Clumsy Flow Control for High-Throughput Bufferless On-Chip Networks

Hanjoon Kim, Yonggon Kim, John Kim
Department of Computer Science, KAIST
{hanj, ilios, jjk12}@kaist.ac.kr

Abstract—Bufferless on-chip networks are an alternative type of on-chip network organization that can improve the cost-efficiency of an on-chip network by removing router input buffers. However, bufferless on-chip network performance degrades at high load because of the increased network contention and large number of deflected packets. The energy benefit of bufferless network is also reduced because of the increased deflection. In this work, we propose a novel flow control for bufferless on-chip networks in high-throughput manycore accelerator architectures to reduce the impact of deflection routing. By using a *clumsy flow control* (CFC), instead of the per-hop flow control that is commonly used in buffered on-chip networks, we are able to reduce the amount of deflection by up to 92% on high-throughput workloads. As a result, on average, CFC can approximately match the performance of a baseline buffered router while reducing the energy consumption by approximately 39%.

Index Terms—on-chip networks, bufferless router, flow control, deflection routing

1 Introduction

With the increasing number of cores, on-chip networks (NoC) are an important aspect of future manycore processor architecture. In this work, we leverage recently proposed bufferless NoC and explore how it can be leveraged in manycore accelerator architectures such as GPGPUs. While bufferless NoC can provide benefits such as lower cost because of no input buffers, one challenge with bufferless NoC is the performance degradation as the network load increases. The manycore accelerator architectures that we focus on require high network throughput. To overcome this, we propose a flow control for bufferless NoC such that it becomes suitable for such high-throughput architectures.

Flow control in interconnection networks defines how the shared network resources are utilized, especially when contention occur for shared resources [4]. A commonly used flow control in NoC is buffered flow control, as buffers decouple the allocation of channel bandwidth resources. For example, if two packets contend for the same output port in the same cycle, one packet is granted access to the output port channel bandwidth while the other packet is temporarily buffered. Buffered flow control needs to communicate the availability of buffers in neighboring routers and credit-based flow control is commonly used in buffered flow control.

Recently, bufferless flow control has been extended to NoC to reduce the cost of the network [11], [8]. Bufferless flow control removes on-chip router buffers, but additional mechanisms are needed when contentions occur. Packets can either be misrouted or be dropped and retransmitted. However, it has been previously shown that a bufferless network results in lower network throughput without input buffers and is not suitable when the network load increases. To overcome the limitations, we propose to introduce flow control into bufferless networks; we refer to this flow control as *clumsy flow control* (CFC). CFC is based on conventional credit-based flow control but since there are no router input buffers, CFC is *destination*-based because the credit information is based on the packet's destination and not on the intermediate per-hop routers. CFC is also an *approximate* or clumsy flow control because exact buffer information is not necessarily required. For example, if more packets are sent than the available buffer space, the other packets will simply be deflected and not necessarily dropped. However, by throttling network traffic, CFC minimizes network congestion and improve overall performance while improving the efficiency of the bufferless NoC.

2 Background

2.1 Methodology

We evaluate bufferless NoC for manycore accelerator architectures such as GPGPU, which is a massively multi-threaded, throughput-oriented architecture. The NoC for such architectures need to support high throughput. The traffic in these architecture follow the many-to-few-to-many [1] traffic pattern – from the many cores to the few memory controllers, and then, back to the many cores. The GPGPU architecture that we assume has two separate traffic classes – request and reply traffic. We use two separate networks in parallel, similar to [1], with one network for request traffic and another network for reply traffic, in order to avoid protocol deadlock. The baseline bufferless router is a deflection-based bufferless router microarchitecture, similar to what was proposed earlier [11].

For synthetic workload, we use a cycle-accurate network simulator [4]; for the different applications, we use the GPGPU-Sim [2] simulator. The configuration for GPGPU-sim is listed in Table 2 and other parameters are similar to what was used in [1] – we use a 6×6 2-D mesh network with staggered memory controller placement and 8 bytes channel size for each network. For the buffered baseline, we assume 8 buffer entries and 4 VCs per port and XY routing. ORION 2.0 [9] was modified to measure the energy consumption of both buffered and bufferless NoC. Applications that we used are listed in Table 1. We selected applications with different characteristics – including memory bandwidth requirements, and ratio of read and write requests – in order to show the impact of bufferless architecture across a wide range of applications.

