routing (MinBD) [20], a router design which combines both bufferless and buffered paradigms in a fine-grained and efficient way. MinBD uses deflection routing, but also incorporates a small buffer. The router does not switch between modes, but instead, always operates in a minimally-buffered deflection mode, and can buffer or deflect any given flit. When a flit first arrives, it does not enter a buffer, but travels straight to the routing logic. If two flits contend for the same output, the routing logic chooses one to deflect, as in a bufferless router. However, the router can choose to buffer up to one deflected flit per cycle rather than deflecting it. This fine-grained buffering-deflection hybrid approach significantly reduces deflection rate (by 54% [20]), and improves performance, as we show. It also incurs only a fraction of the energy cost of a conventional buffered router, because only a relatively small fraction of flits are buffered (20% of all flits). MinBD provides higher energy efficiency while also providing high performance, compared to a comprehensive set of baseline router designs. In this chapter, we will discuss:

- Bufferless deflection routing [17, 38], which uses deflection in place of buffering to resolve contention between flits. We will introduce the basic design of the BLESS [38] and CHIPPER [17] routers, and discuss the deflection arbitration, livelock freedom, and packet reassembly problems associated with bufferless routing.
- A new NoC router, *MinBD* (minimally-buffered deflection routing) [20], that combines deflection routing with minimal buffering. The router performs deflection routing, but can choose to buffer up to one flit per cycle in a small side buffer, which significantly reduces deflection rate and enhances performance compared to a pure bufferless design while requiring smaller buffer space than a conventional input-buffered design.
- An evaluation of MinBD against aggressive NoC router baselines: a two-cycle virtual channel buffered router [11] with empty buffer bypassing [37, 50] at three buffer capacities (with a sensitivity analysis over many more configurations), CHIPPER [17], and a hybrid bufferless-buffered design, AFC [29], with SPEC CPU2006 [45] multiprogrammed workloads on 16- and 64-node CMP systems. From our results, we conclude that MinBD has the best energy efficiency over all of these prior design points, while achieving competitive system throughput and logic critical path delay with the input-buffered router (the best-performing baseline) and competitive area and power consumption with the pure-bufferless router (the smallest and most power-efficient baseline).

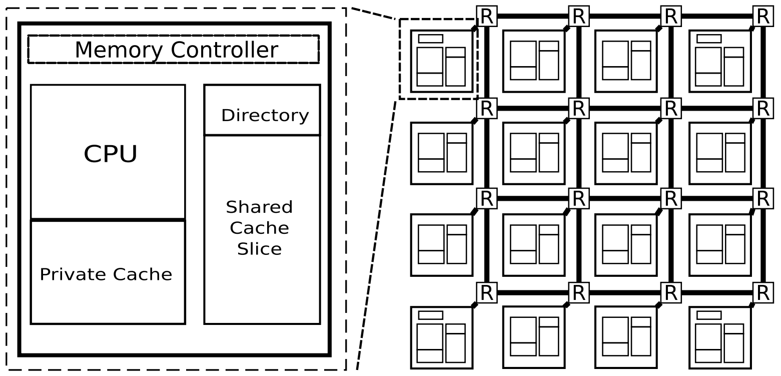

Fig. 1.1 An example Network-on-Chip (NoC)-based system: an on-chip packet-switched network connects nodes which often consist of cores, cache slices, and memory controllers.

1.2 Background

This section provides background on NoC-based cache-coherent CMPs, and on bufferless deflection routing. We assume the reader is familiar with the basic operation of conventional input-buffered routers. The key idea of such routers is to buffer every flit that enters the router from an input port before the flits can arbitrate for output ports. Dally and Towles [11] provide a good reference on these routers.

NoCs in cache-coherent CMPs: On-chip networks form the backbone of memory systems in many recently-proposed and prototyped large-scale CMPs (chip multiprocessors) [46, 28, 51]. Most such systems are cache-coherent shared memory multiprocessors. Packet-switched interconnect has served as the substrate for large cache-coherent systems for some time (e.g., for large multiprocessor systems such as SGI Origin [34]), and the principles are the same in a chip multiprocessor: each core, slice of a shared cache, or memory controller is part of one "node" in the network, and network nodes exchange packets that request and respond with data in order to fulfill memory accesses. A diagram of a typical system is shown in Fig. 1.1. For example, on a miss, a core's private cache might send a request packet to a shared L2 cache slice, and the shared cache might respond with a larger packet containing the requested cache block on an L2 hit, or might send another packet to a memory controller on an L2 miss. CMP NoCs are typically used to implement such a protocol between the cores, caches and memory controllers.

1.2.1 Bufferless Deflection Routing in NoCs: BLESS

Bufferless Deflection Routers: Bufferless deflection routing was first proposed by Baran [3]. Bufferless deflection routing operates without in-network buffering. Instead, a unit of traffic continuously moves between network nodes until it reaches its destination. When contention occurs for a network link, a bufferless deflection router sends some traffic to another output link instead, *deflecting* it. Hence, the use of buffers is replaced by occasional extra link traversals.

Bufferless deflection routing has found renewed interest in NoC design because on-chip wires (hence, network links) are relatively cheap, in contrast to buffers, which consume significant die area and leakage power [4, 5, 7, 29, 38]. Several evaluations of bufferless NoC design [17, 26, 29, 38] have demonstrated that removing the buffers in NoC routers, and implementing a routing strategy which operates without the need for buffers (such as the one we describe below), yield energy-efficiency improvements because occasional extra link traversals due to deflections consume relatively less energy than the dynamic energy used to buffer traffic at every network hop and the static energy consumed whenever a buffer is turned on. (Our motivational experiments in §1.3 demonstrate the performance and energy impact of such a network design in more detail.) Although other solutions exist to reduce the energy consumption of buffers, such as dynamic buffer bypassing [37, 50] (which we also incorporate into our baseline buffered-router design in this chapter), bufferless deflection routing achieves additional savings in energy and area by completely eliminating the buffers.

One recent work proposed BLESS [38], a router design that implements bufferless deflection routing, which we describe here. The fundamental unit of routing in a BLESS network is the *flit*, a packet fragment transferred by one link in one cycle. Flits are routed independently in BLESS.[3] Because flits are routed independently, they must be reassembled after they are received. BLESS assumes the existence of sufficiently-sized reassembly buffers at each node in order to reconstruct arriving flits into packets. (Later work, CHIPPER [17], addresses the reassembly problem explicitly, as we discuss below.)

Deflection Routing Arbitration: The basic operation of a BLESS bufferless deflection router is simple. In each cycle, flits arriving from neighbor routers enter the router pipeline. Because the router contains no buffers, flits are stored only in pipeline registers, and must leave the router at the end of the pipeline. Thus, the router must assign every input flit to some output port. When two flits request the same output port according to their

[3] BLESS' independent flit routing stands in contrast to conventional wormhole or VC (virtual-channel) routing, in which a packet's body flits always follow its head flits: because a deflection can occur in any cycle, any flit in a BLESS network could be separated from the rest of its packet and must carry its own routing information. This is described more fully in Moscibroda and Mutlu [38].

ordinary routing function, the router deflects one of them to another port (this is always possible, as long as the router has as many outputs as inputs). BLESS performs this router output port assignment in two stages: flit ranking and port selection [38]. In each cycle, the flits that arrive at the router are first *ranked* in a priority order (chosen in order to ensure livelock-free operation, as we describe below). At the same time, the router computes a list of productive output ports (i.e., ports which would send the flit closer to its destination) for each flit. Once the flit ranking and each flits' productive output ports are available, the router assigns a port to each flit, starting from the highest-ranked flit and assigning ports to flits one at a time. Each flit obtains a productive output port if one is still available, and is "deflected" to any available output port otherwise. Because there are as many output ports as input ports, and only the flits arriving on the input ports in a given cycle are considered, this process never runs out of output ports and can always assign each flit to some output. Hence, no buffering is needed, because every flit is able to leave the router at the end of the router pipeline.

Livelock freedom in BLESS: Although a BLESS router ensures that a flit is always able to take a network hop to *some* other router, a deflection takes a flit further from its destination, and such a flit will have to work its way eventually to its destination. In such a network design, explicit care must be taken to ensure that all flits eventually arrive at their destinations (i.e., that no flit circles, or *gets stuck*, in the network forever). This property is called *livelock freedom*. Note that conventional virtual channel-buffered routers, which buffer flits at every network hop, are livelock-free simply because they never deflect flits: rather, whenever a flit leaves a router and traverses a link, it always moves closer toward its destination (this is known as *minimal routing* [11]).

BLESS ensures livelock freedom by employing a priority scheme called *Oldest-First* [38]. Oldest-First prioritization is a total order over all flits based on each flit's age (time it has spent in the network). If two flits have the same age (entered the network in the same cycle), then the tie is broken with other header fields (such as sender ID) which uniquely identify the flit. This total priority order leads to livelock-free operation in a simple way: there must be one flit which is the oldest, and thus has the highest priority. This flit is always be prioritized during flit-ranking at every router it visits. Thus, it obtains its first choice of output port and is never deflected. Because it is never deflected, the flit always moves closer toward its destination, and will eventually arrive. Once it arrives, it is no longer contending with other flits in the network, and some other flit is the oldest flit. The new oldest flit is guaranteed to arrive likewise. Inductively, all flits eventually arrive.

Flit injection and ejection: A BLESS router must inject new flits into the network when a node generates a packet, and it must remove a flit from the network when the flit arrives at its destination. A BLESS router makes a *local decision* to inject a flit whenever, in a given cycle, there is an empty slot on any of its input ports [38]. The router has an injection queue where flits wait until this injection condition is met. When a node is not able to inject, it is *starved*; injection starvation is a useful proxy for network congestion which has been used to control congestion-control mechanisms in bufferless deflection networks [8, 39, 40].

When a flit arrives at its destination router, that router removes the flit from the network and places it in a *reassembly buffer*, where it waits for the other flits from its packet to arrive. Flits in a packet may arrive in any order because each flit is routed independently, and might take a different path than the others due to deflections. Once all flits in a packet have arrived in that packet's reassembly buffer, the packet is delivered to the local node (e.g., core, cache, or memory controller). A BLESS router can eject up to one flit per cycle from its inputs to its reassembly buffer. Fig. 1.2 depicts the reassembly buffers as well as the injection queue of a node in a BLESS NoC.

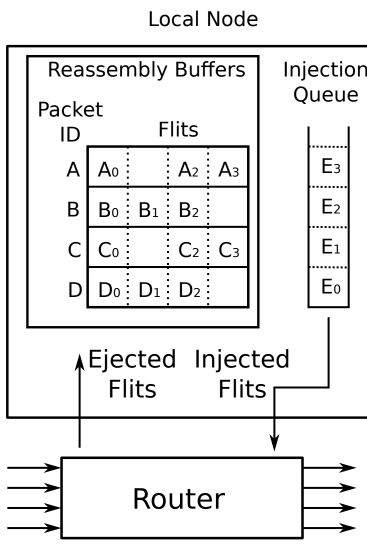

Fig. 1.2 Reassembly buffers and injection queue in a BLESS NoC.

