Buffered Deflection Routing for Networks-On-Chip

Gadi Oxman
School of Electrical Engineering
Tel Aviv University, Tel Aviv, Israel
gdaliaox@post.tau.ac.il

Shlomo Weiss
School of Electrical Engineering
Tel Aviv University, Tel Aviv, Israel
weiss@eng.tau.ac.il

Yitzhak (Tsahi) Birk
Faculty of Electrical Engineering
Technion - Israel Institute of Technology, Haifa, Israel
birk@ee.technion.ac.il

ABSTRACT

Bufferless deflection routing works surprisingly well when the network traffic is light or medium, and can outperform a virtual channel router with a small number of buffers, but when the network is working closer to saturation, classic buffered virtual channel routers can sustain higher data rates, provided enough buffers are used. In this paper, we investigate extending bufferless deflection routing using the addition of router buffers. We propose two *buffered deflection routing* flow control algorithms that naturally extend bufferless deflection routing and still keep its attractive characteristics. We evaluate the proposed algorithms using a cycle accurate NoC simulator, and compare the results to bufferless deflection routing and virtual channel router with the same number of buffers. Our results show that buffered deflection routing offers substantial throughput gains while allowing a very efficient use of each added buffer, and can therefore be attractive for on-chip networks under heavy load.

Categories and Subject Descriptors

C.1.2 [**Processor Architectures**]: Multiple Data Stream Architectures—*Interconnection architectures*

Keywords

Networks-On-Chip, buffering, routing architectures, flow control, deflection routing

1. INTRODUCTION

Network-on-Chip (NoC) is being suggested as the main vehicle to connect on chip processors and other cores, while having packets flow between processors on this network [6]. Three key design parameters of a NoC are its topology, routing, and flow control. The topology of the network details the processors, routers, and the available communication links between them. Routing describes the path a packet takes. Flow control ensures that the buffers of each router do not overflow as packets flow through them. Large packets are often broken into small pieces called flits (flow control digits).

Deflection routing [2, 8] combines flow control and routing. Routing decisions are local to each router—a packet is either sent towards its target, or during congestion may be temporarily deflected from its path to destination. While deflection routing requires special care to address livelock danger and in-order packet delivery, it offers several desired properties, such as simple router design, low power footprint, automatic adaptation to hot spots in the network, automatic deadlock avoidance, inherent fault tolerance, and local routing decisions. These properties make deflection routing an attractive alternative for on-chip networks.

However *bufferless* deflection routing may not be the best design. Adding a small number of buffers may help to improve performance without significantly changing the attractive characteristics of deflection routing. This paper investigates deflection routing in on-chip networks. It demonstrates, by means of cycle accurate NoC simulation, how throughput of deflection routing on mesh NoC can be enhanced using router buffering, without resorting to a virtual channel [5] based router. This paper makes the following contribution.

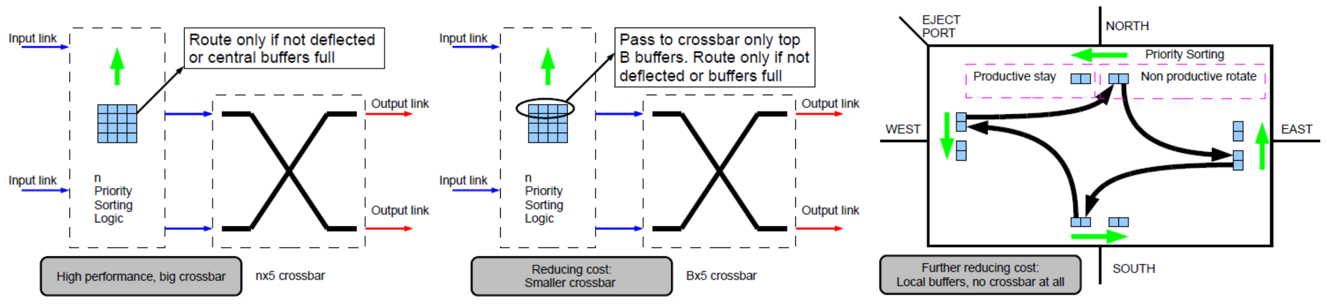
1. We introduce two new algorithms for buffered deflection routing: CENTRAL and RING.

