MinBD: A Minimally-Buffered Deflection Router Approaching Conventional Buffered-Router Performance

Chris Fallin
cfallin@cmu.edu

Gregory Nazario
gnazario@cmu.edu

Xiangyao Yu
yxythu@gmail.com

Kevin Chang
kevincha@andrew.cmu.edu

Rachata Ausavarungnirun
rausavar@ece.cmu.edu

Onur Mutlu
onur@cmu.edu

Computer Architecture Lab (CALCM)
Carnegie Mellon University

SAFARI Technical Report No. 2011-008

September 13, 2011

Abstract

As interconnect becomes an important component of modern Chip Multiprocessors (CMPs), significant work has gone into finding the best tradeoff between performance and energy efficiency for the Network-on-Chip (NoC). Increasing core counts lead to high bandwidth demands and tight power and area constraints. One recent line of work examines *bufferless deflection routing*, which eliminates buffers in the NoC and instead handles contention by *deflecting* (misrouting) traffic.

While bufferless NoC design has shown promising area and power reductions, and offers similar performance to conventional buffered designs for many workloads, such designs provide lower throughput than conventional, more complex, buffered routers at high network load. This degradation is a significant hurdle for widespread adoption of bufferless NoCs.

In this work, we introduce a new minimally-buffered deflection router design, *MinBD* (Minimally-Buffered Deflection), that aims to obtain the performance of buffered design with nearly the area and power reductions of bufferless design. Unlike past routers that combine buffers and deflection, MinBD buffers only a fraction of the traffic that passes through it, and it decides which traffic to buffer on a fine-grained basis. We observe that a significant portion of the performance degradation in bufferless networks comes not from fundamental limits of deflection routing, but from small inefficiencies in previous bufferless designs that can be corrected: specifically, a bottleneck in ejecting traffic from the network, and inefficient deflection arbitration. The remaining performance gap can be closed by using small router buffers. Compared to previous bufferless deflection router designs, MinBD contributes (i) a router microarchitecture with a wide ejection path and improved flit arbitration relative to previous designs, and (ii) small side-buffers to hold some traffic that would have otherwise been deflected. We show that MinBD degrades performance only 4.6% from a conventional buffered network over a set of 60 network-intensive workloads in 4x4 networks, with 72.5% less network power on average, 80.0% less router area, and a competitive clock frequency.

1 Introduction

Interconnect is a first-order component of current and future multicore and manycore CMPs (Chip Multiprocessors). As manycore designs scale up, packet-switched interconnects replace conventional buses or crossbars [7]. Such networks – Networks-on-Chip, or NoCs – transfer packets, consisting of one or multiple flits, between cores on a chip. As manycore chips incorporate increasing numbers of cores, on-chip accelerators and other components, the bandwidth demand on the network increases. The NoC's design can thus become critical for system performance. Many current commercial CMPs use ring topologies [14, 27, 28], but several have moved to 2D-mesh NoCs (e.g., Tilera [34]), and

future manycores in general will likely move toward 2D-mesh [6] or other higher-dimension networks [17] in order to provide adequate performance scaling.

However, conventional buffered mesh NoCs are projected to consume a large portion of chip power. The Intel Terascale 80-core prototype reported approximately 30% of chip power consumed by the NoC [13]. Due to thermal design limits, power used by the NoC reduces the available power for other components. In addition, NoCs can consume a significant portion of on-die area, which could otherwise be used for more cores or other on-chip components. Hence, NoC area and power are first-order concerns, alongside performance.

To address these concerns, previous work proposes *bufferless deflection routing on chip* [9, 23]. By removing in-router buffering for traffic, and instead deflecting contending traffic to alternate network links, such a network achieves significant complexity, area, and power reductions over a baseline with buffers. A realistic implementation of this network design, CHIPPER [9], was shown to reduce NoC area and power by 36.2% and 54.9% respectively. Unfortunately, such bufferless deflection routing designs exhibit poor performance under workloads that stress the network: CHIPPER degrades performance by 13.6% on average across a large set of server, SPEC, and desktop applications [9], and up to 49.8% in one such workload. This *high-load problem* is a major inhibitor to realizing the power, area, and complexity-reduction advantages of bufferless designs in mainstream systems. While some previous work observes that combining buffered and bufferless designs in a dynamically-switching hybrid scheme [15] or using source throttling to reduce network congestion in bufferless networks [24, 25] can reduce the problem (by switching away from a bufferless network at high load or simply reducing the network's load), neither of these approaches studies the behavior of the deflection routing itself. Each approach continues to include a bufferless deflection network that is less efficient than it could be, and works around the deflection network's shortcomings in other ways. In contrast, we are interested in *why* bufferless routing performs worse than buffered routing; hence, we find and correct fundamental inefficiencies in bufferless deflection routing in general, and in a previous bufferless router design specifically.

Thus, **our major contribution** is to identify several of the root causes of this performance degradation, and propose a new deflection router design with nearly the performance of conventional buffered routers. Our key observation is that bufferless performance degradation is largely due to simple bottlenecks and design issues. In particular, ejection of traffic from the network becomes a bottleneck and causes additional deflections, and inefficient flit arbitration can cause unnecessarily high deflection rates. After proposing microarchitectural mechanisms to address both of these issues, we find that a small buffer that is used to hold only a portion of the network traffic, when it would have otherwise been deflected, is more effective than conventional input buffers for a given buffer size. Using this small buffer allows a deflection-based router to nearly match the performance of a VC-buffered router, with much less power consumption and router area footprint. These contributions make up our new deflection router design, *MinBD*, which attains performance only 4.6% less than a conventional buffered network on average, with 72.5% less average power and 80.0% less router area.[1]

Finally, we show that our new design, MinBD, is applicable to, and beneficial when used in, high-radix network designs. The new router design also provides benefit when used in systems that perform locality-aware data mapping, or use other techniques to reduce communication overhead and create more locality. We show that either our approach is effective in closing any performance gap that remains between conventional purely-bufferless designs and VC-buffered designs, or, when little performance gap exists, it provides significant energy reductions by carrying a given network load more efficiently. Overall, we demonstrate that MinBD is a general-purpose interconnect design that is significantly simpler and lower-cost than conventional, high-performance buffered designs, while significantly higher-performance than baseline simple, low-cost bufferless deflection designs.

2 Background

On-chip networks form the backbone of memory systems in most recently-proposed large-scale CMPs (chip multiprocessors) [3, 33, 39]. Most such systems are cache-coherent shared memory multiprocessors. Interconnect has served as the substrate for large cache-coherent systems for some time (e.g., for large multiprocessor systems such as SGI Origin [5, 19]), 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

[1] As we will show in detailed evaluations, comparing our design with small buffers to a conventional buffered design with larger buffers does not unfairly inflate the router area or power of the prior work, because if we reduce buffer size in the conventional design to match the buffer size in MinBD, the small-buffered conventional design performs 1.19% worse than MinBD on average. Furthermore, MinBD still uses 70.5% less router area, because it removes the conventional design's crossbar, and because one side-buffer is more area-efficient per buffer cell than multiple separate input buffers.

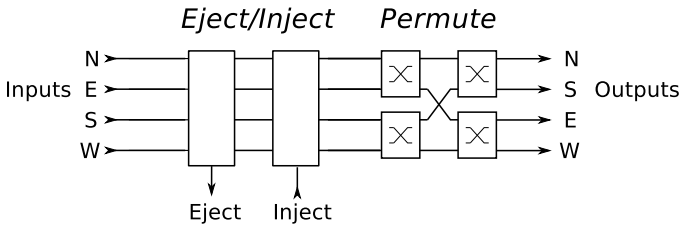


Figure 1: Baseline CHIPPER pipeline, as described in [9]. Figure based on Fig. 3 in [9].

data in order to fulfill memory accesses. 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 primarily to implement such a protocol between the cores, caches and memory controllers.

While many on-chip network designs have been proposed, we focus on bufferless interconnects in this work. Bufferless deflection routing was first proposed by Baran [2]. It has found a recent resurgence in research proposals for NoC design because on-chip wires (and thus, network links) are relatively cheap, in contrast to buffers, which consume significant die area and leakage power [23].