1.2.2 Low-complexity Bufferless Deflection Routing: CHIPPER

CHIPPER [17], another bufferless deflection router design, was proposed to address implementation complexities in prior bufferless deflection routers (e.g., BLESS). The CHIPPER router has smaller and simpler deflection-routing logic than BLESS, which leads to a shorter critical path, smaller die area and lower power.

1.2.2.1 Problems in BLESS

The Oldest-First age-based arbitration in BLESS leads to slow routers with large hardware footprint [17, 26, 37] for several reasons, which we describe here.

Deflection arbitration: First, implementing deflection arbitration in the way that BLESS specifies leads to complex hardware. Routers that use Oldest-First arbitration must sort input flits by priority (i.e., age) in every cycle. This requires a three-stage sorting network for four inputs. Then, the router must perform port assignment in priority order, giving higher-priority flits first choice. Because a lower-priority flit might be deflected if a higher priority-flit takes an output port first, flits must be assigned output ports sequentially. This sequential port allocation leads to a *long critical path*, hindering practical implementation. This critical path (through priority sort and sequential port allocation) is illustrated in Fig. 1.3.

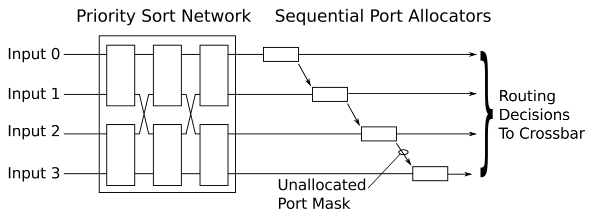

Fig. 1.3 In BLESS, a priority sort network and sequential port allocation are necessary to implement livelock-free routing, yielding a long critical path.

Packet reassembly: Second, as noted above, BLESS makes use of reassembly buffers to reassemble flits into packets. Reassembly buffers are necessary because each flit is routed independently and may take a different path than the others in a packet, arriving at a different time. Moscibroda and Mutlu [38] evaluate bufferless deflection routing assuming an infinite reassembly buffer, and report maximum buffer occupancy.

However, with a finite reassembly buffer that is smaller than a certain size, deadlock will occur in the worst case (when all nodes send a packet simultaneously to a single node). To see why this is the case, observe the example in Fig. 1.4 (figure taken from Fallin et al. [17]). When a flit arrives at the reassembly buffers in Node 0, the packet reassembly logic checks whether a reassembly slot has already been allocated to the packet to which this flit belongs. If not, a new slot is allocated, if available. If the packet already has a slot, the flit is placed into its proper location within the packet-sized buffer. When no slots are available and a flit from a new packet arrives, the reassembly logic must prevent the flit from being ejected out of the network. In the worst case, portions of many separate packets arrive at Node 0, allocating all its slots. Then, flits from other packets arrive, but cannot be ejected, because

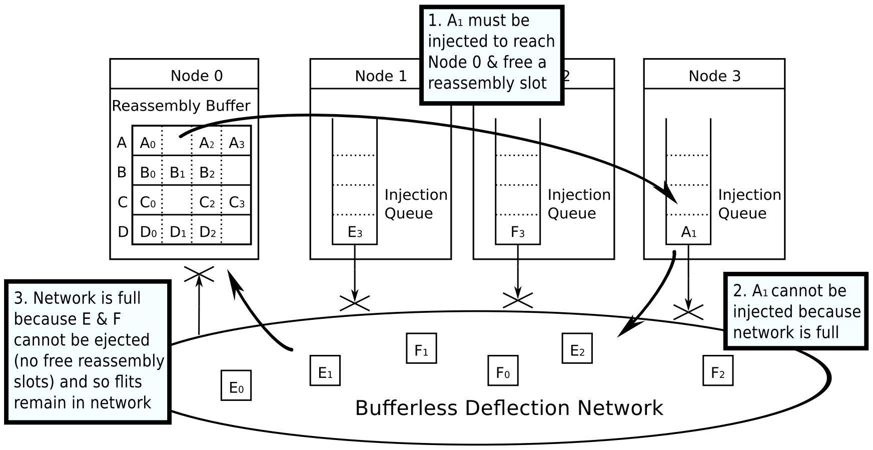
1. A₁ must be injected to reach Node 0 & free a reassembly slot
Node 0
Reassembly Buffer
A A₀ A₂ A₃
B B₀ B₁ B₂
C C₀ C₂ C₃
D D₀ D₁ D₂
Node 1
Node 3
Injection Queue
E₃
F₃
A₁
2. A₁ cannot be injected because network is full
3. Network is full because E & F cannot be ejected (no free reassembly slots) and so flits remain in network
E₀ E₁ F₁ F₀ E₂ F₂
Bufferless Deflection Network

Fig. 1.4 Deadlock due to reassembly-buffer overflow in bufferless routing.

no reassembly slots are free. These flits remain in the network, deflecting and retrying ejection. Eventually, the network will fill with these flits. The flits which are required to complete the partially-reassembled packets may have not yet been injected at their respective sources, and they cannot be injected, because the network is completely full. Thus, deadlock occurs. Without a different buffer management scheme, the only way to avoid this deadlock is to size the reassembly buffer at each node for the worst case when all other nodes in the system send a packet to that node simultaneously. A bufferless deflection router implementation with this amount of buffering would have significant overhead, unnecessarily wasting area and power. Hence, an explicit solution is needed to ensure deadlock-free packet reassembly in practical designs.

1.2.2.2 CHIPPER: A Low-complexity Bufferless Deflection Router

We now outline the operation of CHIPPER [17], a bufferless deflection router design which makes bufferless deflection routing practical by providing for low-cost deflection arbitration and packet reassembly.

Golden Packet-based deflection arbitration: A bufferless deflection router must ensure that the network has *livelock freedom* by providing a strong guarantee that any flit eventually arrives at its destination. BLESS ensured that any flit arrives by enforcing a *total priority order* among all flits, such that the highest-priority (oldest) flit is delivered, then another flit attains the highest priority (becomes the oldest). However, enforcing a total priority order creates significant complexity in a BLESS router (as we described above).

The CHIPPER router design starts from the observation that *minimal* livelock-free routing requires only that one flit is prioritized until it arrives, and that any flit is eventually chosen to be this specially-prioritized flit if it remains in the network long enough. This priority scheme is called *Golden Packet*, and it allows the CHIPPER router to use a simpler design than the BLESS router.

The Golden Packet priority scheme globally prioritizes one *packet* in the network at a time. Flits in this packet (which we call *golden flits*) are prioritized over other flits in the network. (Within the packet, ties are broken by the flit sequence number within the packet.) The prioritization rules are shown in Ruleset 1. When a packet becomes the Golden Packet, it remains so for a *golden epoch*, which is set to a length L that is long enough so that the packet can reach any destination in the network from any source.

The Golden Packet is chosen *implicitly* (i.e., without the need for all routers to explicitly coordinate their choice). The CHIPPER network can uniquely name any packet with a *packet ID* (e.g., source ID and cache-miss MSHR number, or some other transaction identifier). One *packet ID* is designed as golden based on a predefined function of the current time (in clock cycles).[4] In particular, the packet ID which is currently golden is incremented once every L cycles (the golden epoch), and wraps around when all possible packet IDs have each been designated as golden for the length of a golden epoch. In this way, any packet eventually becomes golden if it remains in the network long enough.

Ruleset 1 Golden Packet Prioritization Rules

Golden Tie: If two flits are golden, the lower-numbered flit (first in golden packet) wins.
Golden Dominance: If one flit is golden, it wins over any non-golden flit.
Common Case: Contests between two non-golden flits are decided pseudo-randomly.

The most important consequence of Golden Packet is that each router *only needs to correctly route the highest-priority flit.* This is sufficient to ensure that the first outstanding flit of the Golden Packet is delivered within L cycles. Because the packet will periodically become Golden until delivered, all of its flits are guaranteed delivery.

Because Golden Packet prioritization provides livelock freedom as long as the highest-priority flit is correctly routed, the deflection routing (arbitration) logic does not need to sequentially assign each flit to the best possible port, as the BLESS router's deflection routing logic does (Fig. 1.3). Rather, it only needs to recognize a golden flit, if one is present at the router inputs, and route that flit correctly if present. All other deflection arbitration is best-effort. Arbitration can thus be performed more quickly with simpler logic.

[4] CHIPPER assumes that all routers are in a single clock domain, hence can maintain synchronized golden packet IDs simply by counting clock ticks.

We now describe the CHIPPER router's arbitration logic here; the router's pipeline is depicted in Fig. 1.5 (see Fallin et al. [17] for more details, including the ejection/injection logic which is not described here). The CHIPPER router's arbitration logic is built with a basic unit, the *two-input arbiter block*, shown on the right side of Fig. 1.5. Each two-input arbiter block receives up to two flits every cycle and routes these two flits to its outputs. In order to route its input flits, the two-input arbiter block chooses one *winning* flit. If a golden flit is present, the golden flit is the winning flit (if two golden flits are present, the tie is broken as described by the prioritization rules). If no golden flit is present, one of the input flits is chosen randomly to be the winning flit. The two-input arbiter block then examines the winning flit's destination, and sends this flit toward the arbiter block's output which leads that flit closer to its destination. The other flit, if present, must then take the remaining arbiter block output.

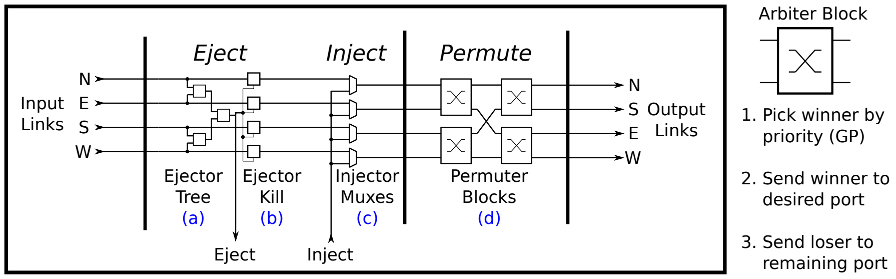

Fig. 1.5 CHIPPER router microarchitecture: router pipeline (left) and detail of a single arbiter block (right).

The CHIPPER router performs deflection arbitration among 4 input flits (from the four inputs in a 2D mesh router) using a *permutation network* of four arbiter blocks, connected in two stages of two blocks each, as shown in the *permute* pipeline stage of Fig. 1.5. The permutation network allows a flit from any router input to reach any router output. When flits arrive, they arbitrate in the first stage, and winning flits are sent toward the second-stage arbiter block which is connected to that flit's requested router output. Then, in the second stage, flits arbitrate again. As flits leave the second stage, they proceed directly to the router outputs via a pipeline register (no crossbar is necessary, unlike in conventional router designs). This two-stage arbitration has a shorter critical path than the sequential scheme used by a BLESS router because the arbiter blocks in each stage work in parallel, and because (unlike in a BLESS arbiter) the flits need *not* be sorted by priority first. The arbiter-block permutation network cannot perform *all* possible flit permutations (unlike the BLESS router's routing logic), but because a golden flit (if present) is always prioritized, and hence always sent to a router output which carries the flit closer to its destination, the network is still livelock-free. Because the permutation network (i) eliminates priority sorting, and (ii) partially parallelizes port assignment, the router critical path is improved (reduced) by 29.1%, performing within 1.1% of a conventional buffered router design [17].