For long packets consisting of multiple flits [1], a reassembly buffer is needed at the endpoints to re-assemble the packet, because flits can arrive out-of-order. There are two types of long packets – write requests and read reply packets. Similar to prior work [5], we use the MSHR structure as reassembly buffers for read reply packets, while the memory queues at the memory controllers are used for the write request packets. An additional mechanism that resolves deadlock caused by write request packets is needed for correctness. In Section 3.3,

1. Multi-flit packets are truncated into single-flit packets and age-based arbitration is used to avoid livelock.

Posted to the IEEE & CSDL on 7/13/2012
DOI 10.1109/L-CA.2012.22 1556-6056/13/$31.00 © 2013 Published by the IEEE Computer Society

TABLE 1
Applications used

Benchmark	Label	Suite
Fast Walsh Transform	fwt	CUDA SDK [12]
Scalar Product	sp	CUDA SDK [12]
Speckle Reducing Anisotropic Diffusion	srad	Rodinia [3]
Breadth-First Search	bfs	Rodinia [3]
Nearest Neighbor	nn	Rodinia [3]
Back Propagation	bp	Rodinia [3]
Weather Prediction	wp	3rd Party [2]
MUMmerGPU	mum	3rd Party [2]
LIBOR Monte Carlo	lib	3rd Party [2]
BlackScholes	blk	3rd Party [2]
Sparse-matrix/dense-vector multiplication	spmv	Parboil [15]
Fast Fourier Transform	fft	Parboil [15]

TABLE 2
Microarchitecture parameters

Shader Cores	**28**
Max Threads per Shader Core	**1024**
On-Chip Network Topology	**2D Mesh**
Interconnect Flit Size (Bytes)	**8**
Num of Virtual Channels	**4**
Interconnect Virtual Channel Buffer Size (Flits)	**8**
L1 Cache Size / Core	**16KB**
DRAM Chips	**16**
Memory Controllers	**8**
DRAM Chips per MC	**2**
DRAM Queue Size	**32**
Memory Scheduler	FR-FCFS

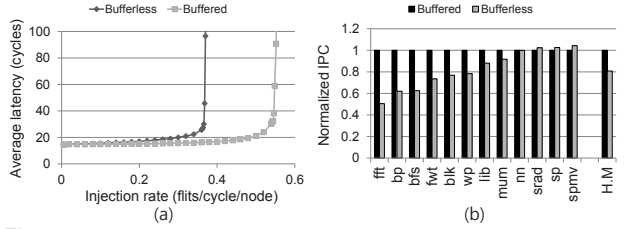

Fig. 1. (a) Latency-throughput curve and (b) performance comparison on CUDA workloads between buffered and bufferless.

we describe the details of how CFC can avoid such multi-flit deadlock.

2.2 Bufferless NoC Limitations

The network-only latency-throughput curves of bufferless and buffered NoC of the 2D mesh network are shown in Figure 1(a) for uniform random traffic. The throughput of the bufferless NoC is limited compared with that of buffered routers, similar to what has been previously observed [11]. The performance of the buffered router will vary depending on the number of virtual channels and the amount of input buffers; however, for bufferless, the throughput is limited as load increases because of the increase in the amount of deflection. The impact of bufferless routers across different applications is shown in Figure 1(b). For a few applications, bufferless NoC actually improves overall performance by a few percent as the deflection routing introduces adaptivity. However, for other workloads, it can result in up to 50% reduction in performance. On average, the bufferless NoC resulted in a 19.2% reduction in performance compared with that of baseline buffered NoC. There is also a very small gain in energy reduction – approximately 5.3% (Figure 3). As a result, baseline bufferless NoC provides very few benefits for these manycore accelerator architectures that have high network load. In the following section, we describe the clumsy flow control that can enable bufferless NoC to be used with minimal impact on overall performance.