2. We demonstrate that the use of buffers in deflection routing improves the throughput under heavy load.

In the sequel, we evaluate how bufferless deflection routing can be improved by introducing buffers, while keeping the spirit of deflection routing. Section 2 describes related work. Section 3 describes the buffered deflection routing algorithms. Section 4 describes our simulation methodology and results. Finally, conclusions are drawn in section 5.

2. RELATED WORK

Deflection routing was first proposed by Baran [2] for distributed communication networks, and features elimination of router buffers. Bononi *et al.* [3] analyze bufferless and single-buffer deflection routing in optical mesh networks. The authors evaluate Shufflenet and Manhattan Street Network topologies, and show analytically that even the use of a single buffer recovers a substantial amount of the throughput lost in the bufferless version of the network. Lu et al. performed an in-depth performance evaluation of bufferless deflection routing on NoC. The authors evaluated different


Input link
Route only if not deflected or central buffers full
Output link
Input link
n Priority Sorting Logic
Output link
High performance, big crossbar
nx5 crossbar
Input link
Pass to crossbar only top B buffers. Route only if not deflected or buffers full
Output link
Input link
n Priority Sorting Logic
Output link
Reducing cost: Smaller crossbar
Bx5 crossbar
EJECT PORT
NORTH
Priority Sorting
Productive stay
Non productive rotate
WEST
EAST
SOUTH
Further reducing cost: Local buffers, no crossbar at all

(a) CENTRAL(N_b,ALL) (b) CENTRAL(N_b,B) (c) RING

Figure 1: Buffered deflection routing algorithms with $N_b = 16, B = 4$. The central algorithm maintains 16 shared central buffers, where each cycle, any flit out of the best B candidates can be routed to any port. CENTRAL(N_b,ALL) uses all the buffers and inputs as candidates for routing. In the ring algorithm, the 16 buffers are divided to groups of 4 buffers per port where each group is local its port, and flits can enter/exit from/to that port only from the local buffers group, simplifying implementation. The non productive half of the ring buffers are rotated clockwise each cycle in search for a productive port.

topologies, as well as different routing algorithms and deflection policies [8]. Moscibroda and Mutlu performed a comprehensive study [10] of bufferless deflection routing and demonstrated its performance, area, and power benefits for NoC. In the same paper the authors also proposed a buffered variant that buffers flits in FIFOs, one FIFO per each router input port. Michelogiannakis et al. performed a comparison of virtual channel based buffered routing to bufferless deflection routing [9]. Abad et al. proposed a novel *rotary* router architecture [1] which replaces the usual router crossbar with two independent rings, each of them built from FIFO buffers, with the buffers rotating around the ring clockwise in one ring, and anti-clockwise in the other. The rotary router uses virtual cut through and bubble flow control mechanism to avoid deadlock, and uses rotation around the ring to substantially reduce head-of-line blocking effect.

3. BUFFERED DEFLECTION ROUTING

In this section we extend bufferless deflection routing with the addition of N_b flit buffers to each router. In the next subsections, we describe two algorithms using those N_b buffers: CENTRAL, and RING, both keeping the spirit of bufferless deflection routing in mind. While reading the description, periodically refer to Figure 1, which illustrates schematically the main difference between the algorithms.

Both algorithms inherit safety against end-to-end deadlock from ordinary bufferless deflection routing. Safety against deadlock is a major concern in the design of routers, which must make sure that the next router has a space for a packet before sending it, and therefore have to protect against an end-to-end deadlock danger in which all the routers in the network are stuck without being able to send packets to their neighbors. In deflection routing, each router locally decides to send a flit towards the next router without having to consult with it whether it can accept it – it is always ready, and this property of deflection routing is preserved in our buffered versions. Therefore, a deadlock situation in which no packet can advance is prevented.

On the other hand, since a flit can be deflected, there is a danger of a livelock condition, in which a flit will endlessly travel in the network without ever reaching its destination. Similar to bufferless deflection routing, livelock is avoided by assigning each flit a *flit priority* [8], which can be, for example, the flit age (number of clocks which elapsed since it was injected into the network). The flit age is attached to the flit header, and travels with it across the network. Each clock the flit is still in the network, it is incremented by one. In each router, older flits receive priority over younger flits and are routed to their destinations earlier, a property which is preserved as well by our buffered versions. Livelock is thus avoided, by ensuring that the oldest flit in the NoC is always able to make progress towards its destination.