Addressing packet reassembly deadlock with Retransmit-Once: Fallin et al. [17] observe that the reassembly deadlock problem is fundamentally due to a lack of global flow control. Unlike buffered networks, which can pass tokens upstream to senders to indicate whether downstream buffer space is available, a bufferless deflection network has no such backpressure. Allowing receivers to exert backpressure on senders solves the problem. Thus, CHIPPER introduces a new low-overhead flow control protocol, *Retransmit-Once*, as its second major contribution.

Retransmit-Once *opportunistically* assumes that buffer space will be available, imposing no network overhead in the common case. When no space is available, any subsequent arriving packet is dropped at the receiver. However, the receiver makes note of this dropped packet. Once reassembly buffer space becomes available, the reassembly logic in the receiver *reserves* buffer space for the previously dropped packet, and the receiver then requests a retransmission from the sender. Thus, at most one retransmission is necessary for any packet. In addition, by dropping only short request packets (which can be regenerated from a sender's request state), and using reservations to ensure that longer data packets are never dropped, Retransmit-Once ensures that senders do not have to buffer data for retransmission. In our evaluations of realistic workloads, retransmission rate is 0.021% maximum with 16-packet reassembly buffers, hence the performance impact is negligible. Fallin et al. [17] describe the Retransmit-Once mechanism in more detail and report that it can be implemented with very little overhead by integrating with cache-miss buffers (MSHRs) in each node.

1.3 Motivation: Performance at High Load

Previous NoC designs based on bufferless deflection routing, such as BLESS [38] and CHIPPER [17] which we just introduced, were motivated largely by the observation that many NoCs in CMPs are over-provisioned for the common-case network load. In this case, a bufferless network can attain nearly the same application performance while consuming less power, which yields higher energy efficiency. We now examine the buffered-bufferless comparison in more detail. Fig. 1.6 shows (i) relative application performance (weighted speedup: see §1.5), and (ii) relative energy efficiency (performance per watt), when using a bufferless network, compared to a conventional buffered network. Both plots show these effects as a function of network load (average injection rate). Here we show a virtual channel buffered network (4 VCs, 4 flits/VC) (with

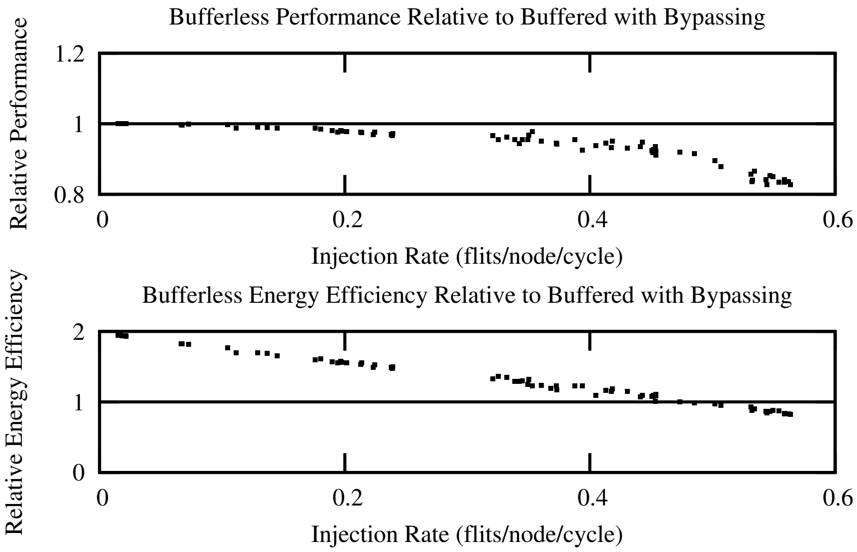

Fig. 1.6 System performance and energy efficiency (performance per watt) of bufferless deflection routing, relative to conventional input-buffered routing (4 VCs, 4 flits/VC) that employs buffer bypassing, in a 4x4 2D mesh. Injection rate (X axis) for each workload is measured in the baseline buffered network.

buffer bypassing) and the CHIPPER bufferless deflection network [17] in a 4x4-mesh CMP (details on methodology are in §1.5).

For low-to-medium network load, a bufferless network has performance close to a conventional buffered network, because the deflection rate is low: thus, most flits take productive network hops on every cycle, just as in the buffered network. In addition, the bufferless router has significantly reduced power (hence improved energy efficiency), because the buffers in a conventional router consume significant power. However, as network load increases, the deflection rate in a bufferless deflection network also rises, because flits contend with each other more frequently. With a higher deflection rate, the dynamic power of a bufferless deflection network rises more quickly with load than dynamic power in an equivalent buffered network, because each deflection incurs some extra work. Hence, bufferless deflection networks lose their energy-efficiency advantage at high load. Just as important, the high deflection rate causes each flit to take a longer path to its destination, and this increased latency reduces the network throughput and system performance.

Overall, neither design obtains both good performance and good energy efficiency at all loads. If the system usually experiences low-to-medium network load, then the bufferless design provides adequate performance with low power (hence high energy efficiency). But, if we use a conventional buffered design to obtain high performance, then energy efficiency is poor in the low-load case, and even buffer bypassing does not remove this overhead because buffers consume static power regardless of use. Finally, simply switching between these two extremes at a per-router granularity, as previously proposed [29], does not address the fundamental inefficiencies in the bufferless routing mode, but rather, uses input buffers for all incoming flits at a router when load is too high for the bufferless mode (hence retains the relative energy-inefficiency of buffered operation at high load). We now introduce MinBD, the *minimally-buffered deflection router*, which combines bufferless and buffered routing to reduce this overhead.

1.4 MinBD: Minimally-Buffered Deflection Router

The *MinBD* (minimally-buffered deflection) router is a new router design that combines bufferless deflection routing with a small buffer, called the "side buffer." We start by outlining the **key principles** which the design follows to reduce deflection-caused inefficiency by using buffering:

1. When a flit would be deflected by a router, it is often better to buffer the flit and arbitrate again in a later cycle. Some buffering can avoid many deflections.
2. However, buffering every flit leads to unnecessary power overhead and buffer requirements, because many flits will be routed productively on the first try. The router should buffer a flit only if necessary.
3. Finally, when a flit arrives at its destination, it should be removed from the network (ejected) quickly, so that it does not continue to contend with other flits.

Basic High-Level Operation: The MinBD router does not use input buffers, unlike conventional buffered routers. Instead, a flit that arrives at the router proceeds directly to the routing and arbitration logic. This logic performs deflection routing, so that when two flits contend for an output port, one of the flits is sent to another output instead. However, unlike a bufferless deflection router, the MinBD router can also *buffer* up to one flit per cycle in a single FIFO-queue side buffer. The router examines all flits at the output of the deflection routing logic, and if any are deflected, one of the deflected flits is removed from the router pipeline and buffered (as long as the buffer is not full). From the side buffer, flits are re-injected into the network by the router, in the same way that new traffic is injected. Thus, some flits that would have been deflected in a bufferless deflection router are removed from the network temporarily into this side buffer, and given a second chance to arbitrate for a productive router output when re-injected. This reduces the network's deflection rate (hence improves performance and energy efficiency) while buffering only a fraction of traffic.

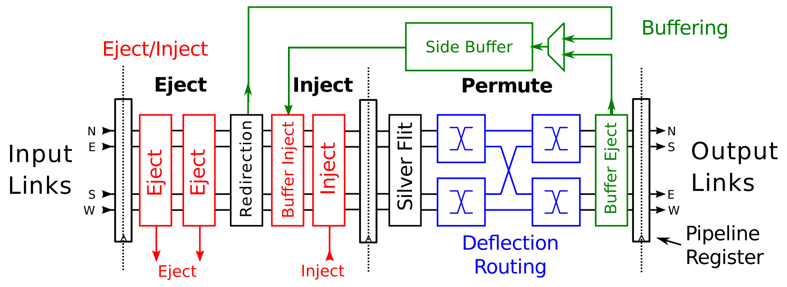

Fig. 1.7 MinBD router pipeline.

Ruleset 2 MinBD Prioritization Rules (based on Golden Packet [17] with new rule 3)

Given: two flits, each *Golden*, *Silver*, or *Ordinary*. (Only one can be Silver.)
1. **Golden Tie**: Ties between two Golden flits are resolved by sequence number (first in Golden Packet wins).
2. **Golden Dominance**: If one flit is Golden, it wins over any Silver or Ordinary flits.
3. **Silver Dominance**: Silver flits win over Ordinary flits.
4. **Common Case**: Ties between Ordinary flits are resolved randomly.

We will describe the operation of the MinBD router in stages. First, §1.4.1 describes the deflection routing logic that computes an initial routing decision for the flits that arrive in every cycle. Then, §1.4.2 describes how the router chooses to buffer some (but not all) flits in the side buffer. §1.4.3 describes how buffered flits and newly-generated flits are injected into the network, and how a flit that arrives at its destination is ejected. Finally, §1.4.4 discusses correctness issues, and describes how MinBD ensures that all flits are eventually delivered.

1.4.1 Deflection Routing

The MinBD router pipeline is shown in Fig. 1.7. Flits travel through the pipeline from the inputs (on the left) to outputs (on the right). We first discuss the deflection routing logic, located in the Permute stage on the right. This logic implements deflection routing: it sends each input flit to its preferred output when possible, deflecting to another output otherwise.

MinBD uses the deflection logic organization first proposed in CHIPPER [17]. The *permutation network* in the Permute stage consists of two-input blocks arranged into two stages of two blocks each. This arrangement can send a flit on any input to any output. (Note that it cannot perform all possible permutations of inputs to outputs, but as we will see, it is sufficient for correct operation that at least one flit obtains its preferred output.) In each two-input block, arbitration logic determines which flit has a higher priority, and sends that flit in the direction of its preferred output. The other flit at the two-input block, if any, must take the block's other output. By combining two stages of this arbitration and routing, deflection arises as a distributed decision: a flit might be deflected in the first stage, or the second stage. Restricting the arbitration and routing to two-flit subproblems reduces complexity and allows for a shorter critical path, as demonstrated in [17].