The fundamental unit of routing in a bufferless network is the *flit*, a packet fragment transferred by one link in one cycle. Flits are routed independently; thus, they must be reassembled after they are received. The basic operation of a bufferless 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, the router deflects one of them to another output. In such a design, careful attention must be given to avoid *livelock*, because in the worst case a flit may be deflected away from its destination forever.

BLESS [23] proposed this deflection-based bufferless routing scheme an alternative to traditional buffered NoCs. BLESS' key contribution is *prioritization by flit age* to avoid routing livelock and ensure that flits are delivered. BLESS proposes a simple routing algorithm that assigns flits to router outputs one at a time, in priority order, which is shown to perform well in [23]. However, later work [9, 11, 22] notes that BLESS is costly and slow when implemented in hardware because it requires a large arbiter with a long critical path, due to the sequential flit output assignment. Later work [9] proposes CHIPPER, a redesigned bufferless deflection router with a simpler microarchitecture and much shorter critical path. The CHIPPER pipeline is shown in Fig. 1. The key insight is that only the highest-priority flit needs to be routed correctly to ensure system correctness; this allows the arbiter design shown in the figure, which decouples the deflection routing problem into smaller independent arbitration problems in the "permute" stage. These independent arbitrations pick one winning flit in each arbiter block, and this flit moves closer to its requested output port.

3 Motivation

Past work in on-chip bufferless networks [9, 11, 23] was motivated largely by the observation that typical NoC designs are overprovisioned for the common case. Indeed, for low-to-medium network load, bufferless networks perform close to buffered networks. However, when the network approaches saturation, bufferless designs fall short of the peak performance of a buffered network. We refer to this difference in performance as the *performance gap*. Fig. 2 demonstrates the performance gap by showing average performance in two sets of workloads: 45 multiprogrammed workloads with low network utilization, and 60 workloads with high network utilization, in a 4x4-CMP network (see §5 for methodology and details). The left half of the figure shows application performance, measured as weighted speedup [31], which was shown to be equivalent to system throughput [8] (average IPC, and harmonic speedup show similar trends). As the figure shows, performance in the bufferless network CHIPPER [9] is comparable to performance in the buffered network at low load (because zero-load router latencies are the same in both designs), but performance degrades by 23.1% at high load. This occurs because high load causes many flits to be deflected, increasing the latency of the bufferless network.

Despite this performance degradation, the advantage of bufferless design becomes clear when we observe average network power, shown in the right half of Fig. 2. At low load, the average power consumed by a bufferless network

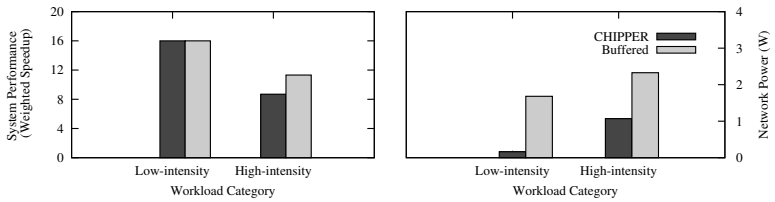


Figure 2: System performance, and average network power, shown for conventional buffered and bufferless networks with low- and high-network utilization workloads in a 4x4-CMP.

is significantly less than that of a buffered network, due to the removal of the buffers as well as simplification of the crossbar. At high load, the average power in a bufferless network rises significantly, because deflections force flits to take more hops on average, increasing the dynamic link and router power. Thus, a bufferless network loses its energy-efficiency advantage at high load both because of reduced application performance and increased network power.

Thus, although the power savings are an attractive feature of bufferless design, the performance gap is a significant obstacle to adopting such networks in mainstream designs, because such designs should handle heavy load efficiently. To solve this problem, previous work has proposed either building a hybrid bufferless-buffered system that switches modes based on load (AFC [15]) or else reducing network congestion in the bufferless network to improve performance [24, 25]. Although both of these approaches are effective to varying degrees at improving performance in a bufferless-based system at high load, they do not examine the fundamental problems in bufferless deflection routing itself. In particular, AFC solves the performance-gap problem by turning on buffers and switching to a buffered-routing mode at each router when load exceeds a threshold. The mechanism thus avoids using bufferless deflection routing at high load. Throttling-based approaches [24, 25] attack the performance gap by applying throttling (i.e., inhibiting nodes from injecting new traffic some of the time), reducing network utilization so that the bufferless deflection routing becomes more effective. In contrast, we aim to attack the performance problem in bufferless networks by examining the bufferless deflection routing itself. We wish to find bottlenecks and inefficiencies and address them directly with new mechanisms that preserve the fundamental area and energy advantages of bufferless deflection routing.

In summary, **our goal** is to understand the fundamental causes of the performance gap between bufferless and buffered networks, and propose a new deflection-based design that retains low network power and small die area while achieving better application performance. We now introduce such a design.

4 MinBD: An Efficient, Minimally-Buffered Deflection Router

We observe that the performance degradation that previous works have observed in bufferless deflection routing is largely not fundamental, but instead is due to particular bottlenecks and inefficiencies in previous designs. We will study several such bottlenecks in one previous bufferless design, CHIPPER [9], as a case study, and show how to design a new router that overcomes these bottlenecks. We will address two particular issues below: ejection bandwidth, and poor deflection arbitration. Then, after these two modifications have closed a portion of the performance gap at high load between bufferless and buffered networks, we introduce a small, central router buffer, the *side-buffer*, to close much of the remaining the gap. These contributions form *MinBD*, the minimally-buffered deflection router.

4.1 Ejection Bottleneck in Bufferless Deflection Routing

In a bufferless deflection network, traffic routes through the network, sometimes making productive hops and sometimes deflecting, until it eventually reaches its destination. When a flit reaches its destination node, the router at that node "ejects" it from the network. However, previous designs such (e.g., CHIPPER) provided only one ejection port at each router. Thus, in a given cycle, the router can only eject one flit; if more than one flit is destined for the local node, the other flits cannot be ejected, and will deflect. Those flits then eventually return to their destination and try again.

We observe that ejection becomes a bottleneck when the deflection rate is high, because traffic arrives in a bursty manner and the single-width ejection port often turns flits away. This not only penalizes the flit that was denied ejection, since it must deflect away and then try ejection again, but also increases network congestion even further.

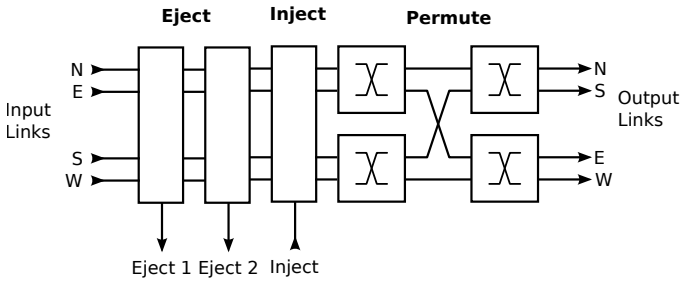


Figure 3: CHIPPER pipeline extended to eject up to two flits per cycle.

To demonstrate this problem, we measure the distribution of the number of flits eligible to eject at a router in a given cycle for our set of network-intensive workloads on a CHIPPER network. In 60 workloads (details in §5) that are network-intensive, at least one flit is eligible to eject 37.7% of the time. Of those cycles, 21.4% of the time, at least two flits are eligible to eject. Because CHIPPER permits only one flit to eject, these additional ejection candidates are instead deflected. Thus, in the CHIPPER design, a significant number of deflections actually occur because of these excess locally-destined flits (8% of all cycles at any given router).

In order to remove this bottleneck, we propose the use of a dual-width ejection path from the router to the local node. We note that ejecting at most two flits per cycle captures most ejection candidates: in our workloads, 97.1% of cycles with any ejection candidates have at most two flits wishing to eject. Across these workloads, the *average* ejection rate remains below one flit/cycle/node; thus, while the single-width datapath would be sufficient in the steady state, the burstiness in ejection significantly benefits from the peak ejection rate of 2 flits/cycle/node.