3 Clumsy Flow Control (CFC)

3.1 CFC Description

A commonly used flow control to manage buffers in buffered on-chip networks is credit-based flow control. Each upstream router maintains a *credit* that represents the number of unoccupied or free downstream buffer entries. As data flows in one direction, credits flow in the opposite direction as shown in Figure 2(a). A packet is partitioned into one or more *flits* [4] and the width of each buffer entry corresponds to a flit width. Before a flit is transmitted downstream, the appropriate credit count is decremented. Once the flit departs the downstream router, a credit is returned back upstream. As packets traverse each hop, credit-based flow control is used to guarantee that there is buffer space in the downstream router. If there are no credits available, flits are stalled in the current router until a credit is returned. This common flow control is implemented at *per-hop* granularity as credit is transferred between neighboring routers.

Algorithm 1 Bufferless Flow Control at Source.

```
At each shader core i
for each read request destined to MC j do
    if r_ij > 0 then
        inject request;
        r_ij--;
    else
        throttle;
    end if
end for    ▷ similar procedure is done for write requests but credit count w_ij is checked
for each reply from MC j do
    if read reply then
        r_ij++;
    else
        w_ij++;
    end if
end for
```

However, in a bufferless on-chip network without any router input buffers, such per-hop credit-based flow control is not needed as arriving flits are guaranteed to make progress: they either move towards their destination if there is no contention or are deflected if there is contention for the same output. [2] In this work, we propose a flow control for bufferless routers that we refer to as *clumsy flow control* (CFC) – an approximate, destination-based credit-based flow control that throttles network traffic to avoid network congestion and minimize deflection in bufferless NoC. CFC is based on credit-based flow control but instead of per-hop flow control, CFC is *destination*-based as the credits represent buffer availability at the packet destination as shown in Figure 2(b).

The credits in CFC are only maintained by the cores and the credits do not represent local router buffer occupancy but represent the buffer availability at the *destination*. With the traffic from the cores to the memory controllers (MC), the destination buffers that we leverage are the memory request queues at the MCs. Thus, each credit represents the ability for each core to inject another *request* into the network based on the memory queue occupancy. By throttling the memory requests through CFC, the number of in-flight requests are minimized and thus, minimize network congestion and deflection routing. Once requests are injected into the network, the credit count is decremented and when a reply is received from the MCs, the corresponding credit is incremented.

The differences between the two types of flow control are summarized in Table 3. Although credits are used to represent available buffer entries, one key difference is the *accuracy* of the credits. In credit-based flow control, credits are an accurate representation of the buffer space; this guarantees that packets are not dropped in the network. However, in CFC, the credits do not necessarily need to be an *exact* representation of the memory queue buffer space but can be an *approximation* – hence clumsy flow control. If more packets are sent compared to the amount of buffer space available, the extra packets will be deflected – following normal behavior of the bufferless router. In addition, credits in credit-based flow control are explicitly

2. Alternative bufferless router implementation can be done where packets are dropped when contention occur [8]. We focus on the deflection routing bufferless implementation but our proposed mechanism can be extended to drop-retransmission bufferless routers as well.

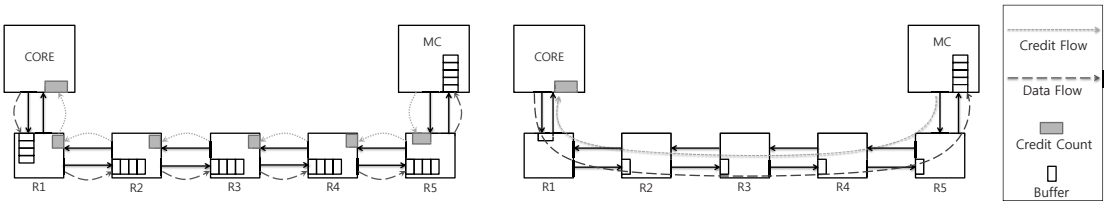

Fig. 2. (a) Credit-based flow control of memory request with buffered network and (b) clumsy flow control (CFC) with bufferless network. For simplicity, only one core and one memory controller (MC) are shown. With more than one MC, each core would have separate credit information for each destination (or MCs) with CFC.