In order to ensure correct operation, the router must arbitrate between flits so that every flit is eventually delivered, despite deflections. MinBD adapts a modified version of the Golden Packet priority scheme [17], which solves this *livelock-freedom problem.* This priority scheme is summarized in Ruleset 2. The basic idea of the Golden Packet priority scheme is that at any given time, at most one packet in the system is *golden.* The flits of this golden packet, called "golden flits," are prioritized above all other flits in the system (and contention between golden flits is resolved by the flit sequence number). While prioritized, golden flits are never deflected by non-golden flits. The packet is prioritized for a period long enough to guarantee its delivery. Finally, this "golden" status is assigned to one globally-unique packet ID (e.g., source node address concatenated with a request ID), and this assignment rotates through all possible packet IDs such that any packet that is "stuck" will eventually become golden. In this way, all packets will eventually be delivered, and the network is livelock-free. (See [17] for the precise way in which the Golden Packet is determined; MinBD uses the same rotation schedule.)

However, although Golden Packet arbitration provides correctness, a *performance* issue occurs with this priority scheme. Consider that most flits are not golden: the elevated priority status provides worst-case correctness, but does not impact common-case performance (prior work reported over 99% of flits are delivered without becoming golden [17]). However, when no flits are golden and ties are broken randomly, the arbitration decisions in the two permutation network stages are not coordinated. Hence, a flit might win arbitration in the first stage, and cause another flit to be deflected, but then lose arbitration in the second stage, and also be deflected. Thus, unnecessary deflections occur when the two permutation network stages are uncoordinated.

In order to resolve this performance issue, we observe that it is enough to ensure that in every router, at least one flit is prioritized above the others in every cycle. In this way, at least one flit will certainly not be deflected. To ensure this when no golden flits are present, MinBD adds a "silver" priority level, which wins arbitration over common-case flits but loses to the golden flits. One silver flit is designated randomly among the set of flits that enter a router at every cycle (this designation is local to the router, and not propagated to other routers). This modification helps to reduce deflection rate. Prioritizing a silver flit at every router does not impact correctness, because it does not deflect a golden flit if one is present, but it ensures that at least one flit will consistently win arbitration at both stages. Hence, deflection rate is reduced, improving performance.

1.4.2 Using a Small Buffer to Reduce Deflections

The key problem addressed by MinBD is *deflection inefficiency at high load*: in other words, when the network is highly utilized, contention between flits occurs often, and many flits will be deflected. We observe that adding a small buffer to a deflection router can reduce deflection rate, because the router can choose to buffer rather than deflect a flit when its output port is taken by another flit. Then, at a later time when output ports may be available, the buffered flit can re-try arbitration.

Thus, to reduce deflection rate, MinBD adds a "side buffer" that buffers only some flits that otherwise would be deflected. This buffer is shown in Fig. 1.7 above the permutation network. In order to make use of this buffer, a "buffer ejection" block is placed in the pipeline after the permutation network. At this point, the arbitration and routing logic has determined which flits to deflect. The buffer ejection block recognizes flits that have been deflected, and picks up to one such deflected flit per cycle. It removes a deflected flit from the router pipeline, and places this flit in the side buffer, as long as the side buffer is not full. (If the side buffer is full, no flits are removed from the pipeline into the buffer until space is created.) This flit is chosen randomly among deflected flits (except that a golden flit is never chosen: see §1.4.4). In this way, some deflections are avoided. The flits placed in the buffer will later be re-injected into the pipeline, and will re-try arbitration at that time. This re-injection occurs in the same way that new traffic is injected into the network, which we discuss below.

1.4.3 Injection and Ejection

So far, we have considered the flow of flits from router input ports (i.e., arriving from neighbor routers) to router output ports (i.e., to other neighbor routers). A flit must enter and leave the network at some point. To allow traffic to enter (be injected) and leave (be ejected), the MinBD router contains injection and ejection blocks in its first pipeline stage (see Fig. 1.7). When a set of flits arrives on router inputs, these flits first pass through the ejection logic. This logic examines the destination of each flit, and if a flit is addressed to the local router, it is removed from the router pipeline and sent to the local network node.[5] If more than one locally-addressed flit is present, the ejection block picks one, according to the same priority scheme used by routing arbitration.

However, ejecting a single flit per cycle can produce a bottleneck and cause unnecessary deflections for flits that could not be ejected. In the workloads

[5] Note that flits are reassembled into packets after ejection. To implement this reassembly, we use the Retransmit-Once scheme, as used by CHIPPER and described in §1.2.2.2.

we evaluate, at least one flit is eligible to be ejected 42.8% of the time. Of those cycles, 20.4% of the time, at least two flits are eligible to be ejected. Hence, in ∼8.5% of all cycles, a locally-addressed flit would be deflected rather than ejected if only one flit could be ejected per cycle. To avoid this significant deflection-rate penalty, MinBD doubles the ejection bandwidth. To implement this, a MinBD router contains two ejection blocks. Each of these blocks is identical, and can eject up to one flit per cycle. Duplicating the ejection logic allows two flits to leave the network per cycle at every node.[6]

After locally-addressed flits are removed from the pipeline, new flits are allowed to enter. There are two injection blocks in the router pipeline shown in Fig. 1.7: (i) re-injection of flits from the side buffer, and (ii) injection of new flits from the local node. (The "Redirection" block prior to the injection blocks will be discussed in the next section.) Each block operates in the same way. A flit can be injected into the router pipeline whenever one of the four inputs does not have a flit present in a given cycle, i.e., whenever there is an "empty slot" in the network. Each injection block pulls up to one flit per cycle from an injection queue (the side buffer, or the local node's injection queue), and places a new flit in the pipeline when a slot is available. Flits from the side buffer are re-injected before new traffic is injected into the network. However, note that there is no guarantee that a free slot will be available for an injection in any given cycle. We now address this starvation problem for side buffer re-injection.

1.4.4 Ensuring Side Buffered Flits Make Progress

When a flit enters the side buffer, it leaves the router pipeline, and must later be re-injected. As we described above, flit re-injection must wait for an empty slot on an input link. It is possible that such a slot will not appear for a long time. In this case, the flits in the side buffer are delayed unfairly while other flits make forward progress.

To avoid this situation, MinBD implements *buffer redirection.* The key idea of buffer redirection is that when this side buffer starvation problem is detected, one flit from a randomly-chosen router input is *forced* to enter the side buffer. Simultaneously, the flit at the head of the side buffer is injected into the slot created by the forced flit buffering. In other words, one router input is "redirected" into the FIFO buffer for one cycle, in order to allow the buffer to make forward progress. This redirection is enabled for one cycle whenever the side buffer injection is starved (i.e., has a flit to inject, but

[6] For fairness, because dual ejection widens the datapath from the router to the local node (core or cache), we also add dual ejection to the baseline bufferless deflection network and input-buffered network when we evaluate performance, but not when we evaluate the power, area, or critical path of these baselines.

no free slot allows the injection) for more than some threshold $C_{threshold}$ cycles (in our evaluations, $C_{threshold} = 2$). Finally, note that if a golden flit is present, it is never redirected to the buffer, because this would break the delivery guarantee.

1.4.5 Livelock and Deadlock-free Operation

MinBD provides livelock-free delivery of flits using Golden Packet and buffer redirection. If no flit is ever buffered, then Golden Packet [17] ensures livelock freedom (the "silver flit" priority never deflects any golden flit, hence does not break the guarantee). Now, we argue that adding side buffers does not cause livelock. First, the buffering logic never places a golden flit in the side buffer. However, a flit could enter a buffer and then become golden while waiting. Redirection ensures correctness in this case: it provides an upper bound on residence time in a buffer (because the flit at the head of the buffer will leave after a certain threshold time in the worst case). If a flit in a buffer becomes golden, it only needs to remain golden long enough to leave the buffer in the worst case, then progress to its destination. MinBD chooses the threshold parameter ($C_{threshold}$) and golden epoch length so that this is always possible. More details can be found in our extended technical report [18].

MinBD achieves deadlock-free operation by using Retransmit-Once [17], which ensures that every node always consumes flits delivered to it by dropping flits when no reassembly/request buffer is available. This avoids packet-reassembly deadlock (as described in [17]), as well as protocol level deadlock, because message-class dependencies [25] no longer exist.

1.5 Evaluation Methodology

To obtain application-level performance as well as network performance results, we use an in-house CMP simulator. This simulator consumes instruction traces of x86 applications, and faithfully models the CPU cores and the cache hierarchy, with a directory-based cache coherence protocol (based on the SGI Origin protocol [9]) running on the modeled NoC. The CPU cores model stalls, and interact with the caches and network in a closed-loop way. The modeled network routers are cycle-accurate, and are assumed to run in a common clock domain. The instruction traces are recorded with a Pintool [36], sampling representative portions of each benchmark as determined by PinPoints [41]. We find that simulating 25M cycles gives stable results with these traces. Detailed system parameters are shown in Table 1.1.

Note that we make use of a *perfect shared cache* to stress the network, as was done in the CHIPPER [17] and BLESS [38] bufferless router evalu-

Table 1.1 Simulated baseline system parameters.

CPU cores	Out-of-order, 3-wide issue and retire (1 memory op/cycle), 16 MSHRs [33]
L1 caches	64 KB, 4-way associative, 32-byte blocks
L2 (shared) cache	Distributed across nodes; perfect (always hits) to penalize our design conservatively & isolate network performance from memory effects
Shared cache mapping	Consecutive cache blocks striped across L2 cache slices
Cache coherence scheme	Directory-based, perfect directory (SGI Origin protocol [34])
Data packet sizes	1-flit request packets, 4-flit data packets
Network Links	1-cycle latency (separate pipeline stage), 2.5mm, 128 bits wide
Baseline bufferless router	CHIPPER [17], 2-cycle router latency; 64-cycle Golden Epoch; Retransmit-Once [17]
Baseline buffered router	(m VCs, n flits/VC): $(8, 8)$, $(4, 4)$, $(4, 1)$; 2-cycle latency, buffer bypassing [37, 50]; Additional configurations evaluated in Fig. 1.9.
AFC (Adaptive Flow Control)	As described in [29]: 4 VCs/channel, 4 flits/VC. 2-cycle latency (buffered & bufferless modes). Implements buffer bypassing as well.
MinBD	2-cycle latency (§1.4); 4-flit side buffer (single FIFO); $C_{threshold} = 2$; 64-cycle Golden Epoch; Retransmit-Once [17]

ations. In this model, every request generated by an L1 cache miss goes to a shared cache slice, and the request is always assumed to hit and return data. This potentially increases network load relative to a real system, where off-chip memory bandwidth can also be a bottleneck. However, note that this methodology is *conservative*: because MinBD degrades performance relative to the buffered baseline, the performance degradation that we report is an upper bound on what would occur when other bottlenecks are considered. We choose to perform our evaluations this way in order to study the true capacity of the evaluated networks (if network load is always low because system bottlenecks such as memory latency are modeled, then the results do not give many insights about router design). Note that the cache hierarchy details (L1 and L2 access latencies, and MSHRs) are still realistic. We remove only the off-chip memory latency/bandwidth bottleneck.