We show an implementation of dual-width ejection in the CHIPPER pipeline in Fig. 3. The ejector logic is duplicated to remove the second highest-priority locally-destined flit after the first locally-destined flit is ejected. Because the critical path in the original design occurs in the arbitration stage, rather than the eject/inject stage [9], this change does not have an impact on router speed. Note that widening the ejection datapath width to the local node requires a second write port to reassembly buffers. For a fair comparison, we assume that this change is made in the local/processor nodes in all evaluated router designs. Thus, we show, and present results relative to, a similarly-modified conventional buffered router with dual-width ejection.

4.2 Enhancing Arbitration with In-Router Prioritization

In a deflection router, flits must arbitrate for outputs in every cycle. The way in which a router determines which flit obtains which output, and which flits are deflected, can have a significant impact on deflection rate and thus the performance of a deflection network. In this section, we show how inefficiencies in the arbitration in the prior design CHIPPER [9] can be corrected. However, a similar technique might be applied to any deflection router that makes arbitration decisions in stages.

The CHIPPER router performs arbitration for output ports based on the insight that in a deflection-routed system, only *one* flit, namely the highest-priority flit, out of a set of inputs flits must be routed correctly to ensure that routing is livelock-free [9]. This allows the routing decision to be decomposed into two rounds of two-flit arbitration decisions in the parallel stages of the permutation network.

However, in CHIPPER, most packets and flits have the same (common traffic) priority. This is because *Golden Packet*, which CHIPPER uses to ensure livelock freedom, globally prioritizes only one packet in the system. All other flits are randomly arbitrated. While this simplifies the router, it leads to suboptimal arbitration, precisely because the priority rules break an arbitration tie *randomly*, and within a router, each flit goes through two stages of arbitration. If the prioritization decisions at these two stages do not agree on which flit has the highest priority, the router may deflect more flits than necessary, because of inconsistent arbitration decisions.

In order to provide better routing in the common case, when all flits have equal priority, we modify the router to randomly pick one flit to prioritize throughout both arbiter stages. We call this the "silver flit" in reference to the Golden Packet priority scheme (which is still in effect). This flit is consistently prioritized over other flits. When the silver flit leaves the router, its silver status is cleared; the "silver" control bit exists only within the router. Because the silver flit is picked randomly at each router, all flits have a fair chance at receiving this status at any given hop.

In order for this scheme to co-exist with Golden Packet, we modify the prioritization rules of Golden Packet to add

another priority level for silver flits between the golden level (which ensures livelock freedom) and the common-case non-prioritized flits. Thus, a silver flit will be prioritized over ordinary flits, but a golden flit will be prioritized over silver flits. Note that this hierarchy allows a silver flit to be chosen and marked independently of whether any golden flits are present; if golden flits are present, the Golden Packet livelock-freedom mechanism will continue to work as before. The modified prioritization rules for Golden Packet are shown in Ruleset 1.

Ruleset 1 Golden Packet with Silver Flits: Prioritization Rules (modified from [9])

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

Implementing the Silver Flit scheme requires two changes to the CHIPPER pipeline. First, each arbiter block in the arbitration stage must be modified to incorporate the additional priority level. This change has only a small impact on router speed, area and power, because each flit carries only a single additional control bit (to mark it as *silver*) and each arbitration stage requires only a small amount of additional circuitry. Second, the router must pick an input flit randomly at each cycle and mark that flit as silver. Note that this choice is completely independent of what flits are actually present at the inputs, and only depends on which inputs receive flits. Thus, the random choice of which flit to designate silver can be done in parallel to route computation, which occurs in the first stage of the pipeline, and this additional logic will not have an impact on the critical path.

4.3 Deflected-Flit Side-Buffering

So far, we have improved the efficiency of deflections in a purely bufferless deflection design. As we will show in §6, these changes taken together eliminate a significant portion of the performance gap between baseline bufferless and buffered designs. To address the remaining performance degradation, we add very limited buffering that completes *MinBD*, the minimally-buffered deflection router.

Key idea: A buffered router does not need to buffer all flits that pass through it. In order to buffer only some flits, and not require buffering in the common case, we propose a *side buffer* adjoined to a baseline bufferless-deflection router (as described in the previous subsections). The *key idea* is, at each cycle, to examine the results of router output arbitration and look for deflected flits. Then, when deemed worthwhile, remove some deflected flits from the router pipeline into the side-buffer, rather than actually deflecting them. At some later time, the router will take these flits from the buffer and re-inject them into the router pipeline. The flits will then try to obtain productive output ports again.

Design choices: There are several key design choices in a side-buffered design. First, the choice of which deflected flits to place in the side-buffer has important implications for performance and correctness. For this reason, the buffer insertion policy takes characteristics of the deflected flits such as whether they are locally-destined, and whether they are currently golden, into account. Second, the number of flits which are placed into the buffer can be tuned to either aggressively buffer all deflected flits, or buffer only a subset of deflected flits, every cycle. For hardware simplicity, and to avoid a large performance impact in the worst case, the MinBD design chooses at most one flit per cycle to buffer. Finally, the design has explicit functionality to guarantee livelock freedom, as we discuss in §4.6.

4.4 Side-Buffer Microarchitecture and Policies

Here we describe how the microarchitecture of the router we have introduced in the previous subsection is extended with a side-buffer. The router pipeline for this design, based on the prior router design CHIPPER [9], is shown in Fig. 4. We will describe each part in turn.

The side-buffer interfaces to the router pipeline in both the inject/eject and permute stages. First, it removes some deflected flits from the pipeline after the permutation network assigns output ports, because the router only knows at this point which flits are deflected. Second, it eventually re-injects the flits that it buffers in the eject/inject stage, using a second instance of the injector that is placed before the ordinary injector (to give priority network access to buffered flits over new traffic). Finally, a "redirection" block is placed in the pipeline preceding the re-injection block in order to provide livelock-free delivery for buffered flits; this is discussed in more detail in §4.6.

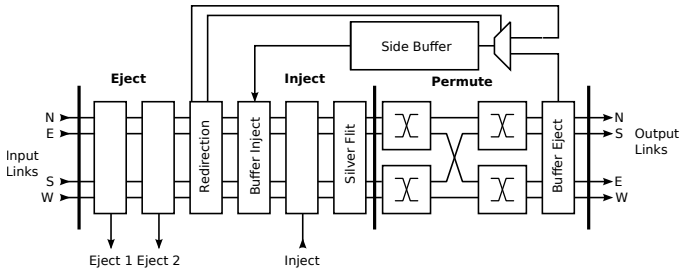


Figure 4: Router pipeline with side-buffer integration.

Algorithms 2 and 3 describe the side-buffer behavior in both the arbitration stage (where flits are removed from the pipeline and inserted into the buffer) and eject/inject stage (where flits are ejected from the buffer and re-injected into the router). The buffer-insertion stage at the end of the pipeline picks at most one flit per cycle that would have been deflected, and buffers that flit instead. Two types of flits are never buffered, however. First, flits that are currently golden (according to the Golden Packet scheme) are never buffered, because doing so would interfere with the livelock-freedom guarantee, as we describe below. Second, flits that are addressed to the local node, but were not able to eject (e.g., when more than two flits arrived at their common destination simultaneously), are never buffered: in such a case, deflection is much less costly than buffering, because the flit is usually able to return immediately to its destination and eject successfully.

In the injection stage, the router implements injection from the side-buffer much as it implements injection from the local node: flits can be re-injected from the side-buffer whenever a free slot is available. Flit re-injection from the side-buffer occurs earlier in the pipeline than injection from the local node; hence, free slots that are present in the network will first be filled by buffered flits, which are already present in the network, and only after then by new traffic. This implicit prioritization of injection helps to ensure forward progress.

Note that because flits that have been side-buffered must eventually be re-injected, and because this re-injection requires free slots in the network, there is no deterministic time bound for eventual delivery on any flit that is buffered in the scheme that we have described so far. This re-injection *starvation* problem, in which a buffered flit may never re-enter the router pipeline, is addressed by *buffer purging*, which we now introduce.