TABLE 3
Comparisons between the two flow controls.

	credit-based flow control	clumsy flow control (CFC)
distance	per-hop	destination
granularity	flits buffer entry	memory queue entry
accuracy	exact : packet is not transmitted if there is no buffer	inexact : packet can still be sent without impacting correctness
credit transmission	separate, dedicated wires[3]	implicit piggybacking

returned to upstream routers, either through dedicated wires or by piggybacking. In comparison, CFC does not require explicit credits to be returned but credits are *implicitly* returned when response packets return from the MC back to the cores.

3.2 Details

Details of CFC are described in Algorithm 1. Each core (i) maintains two credit counts for each memory controller (j), which we represent as r_{ij} and w_{ij} for read and write requests. Initially, $r_{ij} = r$ and $w_{ij} = w \ \forall i, j$, where r and w are the initial credits allocated to each core for read and write requests. For a read request from core i destined to memory controller j, if $r_{ij} > 0$, the core injects the request into the network and decrements r_{ij}. If $r_{ij} = 0$, the requests are throttled until credit becomes available. Once a reply returns from memory controller j back to core i, the appropriate r_{ij} is incremented. For write requests, the corresponding w_{ij} value is used. We denote different implementations of CFC as CFC(r, w). The values of r and w can be any value greater than 0. For smaller values, more throttling is done to minimize the amount of deflection, but smaller values can also limit overall performance. For larger values, there is less throttling but also results in more deflection in the network. As r and w values approaches infinity, CFC corresponds to the baseline bufferless router without any flow control. The impacts of different initial values of credits are evaluated in Section 4.

3.3 Deadlock and Livelock

With deflection routing and bufferless NoC, both livelock and deadlock can be an issue. We avoid livelock by using age-based or oldest-first arbitration [11]. Protocol deadlock is avoided by having separate networks for request and reply (Section 2.1) but another type of deadlock can still occur because of multi-flit packets and the destination re-assembly buffer. If each entry in the destination re-order buffer is partially occupied (i.e., only some of the flits of the packets have arrived), the remaining write request flits need to arrive n order for the system to make progress. However, if the network is full and the remaining write request flits cannot be injected, deadlock will occur. This deadlock is identical to the problem identified earlier as reassembly buffer overflow [5] and occurs because of a lack of flow control in the bufferless NoC. In this work, we leverage CFC in order to avoid such deadlock. With CFC(r,w), if the sum of w values across all the cores is smaller than the write memory queue size, write request packets are guaranteed space at the destination. In this work, we apply CFC to the *forward* path (i.e., from the cores to the MCs) but not on the return path from the MCs to the cores. No flow control is explicitly needed on the return path since the forward path flow control also limits the replies from the MCs back to the cores.

3. Piggybacking can also be used but is not common in on-chip networks because of the abundant amount of on-chip wires.

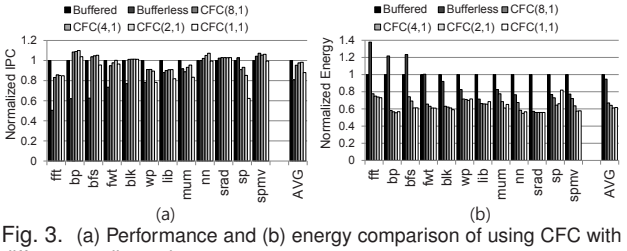

Fig. 3. (a) Performance and (b) energy comparison of using CFC with different credit numbers.