Baseline Routers: We compare MinBD to a conventional input-buffered virtual channel router [11] with buffer bypassing [37, 50], a bufferless deflection router (CHIPPER [17]), and a hybrid bufferless-buffered router (AFC [29]). In particular, we sweep buffer size for input-buffered routers. We describe a router with m virtual channels (VCs) per input and n flits of buffer capacity per VC as an (m, n)-buffered router. We compare to a $(8, 8)$, $(4, 4)$, and $(4, 1)$-buffered routers in our main results. The $(8, 8)$ point represents a very large (overprovisioned) baseline, while $(4, 4)$ is a more reasonable general-purpose configuration. The $(4, 1)$ point represents the *minimum* buffer size for deadlock-free operation (two message classes [25], times two to avoid routing deadlock [10]). Furthermore, though 1-flit VC buffers would reduce throughput because they do not cover the credit round-trip latency, we optimistically assume zero-latency credit traversal in our simulations (which benefits the baseline design, hence is conservative for our claims). Finally, we simulate smaller (non-deadlock-free) designs with 1 and 2 VCs per input for our power-performance tradeoff evaluation in §1.6.2 (Fig. 1.9), solely for completeness (we were able to avoid deadlock at moderate loads and with finite simulation length for these runs).

Application Workloads: We use SPEC CPU2006 [45] benchmarks (26 applications[7]) to construct 75 multiprogrammed workloads (which consist of many single-threaded benchmark instances that run independently). Each workload consists of 16 or 64 randomly-selected applications which are randomly mapped onto the mesh. Workloads are categorized by average network injection rate in the baseline $(4,4)$-buffered system (measured in flits/cycle/node). For 4x4 workloads, these injection rate categories are $(0, 0.15)$, $(0.15, 0.3)$, $(0.3, 0.4)$, $(0.4, 0.5)$, and > 0.5; for 8x8 workloads, $(0, 0.05)$, $(0.05, 0.15)$, $(0.15, 0.25)$, and > 0.25 (an 8x8 mesh saturates at a lower load than a 4x4 mesh, due to limited bisection bandwidth). Each category contains 15 workloads.

Synthetic-Traffic Workloads: To show robustness under various traffic patterns, we evaluate 4x4 and 8x8 networks with uniform-random, bit-complement, and transpose synthetic traffic [11] in addition to application workloads. For each pattern, network injection rate is swept from zero to network saturation.

Performance Metrics: To measure system performance, we use the well-known *Weighted Speedup* metric [44]: $WS = \sum_{i=1}^{N} \frac{IPC_i^{shared}}{IPC_i^{alone}}$. All IPC_i^{alone} values are measured on the baseline bufferless network. Weighted speedup correlates to system throughput [16] and is thus a good general metric for multiprogrammed workloads.

Power Modeling: We use a modified and improved version of ORION 2.0 [49] (configured for 65nm), as developed by Grot et al. [24], as well as Verilog models synthesized with a commercial 65nm design library. We use Verilog models of router control logic, and add datapath area and power using ORION models for crossbars, buffers, links, and pipeline registers. CHIPPER and MinBD do not use a conventional crossbar, instead decoupling the crossbar into a permutation network, which we model using muxes. We rely on ORION's modeling for the baseline input-buffered router's control logic (e.g., arbitration). This hybrid synthesis / ORION approach models each portion of the router in a way that captures its key limitations. The control logic is logic- rather than wiring-dominated, and its arbitration paths determine the critical path; hence, standard-cell synthesis will model area, power and timing of MinBD and CHIPPER control logic with reasonable accuracy. The router datapath is wiring-dominated and relies on heavily-optimized components with custom layouts such as large crossbars and wide muxes, pipeline registers and memories. ORION explicitly models such router components.

[7] 410.bwaves, 416.gamess and 434.zeusmp were excluded because we were not able to collect representative traces from these applications.

We report energy efficiency as performance-per-watt, computed as weighted speedup divided by average network power.

1.6 Evaluation

In this section, we evaluate MinBD against a bufferless deflection router [17] and an input-buffered router with buffer bypassing [37, 50], as well as a hybrid of these two, AFC [29], and demonstrate that by using a combination of deflection routing and buffering, MinBD achieves performance competitive with the conventional input-buffered router (and higher than the bufferless deflection router), with a smaller buffering requirement, and better energy efficiency than all prior designs.

1.6.1 Application Performance

Fig. 1.8 (top pane) shows application performance as weighted speedup for 4x4 (16-node) and 8x8 (64-node) CMP systems. The plots show average results for each workload category, as described in §1.5, as well as overall average results. Each bar group shows the performance of three input-buffered routers: 8 VCs with 8 flits/VC, 4 VCs with 4 flits/VC, and 4 VCs with 1 flit/VC. Next is CHIPPER, the bufferless deflection router, followed by AFC [29], a coarse-grained hybrid router that switches between a bufferless and a buffered mode. MinBD is shown last in each bar group. We make several observations:

1. MinBD improves performance relative to the bufferless deflection router by 2.7% (4.9%) in the 4x4 (8x8) network over all workloads, and 8.1% (15.2%) in the highest-intensity category. Its performance is within 2.7% (3.9%) of the $(4, 4)$ input-buffered router, which is a reasonably-provisioned baseline, and within 3.1% (4.2%) of the $(8, 8)$ input-buffered router, which has large, power-hungry buffers. Hence, adding a side buffer allows a deflection router to obtain significant performance improvement, and the router becomes more competitive with a conventional buffered design.

2. Relative to the 4-VC, 1-flit/VC input-buffered router (third bar), which is the smallest deadlock-free (i.e., correct) design, MinBD performs nearly the same despite having less buffer space (4 flits in MinBD vs. 16 flits in $(4, 1)$-buffered). Hence, buffering only a portion of traffic (i.e., flits that would have been deflected) makes more efficient use of buffer space.

3. AFC, the hybrid bufferless/buffered router which switches modes at the router granularity, performs essentially the same as the 4-VC, 4-flit/VC input-buffered router, because it is able to use its input buffers when load

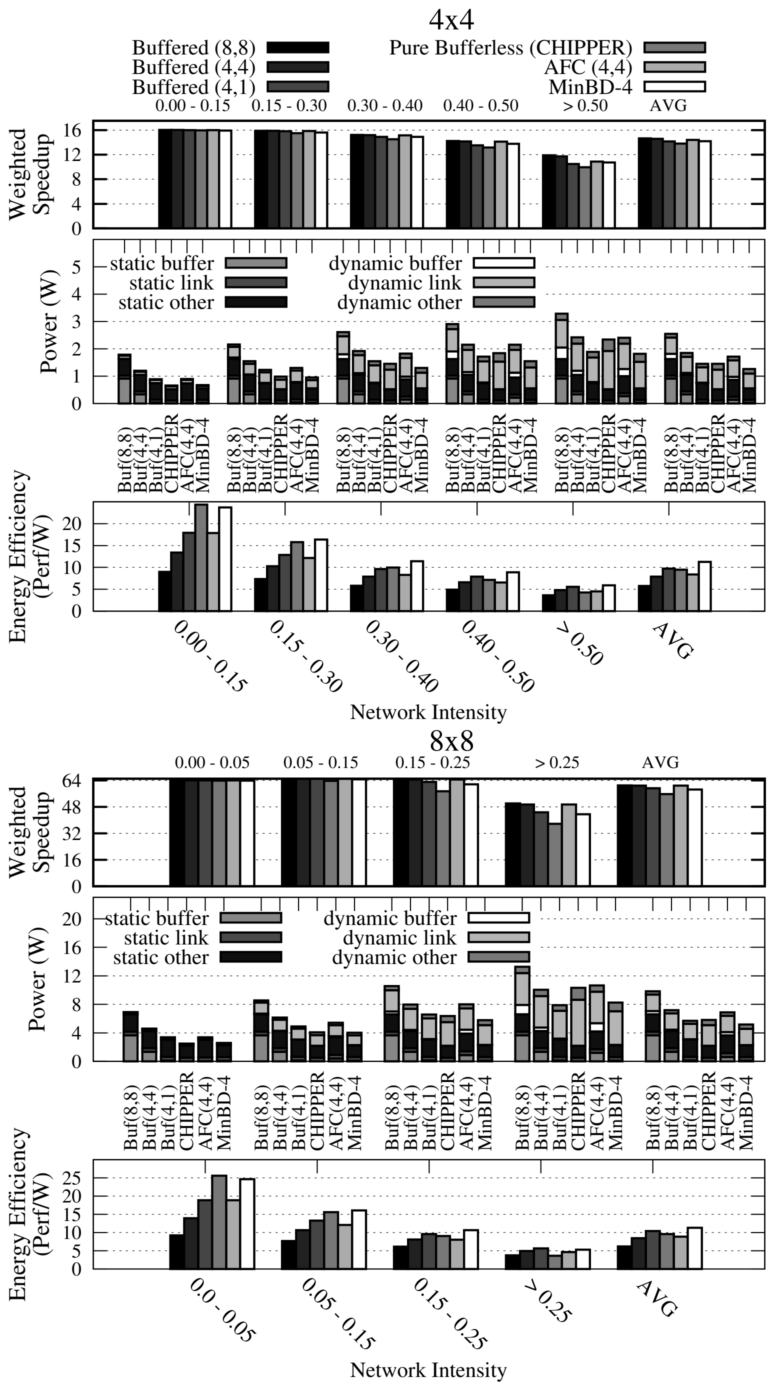

Fig. 1.8 Performance (weighted speedup), network power, and energy efficiency (performance per watt) in 16 (4x4) and 64 (8x8)-node CMP systems.

increases. However, as we will see, this performance comes at an efficiency cost relative to our hybrid design.

1.6.2 Network Power and Energy Efficiency

Network Power: Fig. 1.8 (middle pane) shows average total network power, split by component and type (static/dynamic), for 4x4 and 8x8 networks across the same workloads. Note that static power is shown in the bottom portion of each bar, and dynamic power in the top portion. Each is split into buffer power, link power, and other power (which is dominated by datapath components, e.g., the crossbar and pipeline registers). We make several observations:

1. Buffer power is a large part of total network power in the input-buffered routers that have reasonable buffer sizes, i.e., $(4, 4)$ and $(8, 8)$ (VCs, flits/VC), even with empty-buffer bypassing, largely because static buffer power (bottom bar segment) is significant. Removing large input buffers reduces static power in MinBD as well as the purely-bufferless baseline, CHIPPER.[8] Because of this reduction, MinBD's total network power never exceeds that of the input-buffered baselines, except in the highest-load category in an 8x8-mesh (by 4.7%).

2. Dynamic power is larger in the baseline deflection-based router, CHIPPER, than in input-buffered routers: CHIPPER has 31.8% (41.1%) higher dynamic power than the $(4, 4)$-buffered router in the 4x4 (8x8) networks in the highest-load category. This is because bufferless deflection-based routing requires more network hops, especially at high load. However, in a 4x4 network, MinBD consumes less dynamic power (by 8.0%) than the $(4, 4)$-buffered baseline in the highest-load category because reduced deflection rate (by 58%) makes this problem relatively less significant, and allows savings in buffer dynamic energy and a simplified datapath to come out. In an 8x8 network, MinBD's dynamic power is only 3.2% higher.