Algorithm 2 Insertion into the Side-Buffer

At each cycle, in arbitration stage:
Perform ordinary bufferless deflection routing
for each deflected flit f_i at output of arbitration stage **do**
  if not currently purging buffer **and** f_i is not golden **and** f_i is not addressed to current node **then**
    mark f_i as buffer-eligible
  end if
end for
if at least one flit is buffer-eligible **then**
  randomly pick one buffer-eligible flit f_{buf}
  remove f_{buf} from router pipeline and place into side-buffer FIFO
end if

4.5 Avoiding Side-Buffer Starvation: Buffer Purging

The side-buffer re-injects flits into the router pipeline in the same way that the router injects new traffic in to the network. A "free slot," or cycle where no input flit is present, is necessary for a flit to be injected or re-injected. If no such free slots arrive – if the input links are continuously receiving flits from neighbor routers – then the flits in the side-buffer may be stuck indefinitely. In order to address this side-buffer starvation problem, we implement an explicit mechanism to force flits out of the buffer, which we call *buffer purging*. The *key idea* of router operation during a buffer purge is that free slots to allow re-injection are *created* by forcing an input flit directly into the buffer, and injecting a flit currently in the buffer in the input flit's place. The router thus allows older flits to make forward progress through the network in exchange for buffering newer flits.

This operation is made more clear by examining the router pipeline in Fig. 4 and side-buffering Algorithms 2 and 3.

Algorithm 3 Ejection from the Side Buffer

```
At each cycle, in eject/inject stage:
if currently purging buffer then
    if side-buffer is not empty then
        select one flit after eject (before redirection) stage, send to side-buffer tail
        take a flit from side-buffer head, re-inject into pipeline
    else
        complete side-buffer purge
    end if
else
    if side-buffer is not empty and at least one flit slot is available after ejector then
        take a flit from side-buffer head and place in router pipeline
    end if
end if
```

When a buffer purge is active, one randomly-chosen flit among all flits present at the router inputs is sent directly to a MUX at the tail of the side-buffer FIFO. This flit is inserted into the side-buffer. As a consequence, a free slot is now present in the router pipeline. This free slot is then immediately used to re-inject the head flit from the side-buffer. In essence, the randomly-chosen input is made to "flow through" the side-buffer FIFO while buffer purging is active, so that the buffered flits are forced back into the network. Finally, at the ordinary side-buffer insertion stage (post-arbitration), the only necessary change is to inhibit side-buffering of deflected flits when a purge is active, because the buffer is receiving flits directly from a router input.

Buffer purging is triggered after injection from the side-buffer into the router is starved for $C_{threshold}$ cycles. That is, when the side-buffer has at least one flit, and the buffer has not re-injected any flits after a certain threshold, the router enters buffer-purging mode. This mode is active for one cycle, in order to purge the head flit from the buffer. This behavior is described by Algorithm 4. In the worst case (when the only buffer forward progress occurs because of purging), for a buffer of N flits, the tail flit of the side-buffer might be stuck in the buffer for $N * C_{threshold}$ cycles. However, such a case almost never occurs. Nevertheless, this deterministic bound is significant because it ensures forward progress, no matter how congested the network becomes.

Finally, we note that in the previous discussion, we have not yet addressed the case when flits are golden. The Golden Packet scheme introduced by CHIPPER [9] and extended in §4.2 introduces a priority level of "golden," given to one packet in the system at a time to ensure the delivery of that one packet. The flits of this packet, "golden flits," must be prioritized over all other flits. If such golden flits are forced into the side-buffer in order to allow the buffer's head flit to make forward progress, we might introduce a priority inversion, and destroy the end-to-end delivery guarantee of Golden Packet. We now address this interaction.

Algorithm 4 Buffer Purging to Prevent Buffer Starvation

```
Initially:
C_blocked ← 0
At each cycle:
if The side-buffer is not empty and re-injection is blocked then
    if C_blocked < C_threshold then
        C_blocked ← C_blocked + 1
    else
        C_blocked ← 0
        enter buffer-purge mode for one cycle
    end if
else
    C_blocked ← 0
end if
```

4.6 Livelock Freedom with Side-Buffering and Golden Packet

We must ensure that the *livelock freedom* guarantee provided by Golden Packet is upheld when side-buffers are added to the network. The fundamental problem that might occur is that a *golden flit* might become stuck in a buffer. If this were to occur, then the livelock freedom provided by Golden Packet would break down, because the Golden Packet would no longer be delivered within the golden epoch.

However, we argue that the failure to deliver a Golden Packet can never occur even with side-buffering implemented. This is because of two key principles. First, a flit that is *currently* golden is never placed in the side-buffer when deflected; instead, it is allowed to continue out of the router. (As noted above, a deflected golden flit is very rare in any case, because a golden flit can only be forced into a deflection by another golden flit from the Golden Packet.) Second, flits that *become* golden while side-buffered must be purged out of the buffer. But by choosing the starvation threshold for buffer purging and the golden epoch length appropriately with respect to each other, we can ensure that forward progress occurs.

In particular, we choose the purging threshold and golden epoch length so that in one golden epoch, at least one golden flit (of all flits in the Golden Packet) that is not buffered, and at least one golden flit that is buffered, will be delivered. (Note that the original Golden Packet scheme guaranteed only that at least one flit of the Golden Packet would be delivered.) Such a guarantee is necessary to ensure that a given packet will eventually be delivered, because at the start of a golden epoch for this packet, any of the packet's flits may be buffered or not buffered. If the packet becomes golden enough times, all of its flits will eventually be delivered by this guarantee.

To provide delivery for one non-buffered golden flit and one buffered golden flit, the golden epoch is chosen so that all buffered flits must leave buffers, with enough additional time that any golden flit forced out of a buffer will then be delivered. (If this additional time were not added, a golden flit might be purged out of a buffer, only to be immediately re-buffered when the golden epoch ended.) Note from the previous subsection that for N_{flit}-flit buffers, N_{flit} buffer-purging activations might be necessary to ensure all flits leave the buffer, and each activation occurs after $C_{threshold}$ re-injection starvation cycles. Thus, after $N_{flit} * C_{threshold}$ cycles, all flits in buffers must have left the buffers. Then, for a network with a maximum hop distance (diameter) of D and per-hop time of C_{hop} cycles, $D * C_{hop}$ cycles ensure that one golden flit is delivered. Assuming that this latter bound is lower than the buffer-purge bound, at least one un-buffered golden flit will be delivered while any buffered golden flits are purging, so we need only to add additional time to ensure a purged golden flit is also delivered. We thus must choose a golden epoch at least as long as $N_{flit} * C_{threshold} + D * C_{hop}$ cycles. In practice, for our parameters of $N_{flit} = 16$, $C_{hop} = 3$ cycles, and $D = 6$ (for a 4x4 network), we choose $C_{threshold} = 2$ cycles and a golden epoch length of 64 cycles.

5 Methodology

To obtain application-level performance as well as network performance results, we use a cycle-accurate 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 running over the modeled network-on-chip. The CPU cores model stalls, and interact with the caches and network in a closed-loop way. Both the modeled CPU cores and the modeled network routers are cycle-accurate, and are assumed to run in a common clock domain. The instruction traces are recorded with a Pin-tool [21], sampling representative portions of each benchmark as determined by PinPoints [26]. We find that simulating 25M cycles gives stable results with these traces, and additional studies at alternate application start points indicate that our results are representative. Detailed system parameters are shown in Table 1.

Note that we make use of a *perfect shared cache* to stress the network, as was done in the CHIPPER [9] and BLESS [23] bufferless router evaluations. 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. In a real system, some shared cache requests would miss, and would wait on a memory request to complete. Because we focus on the NoC bottleneck, we are interested in isolating the effects of NoC design from the impact of the particular memory-system design. In any given system configuration, either there exist workloads that stress the network as we do, or else the NoC is overprovisioned and can be scaled down. Regardless of specific parameters, this methodology allows us to study fundamental network capacity. .

Application Workloads: We use SPEC CPU2006 [32] benchmarks, and compose multiprogrammed workloads from many single-threaded benchmark instances that run independently. Note that we use multiprogrammed workloads for two reasons. First, multiprogrammed workloads are representative of a large class of real-world uses for large CMPs, such as virtual machine aggregation or other shared server use in datacenters. Second, the focus of this work is on network-level optimizations rather than application-aware mechanisms. Thus, improvements that we show with these workloads are due to fundamental NoC enhancements, and will be applicable to any workload that is network-intensive.