4 Evaluation

In this section, we evaluate the performance and energy impact of using CFC, using the configuration described earlier in Section 2.1. We evaluate CFC(r,w) but use $w = 1$ to avoid write deadlock and vary the value of r. Figure 3 shows the performance and energy results for CFC(r,1). As mentioned earlier, a baseline bufferless NoC results in significant loss of performance by up 50%, and an approximately 20% reduction on average. Among the different CFC(r,1) that we evaluated, CFC(2,1) resulted in the best overall performance, on average, and performance decreases when r is either increased or decreases As r increases, it introduces more traffic and causes more congestion, which degrades overall performance, while for a smaller value of r, throttling limited overall performance. On average, CFC(2,1) resulted in only a performance loss of only 1.8%; in several workloads, CFC(2,1) actually exceeded the performance of the baseline buffered network. CFC(2,1) also improves the performance of the baseline bufferless by 22%. The energy consumed in the baseline bufferless can exceed that of the baseline buffered by up to 38% (Figure 3(b)) and on average, only results in 5.3% reduction in energy. However, by using CFC(r,1), the energy is significantly reduced – CFC(2,1) reduces energy by 39% and 36% compared with the baseline buffered and baseline bufferless NoC, respectively.

To understand the results with CFC(r,w), we first plot the average number of deflections per flit in Figure 4(a). As r decreases, the amount of deflection also decreases as the number of flits in-flight are reduced. Compared with the baseline bufferless, CFC(2,1) results in an approximately 92% reduction

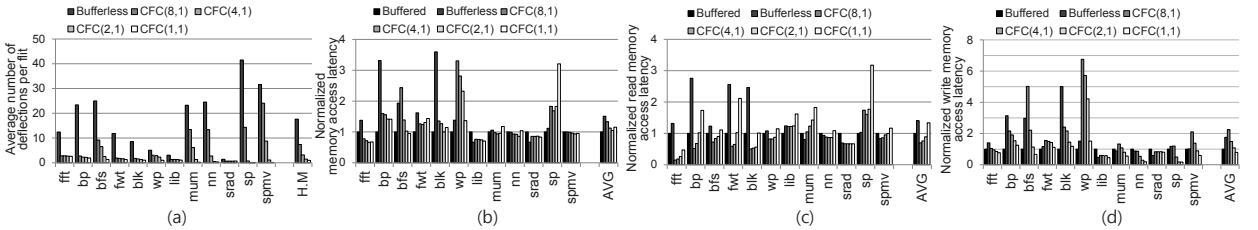

Fig. 4. (a) Average number of deflections and (b) average memory access latency (AMAT) latency per packet. The AMAT are partitioned into (c) read and (d) write requests. All results are normalized to baseline buffered NoC.

in average number of deflections per flit. The impact of both CFC flow control and reduction in deflection improves the average memory access latency (AMAT), as shown in Figure 4(b). The AMAT that we measure includes the request network latency from the core to the MC, the memory access latency, and the reply network latency from the MC back to the core. For the buffered NoC, the latency includes the queuing latency within the network, while for the bufferless and CFC(r,1), the additional source latency caused by throttling is also included; thus, AMAT is measured from when the request is generated till when the data arrives. The reordering latency (i.e., the amount of time it takes for all of the flits to arrive) for multi-flit packets is also included in the bufferless implementations.

While CFC(8,1) increases AMAT by 33% compared with the baseline buffered, CFC(2,1) results in minimal change (an increase of only 4.7%). For workloads such as SP, CFC(2,1) does increase AMAT by 83%, which results in a reduced overall performance but does not necessarily translate into an 83% reduction in performance since the large number of threads is able to tolerate some memory access latency. We also partition AMAT into read AMAT (Figure 4(c)) and write AMAT (Figure 4(d)). As r increases, the read AMAT in general decreases since the throttling is reduced resulting in improved read AMAT, except for CFC(1,1). However, for the write AMAT, all except for CFC(1,1) results in an *increase* of the write request latency as r increases. Although r is related to the read requests, it also impacts the write request latency since the amount of traffic from the read requests impact the write traffic. In this work, we assume a *static* CFC since each core had a static value of r and w. However, it remains to be seen if *dynamic* CFC can be implemented where the credits are *borrowed* between neighboring cores to help overall performance while still avoiding deadlock.