3. MinBD and CHIPPER, which use a permutation network-based datapath rather than a full 5x5 crossbar, reduce datapath static power (which dominates the "static other" category) by 31.0%: the decoupled permutation network arrangement has less area, in exchange for partial permutability (which causes some deflections). Input-buffered routers and AFC require a full crossbar because they cannot deflect flits when performing buffered routing (partial permutability in a non-deflecting router would significantly complicate switch arbitration, because each output arbiter's choice would be limited by which other flits are traversing the router).

[8] Note that network power in the buffered designs takes buffer bypassing into account, which reduces these baselines' dynamic buffer power. The $(4, 4)$-buffered router bypasses 73.7% (83.4%) of flits in 4x4 (8x8) networks. Without buffer bypassing, this router has 7.1% (6.8%) higher network power, and 6.6% (6.4%) worse energy-efficiency.

4. AFC, the coarse-grained hybrid, has nearly the same network power as the $(4, 4)$ buffered router at high load: 0.6% (5.7%) less in 4x4 (8x8). This is because its buffers are enabled most of the time. At low load, when it can power-gate its buffers frequently, its network power reduces. However, AFC's network power is still higher than the pure bufferless router (CHIPPER) or MinBD because (i) it still spends some time in its buffered mode, and (ii) its datapath power is higher, as described above. On average, AFC still consumes 36.8% (18.1%) more network power than CHIPPER, and 33.5% (33.0%) more than MinBD, in the lowest-load category.

Energy efficiency: Fig. 1.8 (bottom pane) shows energy efficiency. We make two key observations:

1. MinBD has the highest energy efficiency of any evaluated design: on average in 4x4 (8x8) networks, 42.6% (33.8%) better than the reasonably-provisioned $(4, 4)$ input-buffered design. MinBD has 15.9% (8.7%) better energy-efficiency than the most energy-efficient prior design, the $(4, 1)$-buffered router.

2. At the highest network load, MinBD becomes less energy-efficient compared to at lower load, and its efficiency degrades at a higher rate than the input-buffered routers with large buffers (because of deflections). However, its per-category energy-efficiency is still better than all baseline designs, with two exceptions. In the highest-load category (near saturation) in an 8x8-mesh, MinBD has nearly the same efficiency as the $(4, 1)$-buffered router (but, note that MinBD is much more efficient than this baseline router at lower loads). In the lowest-load category in a 4x4 mesh, the purely-bufferless router CHIPPER is slightly more energy-efficient (but, note that CHIPPER's performance and efficiency degrade quickly at high loads).

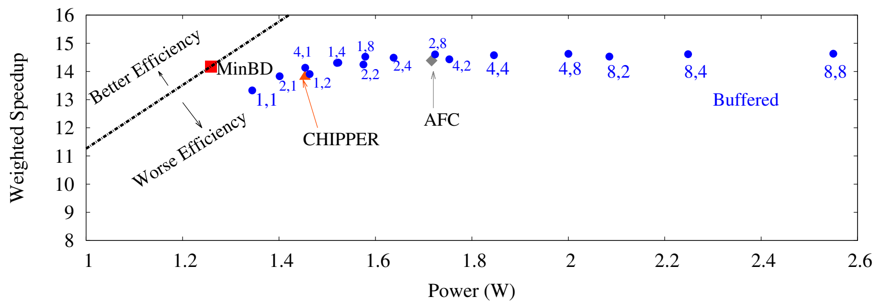

Fig. 1.9 Power (X) vs. application performance (Y) in 4x4 networks. The line represents all points with equivalent performance-per-watt to MinBD.

We conclude that, by achieving competitive performance with the buffered baseline, and making more efficient use of a much smaller buffer capacity (hence reducing buffer power and total network power), MinBD provides better energy efficiency than prior designs. To summarize this result, we show a 2D plot of power and application performance in Fig. 1.9 for 4x4 networks, and a wider range of buffered router designs, as well as MinBD and CHIPPER. (Recall from §1.5 that several of the baseline input-buffered designs are not deadlock free (too few VCs) or have a buffer depth that does not cover credit round-trip latency, but we evaluate them anyway for completeness.) In this plot, with power on the X axis and application performance on the Y axis, a line through the origin represents a fixed performance-per-watt ratio (the slope of the line). This equal-efficiency line is shown for MinBD. Points above the line have better efficiency than MinBD, and points below have worse efficiency. As shown, MinBD presents the best energy efficiency among all evaluated routers. The trend in an 8x8 network (not shown for space) is similar (see technical report [18]).

1.6.3 Performance Breakdown

To understand the observed performance gain in more detail, we now break down performance by each component of MinBD. Fig. 1.10 shows performance (for 4x4 networks) averaged across all workloads for eight *deflection systems*, which constitute all possible combinations of MinBD's mechanisms added to the baseline (CHIPPER) router: dual-width ejection (**D**), silver-flit prioritization (**S**), and the side buffer (**B**), shown with the same three input-buffered configurations as before. The eighth bar (**D+S+B**), which represents all three mechanisms added to the baseline deflection network, represents MinBD. Table 1.2 shows deflection rate for the same set of systems.

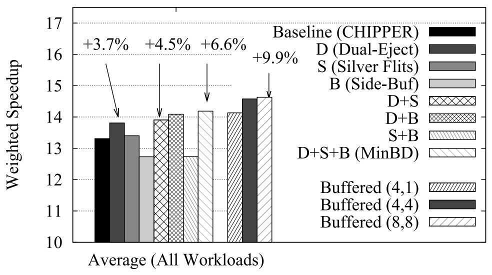

Fig. 1.10 Breakdown of performance gains for each mechanism in MinBD.

Table 1.2 Average deflection rates for deflection-based routers.

Baseline (CHIPPER)	D	S	B	D+S	D+B	S+B	D+S+B (MinBD)
0.28	0.22	0.27	0.17	0.22	0.11	0.16	0.10

Table 1.3 Normalized router area and critical path for bufferless and buffered baselines, compared to MinBD.

Router Design	CHIPPER	MinBD	Buffered $(8, 8)$	Buffered $(4, 4)$	Buffered $(4, 1)$
Normalized Area	1.00	1.03	2.06	1.69	1.60
Normalized Critical Path Length	1.00	1.07	0.99	0.99	0.99

We draw three main conclusions:

1. All mechanisms individually contribute to performance and reduce deflection rate. Dual ejection (**D**) increases performance by 3.7% over baseline CHIPPER.[9] Adding silver-flit prioritization (**D+S**) increases performance by 0.7% over dual-ejection alone. Adding a side buffer to the other two mechanisms (**D+S+B**) increases performance by 2.0% on average.

2. Adding a side buffer by itself (**B**) is not sufficient to attain the performance that MinBD achieves. In fact, when only the side buffer is added, performance drops by 4.3% relative to baseline CHIPPER. The degradation occurs primarily because the increased in-network throughput (enabled by a reduced deflection rate) in the network with side buffers places more ejection pressure on nodes, which exacerbates the ejection bottleneck that we observed in §1.4.1. This performance reduction comes despite a reduced deflection rate: even though flits are not deflected as frequently, they must still be removed from the network efficiently for performance to increase.

3. Adding dual ejection to the side buffered system (**D+B**) to address the ejection bottleneck increases performance to 5.8% above baseline. Silver-flit prioritization in addition to this configuration point yields the MinBD router (eighth bar), which attains 6.6% performance above baseline (2.7% above dual-ejection alone) on average for all workload intensities. Overall, deflection rate reduces by 64% from baseline CHIPPER to MinBD (and 54% from CHIPPER with dual-ejection (**D**) to MinBD, as shown in our primary results).

1.6.4 Synthetic Traffic Performance

We study the network-level performance of MinBD and baseline designs by applying synthetic traffic patterns: uniform random, bit-complement, and transpose [11]. Fig. 1.11 shows latency curves with injection rate swept from zero to saturation for the bufferless deflection router, MinBD, and the $(4, 1)$ and $(8, 8)$ input-buffered router (other input-buffered routers are omitted for clarity; these are the smallest and largest evaluated input-buffered routers, to show lower and upper bounds). Latency curves are shown for uniform

[9] The main results presented in Fig. 1.8 use this data point (with dual ejection) in order to make a fair (same external router interface) comparison.

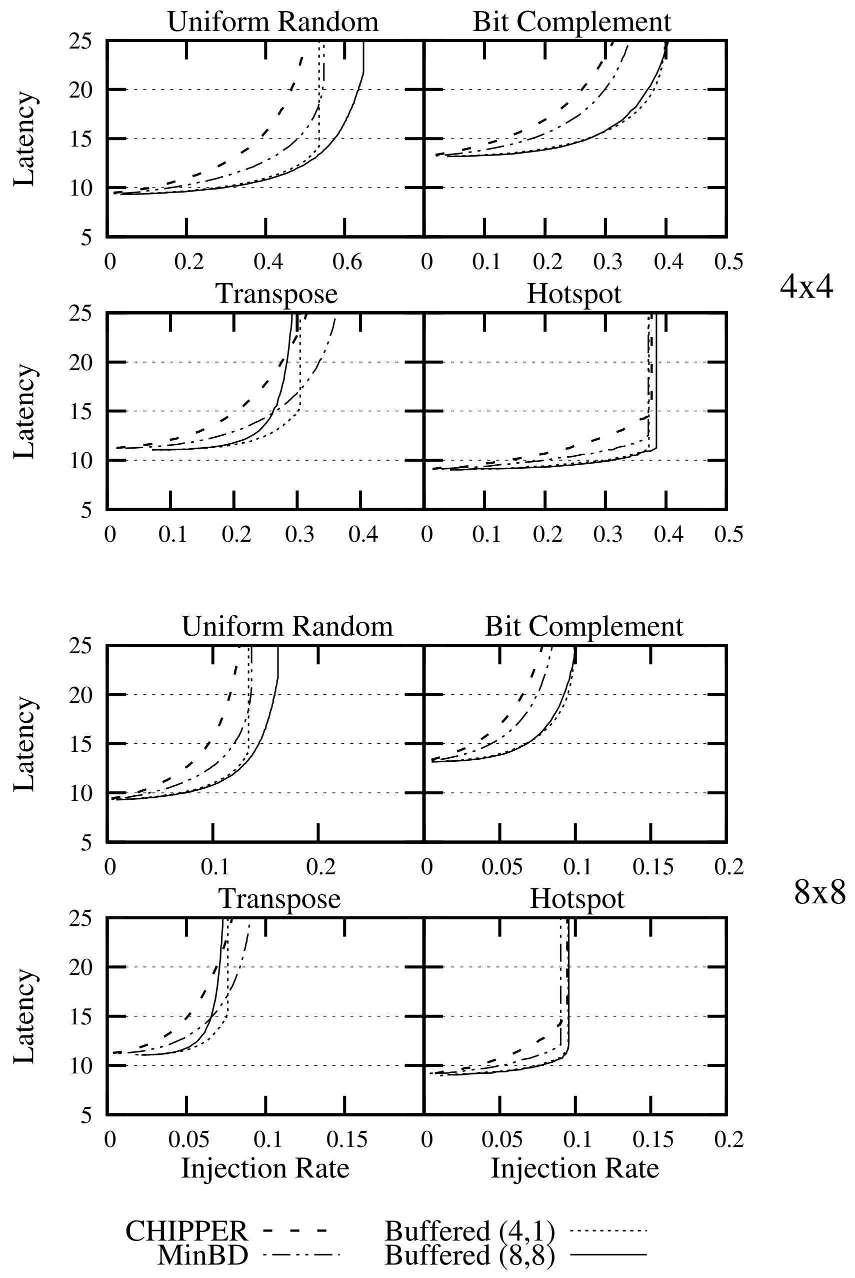

Fig. 1.11 Synthetic traffic evaluations for MinBD, CHIPPER and input-buffered routers (with small and large input buffers), in 4x4 and 8x8 meshes.

random, bit complement, transpose, and hotspot patterns. In the "hotspot" pattern, nodes send requests to a single node at the center of the mesh with 20% probability and to random locations otherwise. (Under a 100% hotspot pattern, the performance of all designs converges because the receiving node's ejection bandwidth is the bottleneck. A 20% skewed hotspot pattern is more realistic because it tests the network's capacity to handle unbalanced load.)