3.1 CENTRAL algorithm

We use the entire set of flits incoming at the input ports I (smaller or equal to the degree D, the number of router ports), with addition of the buffered flits N_b, as candidates in the routing policy. For each flit, we say that a particular port is *productive*, if the distance of the flit to its destination will be shorter after going through this port. Otherwise, we say that the port is non-productive, and the flit will be *deflected* if routed to this port. We use the same rules of the bufferless deflection router for the combined set of input ports plus buffers with one major difference: as long as the buffers are not full, flits are routed *only* to productive ports, and flits that cannot be routed to productive ports will be *buffered* instead of deflected. In case we exhausted the available number of router buffers, buffers are deflected if there is contention on a productive port just as in the bufferless case. Note that the buffers are not associated with an input port, output port, or virtual channel. Rather, they are N_b *central* buffers for the whole router. Safety against livelock is ensured despite the addition of central buffers, since in each router, the highest priority oldest flit will be considered for routing first, and will therefore be able to make progress towards its destination.

While all the $N_b + I$ candidates are ranked according to their priority, the parameter B specifies a potentially smaller number of *best* candidates out of the total flits, which are considered for traversal through the crossbar, and therefore limits the size of the crossbar from N_b inputs down to B inputs, which is more competitive, in terms of implementation, with the bufferless router crossbar. As we'll see in the simulation results, limiting the crossbar size does reduce the performance, but still the performance is improved relative to the bufferless router, and therefore B provides a tradeoff between cost and performance. In summary, the router task is to choose up to D flits to route, out of the best B candidates, $D <= B <= N_b + I$, which have the highest flit priority out of the (N_b+I) total flits. The algorithm is described in Algorithm 1.

Algorithm 1: CENTRAL(N_b,B)

```
1 Rank the ($N_b$+I) flits in decreasing flit priority.
2 Iterate over the highest ranked B flits in decreasing flit
  priority, starting from the highest priority flit.
3 If the buffers are not full, assign the flit only to a
  productive port.
4 If the buffers are full, assign the flit to any available
  port, regardless if it'll be deflected or not.
5 Remaining flits not assigned to ports this cycle go into
  the buffers.
```

3.2 RING algorithm

We divide the N_b router buffers into D groups of $N_p = N_b/D$ buffers per group, where each group is associated with a particular router port. The groups are arranged in a ring structure, and each cycle half of the buffers in each group shift across the ring in a clockwise direction, while half of the buffers in the group stay in the same port. On each cycle, we consider two parameters for each flit: the *flit priority*, and whether this port is *productive* for this flit or not. We rank the flits according to both criteria (this can be implemented in hardware by comparing a number which is a concatenation of the productive flag and the port priority). We then route the highest priority productive flit, if available. In case all the flits are non-productive, we route the lowest priority non-productive flit. Once routing has been performed, we perform a simultaneous rotation of half the buffers in each port around the ring. The buffers which are rotated are the low half of the ranked list – containing highest priority non-productive flits and/or lowest priority productive flits. The most important highest priority productive flits will therefore stay at the port waiting for routing at the next cycles, while the non productive ports will rotate around the ring in search for a productive port. Note that arbitration is local for each port and its corresponding group of buffers at that cycle. Safety against livelock is ensured since there are two possibilities: either the oldest flit is positioned at a productive port and therefore routed through it, or in case it isn't in a productive port, it will be selected for rotation around the ring and will therefore reach a productive port before it completes a full rotation. Therefore, the oldest flit will always make progress towards its destination, avoiding livelock. The algorithm is described in Algorithm 2.