To form each multiprogrammed workload, we randomly choose 64 applications from three pools of CPU2006

CPU cores	Out-of-order, 3-wide issue and retire (1 memory op/cycle), 16 MSHRs
L1 caches	64 KB, 4-way associative, 32-byte blocks
L2 (shared) cache	perfect (always hits) to stress the network; in mapping study, 512 KB/slice with one slice per network node, 16-way
Shared cache mapping	Baseline static-striped across all slices
Cache coherence scheme	Directory-based, perfect directory
Data packet sizes	1-flit request packets, 4-flit data packets
Network Links	1-cycle latency, 2.5mm, 128 bits wide
Baseline bufferless router	CHIPPER [9], 2-cycle latency
Baseline buffered router	8 VCs, 8 flits/VC, 2-cycle latency, buffer bypassing [37]
MinBD buffer purge threshold	2 cycles
MinBD side-buffer size	16 flits

Table 1: System parameters.

benchmarks, as follows. First, we split the CPU2006 benchmarks into "low" (L), "medium" (M), and "high" (H) network-intensity categories based on L1 cache-miss intensity (L1 cache miss rate correlates to network injection rate, since every miss generates network traffic). Applications with less than 5 misses per thousand instructions (MPKI) are categorized Low, applications with between 5 and 50 MPKI are classed Medium, and applications with MPKI greater than 50 are considered High-intensity.

Th build a workload, we determine an "intensity mix" as a subset of the pools, and then randomly pick a certain number of applications from each pool. For example, for a "HL" (high-low) workload, we pick 32 applications from the High category and 32 from the Low category. For the purposes of this work, because we focus on the *performance gap* between bufferless and buffered design that occurs at high network load, we only make use of workloads that draw from (at least) the High category. We evaluate with 15 randomly-constructed workloads from each of the following mixes: High-Low (HL), High-Medium (HM), High-Medium-Low (HML), and all-High (H). Unless otherwise specified, any results that summarize application performance are averages of performance across all 60 workloads.

Synthetic-Traffic Workloads: We evaluate networks with three synthetic traffic patterns: uniform-random, bit-complement, and transpose traffic [6]. For each type of traffic, we sweep network injection rate per node from zero to network saturation, and show network latency as a function of this rate.

Performance Metrics: To measure system performance, we use the well-known *Weighted Speedup* metric [31]: $WS = \sum_{i=1}^{N} \frac{IPC_i^{shared}}{IPC_i^{alone}}$. All IPC_i^{alone} values are measured on the baseline bufferless network. Weighted speedup correlates to system throughput [8] and is thus a good general metric for multiprogrammed workloads. To show that our conclusions are robust to the metrics used, we also present average IPC and harmonic speedup [8], which is shown to be the inverse of average normalized turnaround time. Due to space constraints, we present our main result plots only with weighted speedup.

Power Modeling: We use ORION 2.0 [36] and Verilog models synthesized with a commercial 65nm design library to model the hardware for all network designs. For all routers except the baseline buffered router, we build custom Verilog models of control logic, and add in the datapath area and power using ORION models for crossbars and buffers (in the case of MinBD). Note that CHIPPER and MinBD do not use a conventional crossbar, but rather, decouple the datapath into a permutation network. The datapath for each block in the arbiter permutation network is modeled as a 2x2 crossbar. In all other cases, 5x5 or 5x6 crossbars (for dual-width ejection) are modeled as appropriate. Energy per flit is computed by combining Verilog simulations of control logic with datapath energy (link traversal, buffer read/write, etc.). Our cycle-accurate simulator accumulates event counts, which are then used to weight these energy values appropriately, producing average network power. Note that we report average power rather than energy to be consistent with performance (execution speed), which is also a rate.

6 Evaluation

We now evaluate MinBD relative to baseline bufferless and buffered networks, showing application performance, network-level performance, and power and area resulting from a hardware model. These results demonstrate that MinBD attains nearly the performance of the conventional buffered router baseline, within 4.6% in weighted speedup across a set of 60 network-intensive workloads on a 4x4-mesh CMP. In addition, our design adds only a small amount of power and area overhead to the baseline bufferless deflection network, CHIPPER, preserving the energy-efficiency

appeal of such designs.

6.1 Application Performance and Network Power

Fig. 5 shows the performance (measured as weighted speedup) and average network power of the baseline CHIPPER [9] and buffered designs, as well as several of this work's contributions. Results are presented in five bar-groups, each of which corresponds to one workload-intensity category of 15 workloads. The rightmost bar-group in each plot gives overall averages.

From left to right, the bars in the plot show CHIPPER (the bufferless deflection router), CHIPPER with the dual-width ejection enhancement (§4.1), CHIPPER with both the ejection enhancement and the silver-flit prioritization scheme (§4.2), which we call "MinBD-Lite" to indicate that this is MinBD without a side-buffer, then MinBD itself, followed by the buffered-network baseline, the buffered network with the dual-width ejection enhancement for fair comparison, and a buffered network with total buffer size equivalent to MinBD's side-buffer.

Performance: First, the results demonstrate the performance gap between bufferless and buffered designs that we demonstrated in Fig. 2: on average over all workloads, CHIPPER (which we take as our bufferless baseline) performs 12.1% worse than the buffered baseline. In all-High-intensity workloads (rightmost category), this gap increases to 23.1% on average. Adding a dual-width ejection datapath to both the bufferless design and to the baseline buffered reference point increases performance in both designs, but benefits the bufferless design more: baseline CHIPPER with dual-ejection has 6.1% better performance than baseline CHIPPER alone, whereas a buffered network with dual-ejection benefits only 2.1%. Hence, the gap shrinks to only 8.6% degradation on average from buffered to bufferless. We take this dual-ejection buffered design as our buffered baseline for the remainder of the comparisons, for fairness. Next, adding the silver-flit prioritization scheme enhances performance by 1.0% on average on top of dual-ejection, reducing the gap between buffered and bufferless to 7.7%. We refer to this design point as "MinBD-Lite", because it is a purely-bufferless variant of MinBD, without the side-buffer. Nevertheless, it closes a significant portion of the performance gap. We conclude that with simple ejection and arbitration logic changes, significant inefficiencies can be eliminated for a relatively large gain.

Next, when we add the side-buffer, we obtain the full MinBD design, shown as the fourth bar in each group. As shown, this design degrades performance from the buffered baseline (with dual ejection for fairness) by only 4.6% on average. If we instead compare to the baseline buffered design, without the ejection-width modification, the performance of MinBD comes within 2.7% of the buffered baseline. Thus, we conclude that MinBD nearly closes the performance gap between conventional buffered designs and deflection-routed designs, and is effective for network-intensive workloads.

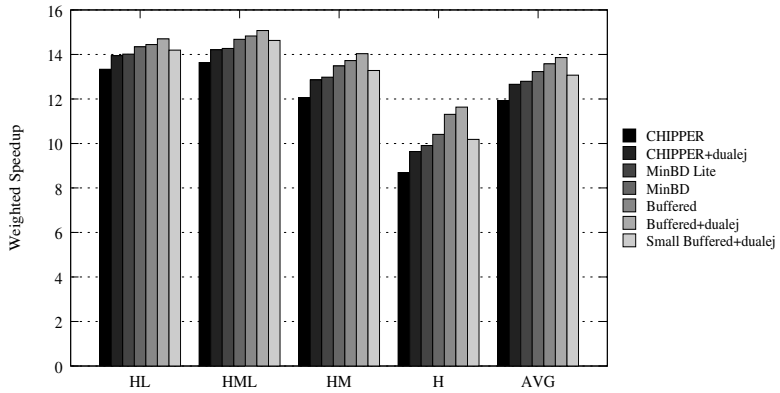


Figure 5: MinBD performance results relative to the baseline CHIPPER design, as well as the buffered-router baseline, in a 4x4 CMP.