Under uniform random traffic (which most closely resembles our multiprogrammed application workloads with striped data mapping), MinBD performs better than the bufferless baseline, with a higher saturation point. MinBD has a lower network saturation point than the input-buffered network with large input buffers, but very similar saturation point to the small $(4, 1)$ input-buffered router, as our earlier results indicated. We conclude that with only 4 flits of buffering per router, MinBD closes nearly half of the gap in network saturation throughput between the bufferless router (CHIPPER) and the largest input-buffered router (with 256 flits of total buffer space), and performs similarly to a smaller input-buffered router with 16 flits of total buffer space.

In addition, non-uniform traffic patterns demonstrate the robustness and adaptivity of deflection routing: in particular, the transpose traffic pattern demonstrates a lower saturation point in the input-buffered router than either deflection-based router (MinBD or CHIPPER). This advantage occurs because deflection routing is *adaptive* (a flit's path can change based on network conditions). Deflections spread traffic away from hotspots and balance load in unbalanced traffic patterns. While adaptive routing is also possible in input-buffered routers, it is more complex because it requires the router to track network load or congestion (locally or globally) and make decisions accordingly. In contrast, deflection routing provides adaptivity by virtue of its ordinary operation.

1.6.5 Sensitivity to Parameters

Side Buffer Size: As side buffer size is varied from 1 to 64 flits, mean weighted speedup (application performance) changes only 0.2% on average across all workloads (0.9% in the highest-intensity category) in 4x4 networks. We conclude that the *presence* of the buffer (to buffer at least one deflected flit) is more important than its size, because the average utilization of the buffer is low. In a 4x4 MinBD network with 64-flit side buffers at saturation (61% injection rate, uniform random), the side buffer is empty 48% of the time on average; 73% of the time, it contains 4 or fewer flits; 93% of the time, 16 or fewer. These measurements suggest that a very small side buffer captures most of the benefit. Furthermore, total network power increases by 19% (average across all 4x4 workloads) when a 1-flit buffer per router is increased to a 64-flit buffer per router. Hence, a larger buffer wastes power without significant performance benefit.

We avoid a 1-flit side buffer because of the way the router is pipelined: such a single-flit buffer would either require for a flit to be able to enter, then leave, the buffer in the same cycle (thus eliminating the independence of the two router pipeline stages), or else could sustain a throughput of one flit only every two cycles. (For this sensitivity study, we optimistically assumed the former option for the 1-flit case.) The 4-flit buffer we use avoids this pipelining issue, while increasing network power by only 4% on average over the 1-flit buffer.

Note that although the size we choose for the side buffer happens to be the same as the 4-flit packet size which we use in our evaluations, this need not be the case. In fact, because the side buffer holds deflected *flits* (not packets) and deflection decisions occur at a per-flit granularity, it is unlikely that the side buffer will hold more than one or two flits of a given packet at a particular time. Hence, unlike conventional input-buffered routers which typically size a buffer to hold a whole packet, MinBD's side buffer can remain small even if packet size increases.

Packet Size: Although we perform our evaluations using a 4-flit packet size, our conclusions are robust to packet size. In order to demonstrate this, we also evaluate MinBD, CHIPPER, and the $(4, 4)$- and $(8, 8)$-input-buffered routers in 4x4 and 8x8 networks using a data packet size of 8 flits. In a 4x4 (8x8) network, MinBD improves performance over CHIPPER by 17.1% (22.3%), achieving performance within 1.2% (8.1%) of the $(4, 4)$-input-buffered router and within 5.5% (12.8%) of the $(8, 8)$-input-buffered router, while reducing average network power by 25.0% (18.1%) relative to CHIPPER, 16.0% (9.4%) relative to the $(4, 4)$-input-buffered router, and 40.3% (34.5%) relative to the $(8, 8)$-input-buffered router, respectively. MinBD remains the most energy-efficient design as packet size increases.

1.6.6 Hardware Cost: Router Area and Critical Path

We present normalized router area and critical path length in Table 1.3. Both metrics are normalized to the bufferless deflection router, CHIPPER, because it has the smallest area of all routers. MinBD adds only 3% area overhead with its small buffer. In both CHIPPER and MinBD, the datapath dominates the area. In contrast, the large-buffered baseline has 2.06x area, and the reasonably-provisioned buffered baseline has 1.69x area. Even the smallest deadlock-free input-buffered baseline has 60% greater area than the bufferless design (55% greater than MinBD). In addition to reduced buffering, the reduction seen in CHIPPER and MinBD is partly due to the simplified datapath in place of the 5x5 crossbar (as also discussed in §1.6.2). Overall, MinBD reduces area relative to a conventional input-buffered router both by significantly reducing the required buffer size, and by using a more area-efficient datapath.

Table 1.3 also shows the normalized critical path length of each router design, which could potentially determine the network operating frequency. MinBD increases critical path by 7% over the bufferless deflection router, which in turn has a critical path 1% longer than an input-buffered router. In all cases, the critical path is through the flit arbitration logic (the permutation network in MinBD and CHIPPER, or the arbiters in the input-buffered router). MinBD increases critical path relative to CHIPPER by adding logic in the deflection-routing stage to pick a flit to buffer, if any. The buffer re-injection and redirection logic in the first pipeline stage (ejection/injection) does not impact the critical path because the permutation network pipeline stage has a longer critical path.

1.7 Related Work

To our knowledge, MinBD is the first NoC router design that combines deflection routing with a small side buffer that reduces deflection rate. Other routers combine deflection routing with buffers, but do not achieve the efficiency of MinBD because they either continue to use input buffers for all flits (Chaos router) or switch all buffers on and off at a coarse granularity with a per-router mode switch (AFC), in contrast to MinBD's fine-grained decision to buffer or deflect each flit.

Buffered NoCs that also use deflection: Several routers that primarily operate using buffers and flow control also use deflection routing as a secondary mechanism under high load. The Chaos Router [32] deflects packets when a packet queue becomes full to probabilistically avoid livelock. However, all packets that pass through the router are buffered; in contrast, MinBD performs deflection routing first, and only buffers some flits that would have been deflected. This key aspect of our design reduces buffering requirements and buffer power. The Rotary Router [1] allows flits to leave the router's inner ring on a non-productive output port after circulating the ring enough times, in order to ensure forward progress. In this case, again, deflection is used as an escape mechanism to ensure probabilistic correctness, rather than as the primary routing algorithm, and all packets must pass through the router's buffers.

Other bufferless designs: Several prior works propose bufferless router designs [17, 21, 26, 38, 48]. We have already extensively compared to CHIPPER [17], from which we borrow the deflection routing logic design. BLESS [38], another bufferless deflection network, uses a more complex deflection routing algorithm. Later works showed BLESS to be difficult to implement in hardware [17, 26, 37], thus we do not compare to it in this work. Other bufferless networks drop rather than deflect flits upon contention [21, 26].

Some earlier large multiprocessor interconnects, such as those in HEP [42] and Connection Machine [27], also used deflection routing. The HEP router combined some buffer space with deflection routing [43]. However, these routers' details are not well-known, and their operating conditions (large off-chip networks) are significantly different than those of modern NoCs.

More recently, Fallin et al. [19] applied deflection routing to a hierarchical ring topology, allowing most routers (those that lie within a ring) to be designed without any buffering or flow control, and using only small buffers to transfer between rings. The resulting design, HiRD, was shown to be more energy-efficient than the baseline hierarchical ring with more buffering. HiRD uses many of the same general ideas as MinBD to ensure forward progress, e.g., enforcing explicit forward-progress guarantees in the worst case without impacting common-case complexity.

Improving high-load performance in bufferless networks: Some work has proposed *congestion control* to improve performance at high network load in bufferless deflection networks [8, 39, 40]. Both works used source throttling: when network-intensive applications cause high network load which degrades performance for other applications, these intensive applications are prevented from injecting network traffic some of the time. By reducing network load, source throttling reduces deflection rate and improves overall performance and fairness. These congestion control techniques and others (e.g., [47]) are orthogonal to MinBD, and could improve MinBD's performance further.

Hybrid buffered-bufferless NoCs: AFC [29] combines a bufferless deflection router based on BLESS [38] with input buffers, and switches between bufferless deflection routing and conventional input-buffered routing based on network load at each router. While AFC has the performance of buffered routing in the highest-load case, with better energy efficiency in the low-load case (by power-gating buffers when not needed), it misses opportunity to improve efficiency because it switches buffers on at a coarse granularity. When an AFC router experiences high load, it performs a mode switch which takes several cycles in order to turn on its buffers. Then, it pays the buffering energy penalty for every flit, whether or not it would have been deflected. It also requires buffers as large as the baseline input-buffered router design in order to achieve equivalent high-load performance. As a result, its network power is nearly as high as a conventional input-buffered router at high load, and it requires fine-grained power gating to achieve lower power at reduced network load. In addition, an AFC router has a larger area than a conventional buffered router, because it must include both buffers and buffered-routing control logic as well as deflection-routing control logic. In contrast, MinBD does not need to include large buffers and the associated buffered-mode control logic, instead using only a smaller side buffer. MinBD also removes the dependence on efficient buffer power-gating that AFC requires for energy-efficient operation at low loads. We quantitatively compared MinBD to AFC in §1.4 and demonstrated better energy efficiency at all network loads.

Reducing cost of buffered routers: Empty buffer bypassing [50, 37] reduces buffered router power by allowing flits to bypass input buffers when empty. However, as our evaluations (which faithfully model the power reductions due to buffer bypassing) show, this scheme reduces power less than our new router design: bypassing is only effective when buffers are empty, which happens more rarely as load increases. Furthermore, buffers continue to consume static power, even when unused. Though both MinBD and empty-buffer-bypassed buffered routers avoid buffering significant traffic, MinBD further reduces router power by using much smaller buffers.