Algorithm 2: RING

```
1 Rank the $N_p$ flits, along with the incoming flit on the
  port, if any, in decreasing productivity and flit priority.
2 In the first routing phase, route highest priority flit
  which is productive, if available.
3 Else if no productive flit available, and the buffers are
  full, route the lowest priority non-productive flit.
4 Rotate half the buffers containing highest priority
  non-productive flits and lowest priority productive flits
  around the ring in search for a productive port.
```

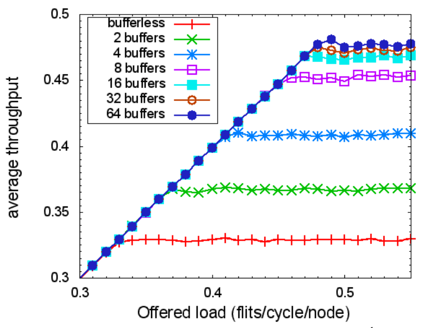


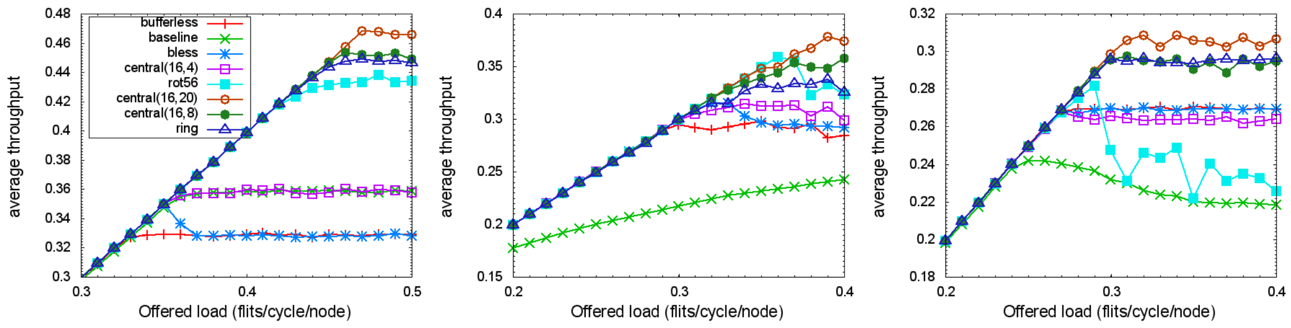
Figure 2: Performance of CENTRAL(N_b,ALL) algorithm with varying number of N_b buffers on a 8x8 mesh with uniform random traffic.

4. SIMULATION METHODOLOGY AND RESULTS

We test the proposed algorithms using a cycle accurate NoC simulator. The baseline virtual channel router is simulated using the booksim 2.0 NoC simulator [7]. The deflection, rotary and *BLESS* algorithms are modeled and simulated using our own NoC simulator. Our NoC simulator simulates the topology, processors, routers, and links. The processor model is a synthetic traffic generator. Each cycle each processor generates a new flit for network injection with a configurable injection probability. An infinite queue is simulated between the processor and the router to which it is connected, in which the flits are waiting for injection according to the deflection router rules (i.e. when one of the available links is available). A single flit can be ejected in each router per cycle. Simulation proceeds in three phases: warmup phase, in which packets are not monitored for statistics to let the network achieve a steady state, evaluation phase, in which packet statistics are monitored, and drain phase, at which point the simulator waits till all evaluation packets arrive to the destination. During the drain phase, packet generation is still enabled but newly injected packets are no longer monitored for statistics. Latency for each flit is monitored from the point the flit was generated till the point the flit was ejected, including the time the flit spent in the source injection queue. The number of deflections for each flit are tracked, and the average number of deflections a flit experiences is computed. Throughput is measured by calculating the number of flits ejected during the evaluation period. Simulation is performed on the 2D mesh, using 64 routers in 8x8 configuration, under uniform random, transpose and tornado traffic [4].

Figure 2 shows the throughput of the *CENTRAL(N_b,ALL)* algorithm, which demonstrates the best performance, with varying number of buffers from 2 to 64. We see that even under heavy load, the router is able to utilize additional buffers to increase performance. However, we can see an effect of diminishing returns as the number of buffers are increased, with 32 and 64 buffers providing nearly the same performance, and 16 buffers not lagging much behind them. The highest "return on investment", per buffer, is with small number of buffers.