Finally, we compare a variant of the buffered baseline that has its buffer size reduced to be equivalent to MinBD's side-buffer: the side-buffer holds 16 flits in our configuration, so we modify the conventional buffered router to use only 4 flits of buffering per physical channel (excluding the local-injection channel), for 16 flits total in 4 PCs. We retain dual ejections for fairness. This configuration (labeled "Small buffered+dualej" in Fig. 5) performs 5.7% worse than the main buffered-router configuration, and 1.1% worse than MinBD. Hence, we conclude that MinBD makes

more efficient use of a given buffering space than the equivalent conventional input-buffered virtual channel router. This is because the router buffers only a portion of the traffic that passes through it, and relies on deflection to handle the remainder of contention; as these results show, the combined deflection/buffering approach makes more efficient use of total network capacity than conventional buffering alone.

Alternate performance metrics: To ensure that our results are robust against choice of metric, we also examine average IPC and harmonic speedup [8]. MinBD degrades system IPC from a buffered network by 2.5%, and degrades harmonic speed by 2.6%, in a 4x4 network. In general, trends are similar in these metrics, indicating that our conclusions are not dependent on weighted speedup as a metric.

Power: Fig. 6 shows average network power for the same sets of workloads shown in Fig. 5. Overall, MinBD reduces average network power from the buffered baseline (single ejection width) by 70%, and from the dual-ejection width buffered baseline (whose performance we showed for fair comparison above) by 72%. This reduction is due to the removal of the large buffers, and simplification of the crossbar. In addition, as we showed with the "Small Buffered" performance comparison above, these large buffers are necessary for the conventional buffered design to perform well; when its buffers are reduced to the size of the buffers in MinBD, the conventional buffered design's performance is worse. Hence this power reduction is not inflated, but is a fair comparison.

MinBD also reduces total average network power from the bufferless (CHIPPER) baseline, primarily due to reduced deflections, which lead to lower dynamic power. Part of this effect is captured simply by allowing dual ejections; from the first bar (CHIPPER) to the second bar (CHIPPER with dual ejections) in the plot, power is reduced by 7.4% on average. Adding the silver-flit mechanism does not impact power significantly. However, adding the side-buffer, which completes the final MinBD design, reduces power further, to 23.3% less than baseline CHIPPER. This reduction is despite the additional static power that the side-buffer consumes: the buffer reduces deflection rate and hence network utilization significantly. Overall, we conclude MinBD consumes the least average power of all evaluated designs.

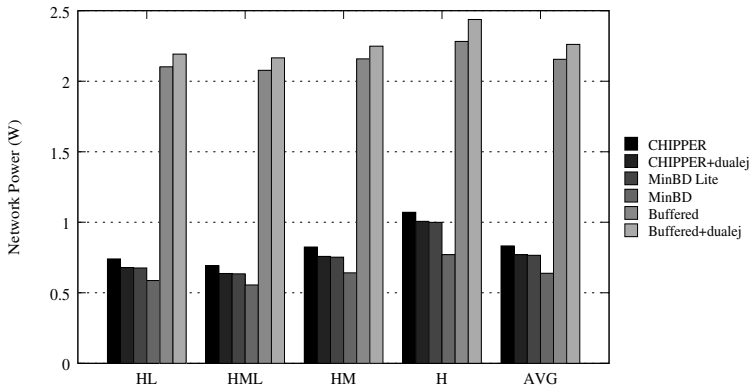
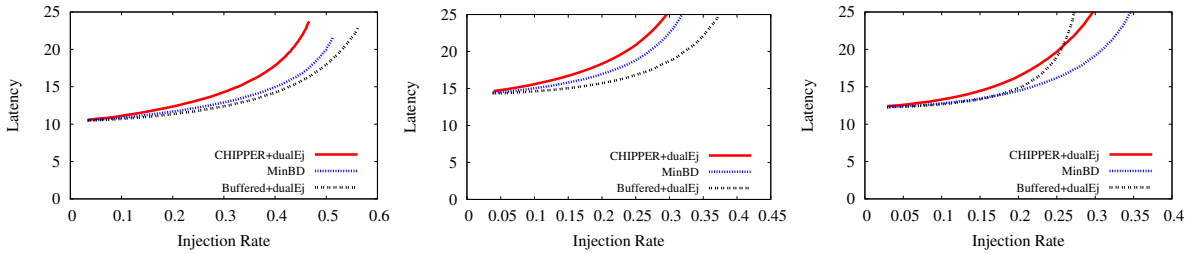


Figure 6: MinBD network power, and baseline CHIPPER and buffered network power, in a 4x4 CMP.

6.2 Network Performance

We study the network-level performance of MinBD and baseline designs by subjecting each network to three synthetic traffic patterns [6]: uniform random traffic, bit-complement (in which each node takes the bit-complement of its own address and sends packets only to that address), and transpose (in which each node sends traffic to its corresponding node reflected across the diagonal). Fig. 7 shows that MinBD performs better than the bufferless baseline in all cases, with a higher saturation point. In uniform-random traffic, which is the best case (because it spreads load evenly across the network), MinBD performs almost identically to the baseline buffered network, despite having much less buffering space. Bit-complement traffic presents a more challenging workload for MinBD, because the traffic pattern is not evenly distributed; the buffers in the baseline buffered network help to handle the load with less contention than either MinBD or baseline CHIPPER, which both rely on deflection to spread load. Interestingly, in the last case, for the Transpose traffic pattern, MinBD improves on the bufferless baselines as before, but the baseline buffered design saturates earlier than baseline CHIPPER. This is because the transpose pattern forces all traffic across the mesh's diagonal, and the non-adaptive dimension-ordered routing used in the baseline buffered design does not deal well with



(a) Uniform Random (b) Bit-Complement (c) Transpose

Figure 7: Latency curves as a function of injection rate for three synthetic traffic patterns in baseline bufferless, MinBD, and baseline buffered designs in a 4x4-mesh configuration.

this pattern: congestion occurs near the ends of the diagonal (corners of the mesh), where more traffic is focused in its fixed X-Y paths. In contrast, deflection in MinBD allows more traffic to make use of links in the center of the chip. Hence, MinBD is able to use its side-buffer to gain an early advantage over baseline purely bufferless designs, but is also able to beat non-adaptive buffered designs at high load due to deflection routing.

6.3 Router Area

Next, we study the impact of our changes on router area. Table 2 shows router area, normalized to the CHIPPER baseline, for CHIPPER, conventional buffered designs (with and without the dual-ejection change for fair comparison) and MinBD. The increased router area from CHIPPER to MinBD (13%) is due to adding the side-buffer (sized for 16 flits) as well as control logic for dual-ejection and arbitration enhancements. In contrast, the conventional virtual-channel buffered router consumes more than 3x the router area of CHIPPER in our model (and the buffers cannot be shrunk to reduce this figure, because performance will be reduced, as our earlier evaluations show). Hence, we conclude that the control logic required for MinBD does not reduce the router-area advantages seen in the baseline bufferless design, and furthermore, that the small side-buffer has a minimal impact on area relative to the full-size buffers in a conventional router.

Router Design	CHIPPER	MinBD	Conventional Buffered (Dual-Ej)	Small Buffered (Dual-Ej)
Normalized Area	1.00	1.13	4.01	3.82

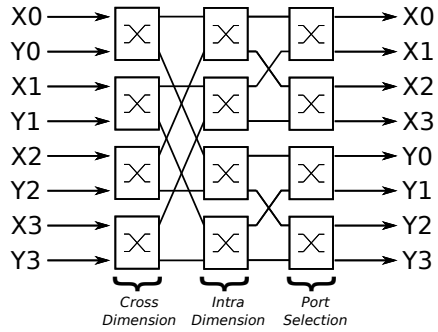
Table 2: Normalized router area comparison for baseline bufferless (CHIPPER) and buffered router designs, compared to MinBD.

6.4 Sensitivity to Alternate Topologies and Alternate Data Mapping

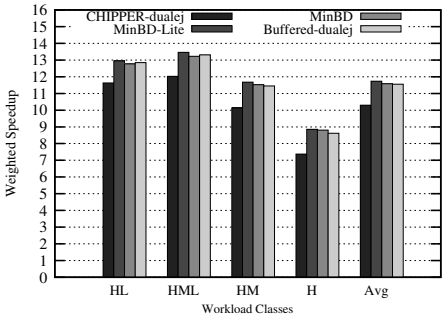
We have observed that MinBD closes much of the performance gap between bufferless and buffered networks in conventional mesh topologies, and assuming a static striped mapping of shared cache data to shared cache slices. We now show that the gains obtained by the new router design are not limited to this particular system design, but are also applicable to other (more dense) tiled-CMP topologies as well as systems with locality-aware data mapping schemes.