5 Related Work

Different bufferless NoC routers have been recently proposed [11], [8], [7]. To overcome the complexity in the control logic of the bufferless NoC, CHIPPER [5] was proposed to minimize the complexity with minimal loss in performance. However, these prior NoC architectures do not necessarily provide high performance at high load when packets continue to be deflected or need to be retransmitted multiple times. Recently, MinBD [6] was proposed, which adds an intermediate buffer to CHIPPER in order to minimize deflection. However, adding intermediate buffers reduces the benefits of the bufferless router. Gómez et al [7] reduced the amount of congestion or packets being dropped by combining dropping packets and misrouting and adding additional channels. In this work, we attempt to reduce network congestion without additional complexity to the network. Ogras and Marculescu [14] predicts buffer usage and throttles when space is not available in a buffered NoC. Nychis et al [13] argued for congestion control in bufferless NoC. They discussed application-aware throttling when multiple applications are being executed simultaneously and proposed a central congestion controller. Our work is similar as we also present an effective congestion control mechanism. However, this work differs as we focus on high-throughput multithreaded workloads and we show how a simple, distributed congestion control through CFC can be used. [10] shows the limitations of the bufferless because of the deflection routing; we overcome these limitations of the bufferless NoC through flow control.

6 Conclusion

Introducing flow control to bufferless on-chip networks (NoC) can extend bufferless NoC to manycore accelerators which results in high load for the network. In this work, we have introduced clumsy flow control (CFC), which reduces the amount of deflection in the network by throttling and reducing congestion in bufferless networks for manycore accelerators. Our results show that CFC results in 92% reduction in the amount of deflection that occurs in the network, thus, providing a 39% reduction in energy on average, with minimal loss in overall performance.

Acknowledgments

This research was supported in part by the MKE, Korea, under the ITRC support program supervised by the NIPA (NIPA-2012-H0301-12-1011) and in part by BST program through the NRF of Korea funded by the MEST(2012-0003579).

References

[1] A. Bakhoda *et al.*, "Throughput-Effective On-Chip Networks for Manycore Accelerators," in *MICRO*, pages 421–432, Dec 2010.
[2] A. Bakhoda *et al.*, "Analyzing CUDA workloads using a detailed GPU simulator." in *ISPASS*, pages 163–174, Apr 2009.
[3] S. Che *et al.*, "Rodinia: A benchmark suite for heterogeneous computing," in *IISWC*, pages 86–97, Oct 2009.
[4] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA: Morgan Kaufmann, 2004.
[5] C. Fallin *et al.*, "CHIPPER: A low-complexity bufferless deflection router," in *HPCA*, pages 144-155, Feb 2011.
[6] C. Fallin *et al.*, "MinBD: A Minimally-Buffered Deflection Router Approaching Conventional Buffered-Router Performance," in *NOCS*, May 2012.
[7] C. Gómez *et al.*, "Reducing packet dropping in a bufferless NoC," in *Euro-Par*, pages 899–909, Aug 2008.
[8] M. Hayenga *et al.*, "SCARAB: a single cycle adaptive routing and bufferless network," in *MICRO*, pages 244–254, Dec 2009.
[9] A. Kahng *et al.*, "ORION 2.0: A fast and accurate NoC power and area model for early-stage design space exploration," in *DATE'09*.
[10] G. Michelogiannakis *et al.*, "Evaluating bufferless flow control for on-chip networks," in *NOCS*, May 2010.
[11] T. Moscibroda and O. Mutlu, "A case for bufferless routing in on-chip networks," in *ISCA*, pages 196–207, Jun 2009.
[12] NVIDIA Corporation. NVIDIA CUDA SDK code samples., "http://developer.nvidia.com/cuda-downloads"
[13] G. Nychis *et al.*, "Next generation on-chip networks: what kind of congestion control do we need?" in *Proceedings of the Ninth ACM SIGCOMM Workshop on Hot Topics in Networks*, Oct 2010.
[14] U. Y. Ogras and R. Marculescu, "Prediction-based flow control for Network-on-Chip traffic," in *DAC*, pages 839–844, Jul 2006.
[15] Parboil Benchmark, "http://impact.crhc.illinois.edu/parboil.php"