Kim [30] proposed a low-cost buffered router design in which a packet uses a buffer only when turning, not when traveling straight along one dimension. Unlike our design, this prior work does not make use of deflection, but uses deterministic X-Y routing. Hence, it is not adaptive to different traffic patterns. Furthermore, its performance depends significantly on the size of the turn-buffers. By using deflection, MinBD is less dependent on buffer size to attain high performance, as we argued in §1.6.5. In addition, [30] implements a token-based injection starvation avoidance scheme which requires additional communication between routers, whereas MinBD requires only per-router control to ensure side buffer injection.

1.8 Conclusion

In this chapter, we introduced deflection routing and discussed two bufferless deflection routers, BLESS [38] and CHIPPER [17]. We then described MinBD [20], a minimally-buffered deflection router design. MinBD combines deflection routing with a small buffer, such that some network traffic that would have been deflected is placed in the buffer instead. Previous router designs which use buffers typically place these buffers at the router inputs. In such a design, energy is expended to read and write the buffer for every flit, and buffers must be large enough to efficiently handle all arriving traffic. In contrast to prior work, a MinBD router uses its buffer for only a fraction of network traffic, and hence makes more efficient use of a given buffer size than a conventional input-buffered router. Its average network power is also greatly reduced: relative to an input-buffered router, buffer power is much lower, because buffers are smaller. Relative to a bufferless deflection router, dynamic power is lower, because deflection rate is reduced with the small buffer.

We evaluate MinBD against a comprehensive set of baseline router designs: three configurations of an input-buffered virtual-channel router [11], a bufferless deflection router, CHIPPER [17], and a hybrid buffered-bufferless router, AFC [29]. Our evaluations show that MinBD performs competitively and reduces network power: on average in a 4x4 network, MinBD performs within 2.7% of the input-buffered design (a high-performance baseline) while consuming 31.8% less total network power on average relative to this input-buffered router (and 13.4% less than the bufferless router, which performs worse than MinBD). Finally, MinBD has the best energy efficiency among all routers which we evaluated. We conclude that a router design which augments bufferless deflection routing with a small buffer to reduce deflection rate is a compelling design point for energy-efficient, high-performance on-chip interconnect.

1.9 Future Work

We believe promising future research directions exist in bufferless and minimally-buffered deflection routers. One promising direction is to develop more effective packet prioritization mechanisms that are applicable to such routers. For example, to improve system performance and fairness, mechanisms have been proposed to perform packet prioritization in a manner that is aware of application characteristics [13] or packet latency slack [14]. Similarly, mechanisms have been proposed to provide quality of service to different applications in buffered routers (e.g., [23, 24, 35]). Development of similar mechanisms for bufferless and minimally buffered routers is an open research problem. Another promising direction is to develop techniques to increase the applicability of bufferless and minimally buffered deflection routing to different topologies, such as express cube topologies [22, 31] and hybrid and hierarchical topologies [2, 15, 24]. Adaptation of bufferless and minimally-buffered deflection routing in a manner that guarantees correctness in different topologies, and the evaluation of resulting techniques is an important area for future work. As a first step, our recent research [19] has developed new techniques to use deflection routing in hierarchical ring topologies, with promising results that show that a minimally-buffered hierarchical ring design significantly improves system energy efficiency. We intend to explore these research directions in the future and hope that this chapter serves as an inspiration to other researchers as well.

Acknowledgments

We thank the anonymous reviewers of our conference papers CHIPPER [17] and MinBD [20] for their feedback. We gratefully acknowledge members of the SAFARI group and Michael Papamichael at CMU for feedback. Chris Fallin is currently supported by an NSF Graduate Research Fellowship (Grant No. 0946825). Rachata Ausavarungnirun is currently supported by the Royal Thai Government Scholarship. Greg Nazario and Xiangyao Yu were undergraduate research interns while this work was done. We acknowledge the generous support of our industrial partners, including AMD, HP Labs, IBM, Intel, NVIDIA, Oracle, Qualcomm, and Samsung. This research was partially supported by grants from NSF (CAREER Award CCF-0953246, CCF-1147397 and CCF-1212962). This article is a significantly extended and revised version of our previous conference papers that introduced CHIPPER [17] and MinBD [20].

References

[1] Abad, P., et al.: Rotary router: an efficient architecture for CMP interconnection networks. ISCA-34 (2007)
[2] Balfour, J., Dally, W.J.: Design tradeoffs for tiled CMP on-chip networks. ICS (2006)
[3] Baran, P.: On distributed communications networks. IEEE Transactions on Communication Systems (1964)
[4] Borkar, S.: Thousand core chips: a technology perspective. DAC-44 (2007)
[5] Borkar, S.: Future of interconnect fabric: a contrarian view. SLIP '10 (2010)
[6] Borkar, S.: NoCs: What's the point? NSF Workshop on Emerging Tech. for Interconnects (WETI), Feb. 2012 (2012)
[7] Bose, P.: The power of communication: Trends, challenges (and accounting issues). NSF WETI, Feb. 2012 (2012)
[8] Chang, K., et al.: HAT: Heterogeneous adaptive throttling for on-chip networks. SBAC-PAD (2012)
[9] Culler, D.E., et al.: Parallel Computer Architecture: A Hardware/Software Approach. Morgan Kaufmann (1999)
[10] Dally, W., Seitz, C.: Deadlock-free message routing in multiprocessor interconnection networks. IEEE Trans. Comp. (1987)
[11] Dally, W., Towles, B.: Principles and Practices of Interconnection Networks. Morgan Kaufmann (2004)
[12] Dally, W.J., Towles, B.: Route packets, not wires: On-chip interconnection networks. DAC-38 (2001)
[13] Das, R., Mutlu, O., Moscibroda, T., Das, C.: Application-aware prioritization mechanisms for on-chip networks. MICRO-42 (2009)
[14] Das, R., et al.: Aérgia: exploiting packet latency slack in on-chip networks. ISCA-37 (2010)
[15] Das, R., et al.: Design and evaluation of hierarchical on-chip network topologies for next generation CMPs. HPCA-15 (2011)
[16] Eyerman, S., Eeckhout, L.: System-level performance metrics for multiprogram workloads. IEEE Micro **28**, 42–53 (2008)
[17] Fallin, C., et al.: CHIPPER: A low-complexity bufferless deflection router. HPCA-17 (2011)

[18] Fallin, C., et al.: MinBD: Minimally-buffered deflection routing for energy-efficient interconnect. SAFARI technical report TR-2011-008: `http://safari.ece.cmu.edu/tr.html` (2011)

[19] Fallin, C., et al.: HiRD: A low-complexity, energy-efficient hierarchical ring interconnect. SAFARI technical report TR-2012-004: `http://safari.ece.cmu.edu/tr.html` (2012)

[20] Fallin, C., et al.: MinBD: Minimally-buffered deflection routing for energy-efficient interconnect. NOCS-4 (2012)

[21] Gómez, C., et al.: Reducing packet dropping in a bufferless noc. Euro-Par-14 (2008)

[22] Grot, B., Hestness, J., Keckler, S., Mutlu, O.: Express cube topologies for on-chip interconnects. HPCA-15 (2009)

[23] Grot, B., Keckler, S., Mutlu, O.: Preemptive virtual clock: A flexible, efficient, and cost-effective qos scheme for networks-on-chip. MICRO-42 (2009)

[24] Grot, B., et al.: Kilo-NOC: A heterogeneous network-on-chip architecture for scalability and service guarantees. ISCA-38 (2011)

[25] Hansson, A., et al.: Avoiding message-dependent deadlock in network-based systems-on-chip. VLSI Design (2007)

[26] Hayenga, M., et al.: SCARAB: A single cycle adaptive routing and bufferless network. MICRO-42 (2009)

[27] Hillis, W.: The Connection Machine. MIT Press (1989)

[28] Hoskote, Y., et al.: A 5-GHz mesh interconnect for a teraflops processor. IEEE Micro (2007)

[29] Jafri, S.A.R., et al.: Adaptive flow control for robust performance and energy. MICRO-43 (2010)

[30] Kim, J.: Low-cost router microarchitecture for on-chip networks. MICRO-42 (2009)

[31] Kim, J., Balfour, J., Dally, W.: Flattened butterfly topology for on-chip networks. MICRO-40 (2007)

[32] Konstantinidou, S., Snyder, L.: Chaos router: architecture and performance. ISCA-18 (1991)

[33] Kroft, D.: Lockup-free instruction fetch/prefetch cache organization. ISCA-8 (1981)

[34] Laudon, J., Lenoski, D.: The SGI Origin: a ccNUMA highly scalable server. ISCA-24 (1997)

[35] Lee, J., Ng, M., Asanovic, K.: Globally-synchronized frames for guaranteed quality-of-service in on-chip networks. ISCA-35 (2008)

[36] Luk, C.K., et al.: Pin: building customized program analysis tools with dynamic instrumentation. PLDI (2005)

[37] Michelogiannakis, G., et al.: Evaluating bufferless flow-control for on-chip networks. NOCS (2010)

[38] Moscibroda, T., Mutlu, O.: A case for bufferless routing in on-chip networks. ISCA-36 (2009)

[39] Nychis, G., et al.: Next generation on-chip networks: What kind of congestion control do we need? Hotnets-IX (2010)

[40] Nychis, G., et al.: On-chip networks from a networking perspective: Congestion and scalability in many-core interconnects. SIGCOMM (2012)

[41] Patil, H., et al.: Pinpointing representative portions of large Intel Itanium programs with dynamic instrumentation. MICRO-37 (2004)

[42] Smith, B.: Architecture and applications of the HEP multiprocessor computer system. SPIE (1981)

[43] Smith, B.: Personal communication (2008)

[44] Snavely, A., Tullsen, D.M.: Symbiotic jobscheduling for a simultaneous multithreaded processor. ASPLOS-9 (2000)

[45] Standard Performance Evaluation Corporation: SPEC CPU2006. `http://www.spec.org/cpu2006` (2006)

[46] Taylor, M., et al.: The Raw microprocessor: A computational fabric for software circuits and general-purpose programs. IEEE Micro (2002)

[47] Thottethodi, M., Lebeck, A., Mukherjee, S.: Self-tuned congestion control for multiprocessor networks. HPCA-7 (2001)

[48] Tota, S., et al.: Implementation analysis of NoC: a MPSoC trace-driven approach. GLSVLSI-16 (2006)

[49] Wang, H., et al.: Orion: a power-performance simulator for interconnection networks. MICRO-35 (2002)

[50] Wang, H., et al.: Power-driven design of router microarchitectures in on-chip networks. MICRO-36 (2003)

[51] Wentzlaff, D., et al.: On-chip interconnection architecture of the tile processor. IEEE Micro **27**(5), 15–31 (2007)