6.4.1 Applicability to Dense Topologies: Flattened Butterfly

Because wires are relatively cheap in on-chip networks, some work proposes denser topologies – i.e., more links for a given number of nodes – in order to provide lower network latency. One such proposal, which we evaluate here, is Flattened Butterfly [16, 17]. This topology connects a given router to all of the routers in its row and column (assuming a tiled CMP), so that any node is only two hops away from any other node. Such a design was shown to offer performance gains over a baseline mesh, because of the lower average hop count. We wish to show that when such a topology is used to reduce network load, MinBD is (i) still applicable, and able to be adapted to the new topology, and (ii) is still able to offer performance gains.



(a) CHIPPER permutation network for flattened butterfly.



(b) Flattened butterfly performance for baseline bufferless (CHIPPER), MinBD-Lite (no side-buffer), and conventional buffered networks in a 4x4 tiled CMP.

Figure 8: MinBD applied to Flattened Butterfly [17, 16] dense-topology networks.

First, we need to account for a higher router radix (number of input and output links). We extend the permutation network-based arbiter design used by both CHIPPER and MinBD accordingly. Fig. 8a shows one possible design, for a 4x4 flattened-butterfly router. Although a flattened-butterfly router in a 4x4 network has a radix of 6 (i.e., has 6 inputs and 6 outputs), we build an 8-input, 8-output permutation network, because power-of-two permutation networks lead to much simpler steering functions (i.e., no dead ends due to incomplete trees). After connecting six of the inputs and outputs to all neighbor routers in the given router's row and column, the two remaining inputs and outputs are connected so that each of the outputs loops directly back to one of the inputs. Steering functions never intend to steer a flit toward such an output, but deflected flits may be sent to these outputs, at which point they are sent back through the arbiter again.

We evaluate flattened-butterfly versions of the baseline conventional buffered router, MinBD, MinBD-Lite, and CHIPPER, with results shown in Fig. 8b. All system parameters remain the same, except that router latency is increased by one cycle in order to account for the higher router radix. This change is made across all routers. We show dual-ejection versions of CHIPPER and conventional buffered baselines, as before. As is shown, MinBD-Lite is effective in improving application performance from a CHIPPER baseline, increasing weighted speedup by 14.0%. In fact, this network design performs slightly better than the buffered baseline, with a weighted speedup 1.6% higher on average. This is due to the adaptivity that deflection routing affords to MinBD-Lite. Interestingly, when the side-buffer is added to form MinBD proper, performance degrades by 1.3% (to only 12.5% above CHIPPER). This indicates that in a dense-topology network, the additional links are effective at providing effective buffering capacity to a deflection router, and there's less penalty to deflect flits than buffering them. Thus, MinBD-Lite is both cheaper in area and power (because it does not require a side-buffer) and provides better performance than MinBD. We conclude that MinBD-Lite is the highest-performing router design that we evaluate in the flattened-butterfly network, and is a compelling alternative to conventional VC-buffered and previously-proposed bufferless deflection routers in dense topologies.

6.4.2 Impact on Systems with Locality-aware Data Mapping

Finally, we evaluate the impact on systems where locality-aware data mapping of cached data to shared cache slices is implemented to reduce network load. By using a locality-aware data mapping, the average packet traversal distance for L1 cache miss requests and L2 data replies are significantly reduced because shared cache blocks are mapped to cache slices close to the nodes that use the cached data. Hence, the network load is significantly reduced, increasing performance. We aim to show that MinBD still attains significant performance gains in such a configuration, or in cases where there is little performance gap to reduce, it can still lead to significant power and area reductions.

We assume a mapping mechanism based on the distance-aware last-level cache scheme CloudCache [20]. CloudCache allows each node in a CMP system to expand or shrink cache capacity depending on its cache utilization, and places a node's data into nearby caches when possible.

Methodology remains the same, with a few exceptions. First, we no longer assume a perfect last-level cache, because the mechanism requires a realistic shared-cache model. However, we model zero-latency memory, with

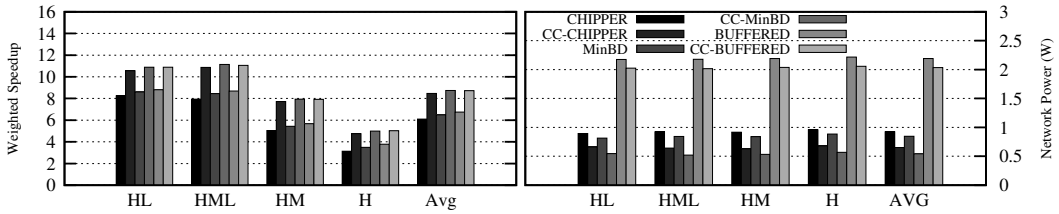


Figure 9: Performance and network power for baseline bufferless (CHIPPER), MinBD, and buffered configurations with realistic memory controllers, and incorporating a locality-aware last-level-cache mapping scheme.

memory controllers at each corner of the mesh: that is, request packets are still sent to memory controllers when a last-level cache miss occurs, but the response packet is sent immediately. This methodology ensures that we study the NoC bottleneck and not a memory bottleneck, as we ensured previously with the perfect last-level cache.

Application performance and network power are shown in Fig. 9. We show baseline CHIPPER, MinBD, and conventional buffered networks; for each configuration, the "CC-" variant uses the CloudCache-based data mapping, while the baseline variant does not use this mapping, but is otherwise identical (including the memory controller configuration). Two major conclusions are visible in this data. First, using the CloudCache-like data mapping, all three networks perform nearly equally on average (0.07% difference between MinBD and buffered; 3.3% reduction in CHIPPER relative to buffered). However, MinBD retains its reduced power relative to prior bufferless router designs (by 16.9%), because lower deflection rate leads to lower dynamic power, and its significantly reduced power with respect to conventional input-buffered routers (by 73.3%). Hence, we conclude that our new router design gives the lowest overall power, with performance nearly identical to the conventional buffered baseline design, when a locality-aware data-mapping is used.

7 Related Work

To our knowledge, MinBD is the first router design that takes a bufferless deflection router and uses a side-buffer to buffer flits that would have been deflected. Our work is also the first, to our knowledge, to observe the ejection bottleneck in bufferless deflection routing design. Other routers have combined deflection routing with buffers, but either primarily use buffers and use deflection as a secondary mechanism, or else switch buffers on and off as required without fundamentally addressing inefficiencies in the bufferless mode.

Buffered NoCs that also use deflection: Several proposed interconnect routers that primarily route using buffers and flow control also use deflection routing as an escape or secondary mechanism under high load. The Chaos Router [18] implements a deflection queue into which flits are placed when they experience output port contention. The Rotary Router [1] allows flits to leave the router's inner ring on an incorrect output port after the flits circulate the ring enough times, in order to ensure forward progress. In both cases, deflection is used as an escape mechanism rather than the primary means of conveying flits. All traffic is buffered by default in both of these designs. In contrast, MinBD primarily deflects flits, and buffers only a limited portion of the network traffic (i.e., only flits that are deflected, and then meet certain other conditions at a given router).

Other bufferless designs: Several prior works have proposed purely bufferless router designs [9, 23, 35]. CHIPPER [9] is the baseline for our work. BLESS [23], another bufferless deflection network, served as the baseline for CHIPPER and uses a more complex deflection routing algorithm that leads to more complex hardware. Because later works showed BLESS to be impractical to implement in hardware [9, 11, 22], we do not compare to it in this work. Other purely bufferless networks drop flits upon contention, rather than deflecting [10, 11]. While all of these designs are viable, we set out with the explicit goal to match the performance of a conventional buffered network, and are willing to use minimal buffering when necessary. Some earlier large multiprocessor interconnects, such as the interconnects of the HEP [30] and Connection Machine [12] multiprocessors, also used deflection routing. The HEP router in particular combined some buffer space with deflection routing [29]. However, the details of these routers are not well-known, and their operating conditions (large off-chip networks) are significantly different than those that modern on-chip interconnects experience.

Optical networks often make use of deflection-routing schemes because of the difficulty of buffering in the optical domain [4, 38]. Although the fundamental deflection principles are the same, the tradeoffs and details of implementa-

tion in optical networks are significantly different than in the on-chip domain.

Hybrid buffered-bufferless NoCs: AFC [15] combines a bufferless deflection router based on BLESS [23] with buffers, and switches between bufferless deflection routing and conventional input-buffered routing based on network load at each router. While this scheme gives the performance of buffered routing in the highest-load case, with the energy efficiency of bufferless routing in the low-load cases, it does not fundamentally tackle the inefficiencies of bufferless routing, simply avoiding its high-load inefficiency by enabling buffers. As a result, its energy in the high-load case is as high as in a buffered router, and it requires aggressive power gating to achieve the energy-efficiency of bufferless routers in the low-load case. 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 eliminates the need to include large buffers and the associated buffered-mode control logic, instead requiring only the smaller side-buffer. MinBD also removes the dependence on efficient buffer power-gating that AFC assumes to be present for energy-efficient operation at low loads.

8 Conclusion

In this paper, we present MinBD, a new minimally-buffered deflection router design. We develop three key insights. First, at high load, a bufferless deflection network suffers from an *ejection bottleneck*. Allowing more than one flit per cycle to eject at each router has a significant impact on improving performance. Second, *inefficient routing arbitration* can lead to a higher-than-necessary deflection rate, and a simple change to the router design reduces this problem. Taken together, these two simple yet effective mechanisms show that careful attention to bottlenecks and inefficiencies in a bufferless router design can eliminate a significant fraction of the performance degradation relative to buffered networks. Finally, we show that by adding a small *side buffer* to the router, and buffering some flits rather than deflecting them, most of the remaining performance gap relative to a traditional input-buffered router is removed. We show that this side-buffer makes more efficient use of a given buffer space than input-buffered routers do. Together, these three mechanisms allow MinBD to perform within 4.6% of a conventional buffered router (by weighted speedup) in a 4x4 network on a set of 60 network-intensive workloads, while reducing network power by 72.5% on average and shrinking router area by 80.0%. Hence, we conclude that MinBD is a compelling energy- and area-efficient yet high-performance router design for future on-chip networks.

9 Acknowledgments

We thank members of CALCM (Computer Architecture Lab at Carnegie Mellon) for their insightful feedback and contribution of ideas to this work. We gratefully acknowledge the support of NSF CAREER Award CCF-0953246, Gigascale Systems Research Center, Intel Corporation ARO Memory Hierarchy Program, and Carnegie Mellon CyLab. We also acknowledge equipment and gift support from Intel.

References

[1] P. Abad et al. Rotary router: an efficient architecture for cmp interconnection networks. *ISCA-34*, 2007. 15

[2] P. Baran. On distributed communications networks. *IEEE Trans. on Comm.*, 1964. 3

[3] L. A. Barroso et al. Piranha: a scalable architecture based on single-chip multiprocessing. *ISCA-27*, 2000. 2

[4] S. Bregni and A. Pattavina. Performance evaluation of deflection routing in optical IP packet-switched networks. *Cluster Computing*, 2004. 15

[5] D. E. Culler et al. *Parallel Computer Architecture: A Hardware/Software Approach*. Morgan Kaufmann, 1999. 2

[6] W. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann, 2004. 2, 10, 12

[7] W. J. Dally and B. Towles. Route packets, not wires: On-chip interconnection networks. *DAC-38*, 2001. 1

[8] S. Eyerman and L. Eeckhout. System-level performance metrics for multiprogram workloads. *IEEE Micro*, 28:42–53, May 2008. 3, 10, 12

[9] C. Fallin, C. Craik, and O. Mutlu. CHIPPER: A low-complexity bufferless deflection router. *HPCA-17*, 2011. 2, 3, 4, 5, 6, 8, 9, 10, 11, 15

[10] C. Gómez et al. Reducing packet dropping in a bufferless noc. *Euro-Par-14*, 2008. 15

[11] M. Hayenga, N. E. Jerger, and M. Lipasti. Scarab: A single cycle adaptive routing and bufferless network. *MICRO-42*, 2009. 3, 15

[12] W. Hillis. *The Connection Machine*. MIT Press, 1989. 15

[13] Y. Hoskote et al. A 5-GHz mesh interconnect for a teraflops processor. *IEEE Micro*, 2007. 2

[14] Intel Corporation. Intel details 2011 processor features. http://newsroom.intel.com/community/intel_newsroom/blog/2010/09/13/intel-details-2011-processor-features-offers-stunning-visuals-built-in. 1

[15] S. A. R. Jafri et al. Adaptive flow control for robust performance and energy. *MICRO-43*, 2010. 2, 4, 16

[16] J. Kim, J. Balfour, and W. Dally. Flattened butterfly topology for on-chip networks. *MICRO-40*, 2007. 13, 14

[17] J. Kim and W. Dally. Flattened butterfly: A cost-efficient topology for high-radix networks. *ISCA-34*, 2007. 2, 13, 14

[18] S. Konstantinidou and L. Snyder. Chaos router: architecture and performance. *ISCA-18*, 1991. 15

[19] J. Laudon and D. Lenoski. The SGI Origin: a ccNUMA highly scalable server. *ISCA-24*, 1997. 2

[20] H. Lee, S. Cho, and B. Childers. Cloudcache: Expanding and shrinking private caches. *HPCA-17*, 2011. 14

[21] C.-K. Luk et al. Pin: building customized program analysis tools with dynamic instrumentation. *PLDI*, 2005. 9

[22] G. Michelogiannakis et al. Evaluating bufferless flow-control for on-chip networks. *NOCS*, 2010. 3, 15

[23] T. Moscibroda and O. Mutlu. A case for bufferless routing in on-chip networks. *ISCA-36*, 2009. 2, 3, 9, 15, 16

[24] G. Nychis et al. Congestion control for scalability in bufferless on-chip networks. SAFARI technical report TR-2011-003: http://www.ece.cmu.edu/~safari/tr.html, 2011. 2, 4

[25] G. Nychis, C. Fallin, T. Moscibroda, and O. Mutlu. Next generation on-chip networks: What kind of congestion control do we need? *Hotnets-IX*, 2010. 2, 4

[26] H. Patil et al. Pinpointing representative portions of large Intel Itanium programs with dynamic instrumentation. *MICRO-37*, 2004. 9

[27] D. Pham et al. Overview of the architecture, circuit design, and physical implementation of a first-generation cell processor. *J. Solid-State Circuits*, 41(1):179–196, Jan 2006. 1

[28] L. Seiler et al. Larrabee: a many-core x86 architecture for visual computing. *SIGGRAPH*, 2008. 1

[29] B. Smith. Personal communication. 15

[30] B. Smith. Architecture and applications of the HEP multiprocessor computer system. *SPIE*, 1981. 15

[31] A. Snavely and D. M. Tullsen. Symbiotic jobscheduling for a simultaneous multithreaded processor. *ASPLOS-9*, 2000. 3, 10

[32] Standard Performance Evaluation Corporation. SPEC CPU2006. http://www.spec.org/cpu2006. 9

[33] M. Taylor, J. Kim, J. Miller, and D. Wentzlaff. The raw microprocessor: A computational fabric for software circuits and general-purpose programs. *IEEE Micro*, Mar 2002. 2

[34] Tilera Corporation. Tilera announces the world's first 100-core processor with the new TILE-Gx family. http://www.tilera.com/news_&_events/press_release_091026.php. 1

[35] S. Tota et al. Implementation analysis of NoC: a MPSoC trace-driven approach. *GLSVLSI-16*, 2006. 15

[36] H. Wang et al. Orion: a power-performance simulator for interconnection networks. *MICRO-35*, 2002. 10

[37] H. Wang, L. Peh, and S. Malik. Power-driven design of router microarchitectures in on-chip networks. *MICRO-36*, 2003. 10

[38] X. Wang et al. Burst optical deflection routing protocol for wavelength routing WDM networks. *SPIE/IEEE Opticom*, 2004. 15

[39] D. Wentzlaff et al. On-chip interconnection architecture of the tile processor. *IEEE Micro*, 27(5):15–31, 2007. 2