We compare the performance of our proposed algorithms with 16 buffers: *RING*, and *CENTRAL* with three configurations of B (20, 8 and 4) to the *BASELINE* virtual channel router [4], *ROTARY* router [1], *BLESS with buffers* [10], and *BUFFERLESS* routing. For comparison, *BASELINE* is simulated with single cycle router latency, 4 virtual channels



(a) uniform (b) transpose – same legend as (a) (c) tornado – same legend as (a)

Figure 3: Comparison of buffered algorithms. Throughput is compared with 16 buffers on a 8x8 mesh, except the rotary router which is using 56 buffers, and bufferless which is shown for reference. Throughput is shown for (a) uniform random, (b) transpose, and (c) tornado traffic patterns. CENTRAL(16,ALL) offers the best performance, while RING and CENTRAL(16,8) only slightly below.

per port with one flit buffer in each, and *dimension ordered routing*. There are no deflections in *BASELINE* as flits will always travel through their preferred paths. *BLESS with buffers* is using an input FIFO with four buffers per port. *CENTRAL(N_b,ALL)* is using all the buffers and inputs as candidates for routing, while *CENTRAL(N_b,8)* is using the top 8 buffers and/or inputs, and *CENTRAL(N_b,4)* is using only the top four. Figure 3 compares the performance with 16 buffers, except *ROTARY* which requires more buffers due to its double ring structure and bubble flow control deadlock avoidance algorithm, and is therefore simulated with 56 buffers.

Simulation shows that the proposed algorithms offer best network performance under heavy load, with RING and *CENTRAL(N_b,8)* performance only slightly below the best performing algorithm *CENTRAL(N_b,ALL)*. Besides uniform random, the proposed algorithms cope well with the harder transpose and tornado traffic patterns, demonstrating that the inherent robustness of deflection routing to hot-spots provided by deflecting the flits to less congested areas, is preserved with the addition of buffers. The additional buffers are able to improve throughput under heavy load because, in contrast to FIFO buffering structures which are subject to head-of-line blocking (in which buffers in the middle of the FIFO are blocked from routing), in the proposed central and ring buffering structures head-of-line blocking is eliminated and every buffer is a candidate for routing.

5. CONCLUSIONS AND FUTURE WORK

We have introduced two new buffered deflection routing algorithms. Both algorithms preserve the benefits of deflection routing while improving its attractiveness by improving performance under heavy load. We showed that using even a small number of buffers reduces the number of deflections and provides substantial throughput gain.

Both our algorithms avoid head-of-line blocking in the buffers, however avoiding FIFOs is not without a cost. Indeed while the *CENTRAL(N_b,ALL)* algorithm performs the best, since every buffer can go to every output, with large number of buffers the crossbar can be too cost prohibitive. The *CENTRAL(N_b,ALL)* is therefore interesting as a limit case of the performance which can be achieved, but might not be feasible. We have therefore proposed a cost reduced version *CENTRAL(N_b,B)* which reduces the crossbar size, and offers a tradeoff between the cost and performance using the parameter B.

We also proposed a low cost ring deflection algorithm, in which each port performs only local scheduling decisions with a reduced N/D number of buffers without requiring a crossbar at all, hence shortening the implementation critical paths and improving the router maximum operation frequency while gaining much of the performance advantage of the more complex CENTRAL algorithm. For example for RING(16), each port will have four buffers, a four-entry priority sorter, and a multiplexor of 4:1, with overall cost which can be similar to that of the bufferless router, but at higher performance. Our future work will focus on in-depth evaluation of the area and power of the proposed routers.

6. REFERENCES

[1] P. Abad, V. Puente, J. A. Gregorio, and P. Prieto. Rotary router: an efficient architecture for cmp interconnection networks. *SIGARCH Comput. Archit. News*, 35:116–125, June 2007.

[2] P. Baran. On distributed communication networks. *IEEE Trans. on Commun. Systems*, CS-12, 1964.

[3] A. Bononi, F. Forghieri, and P. R. Prucnal. Analysis of one-buffer deflection routing in ultra-fast optical mesh networks. In *Proc. IEEE INFOCOM '93*, pages 303–311, 1993.

[4] W. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[5] W. J. Dally. Virtual-channel flow control. In *ISCA '90: Proceedings of the 17th annual international symposium on Computer Architecture*, pages 60–68, 1990.

[6] W. J. Dally and B. Towles. Route packets, not wires: on-chip inteconnection networks. In *DAC '01: Proceedings of the 38th annual Design Automation Conference*, pages 684–689, 2001.

[7] N. Jiang, G. Michelogiannakis, D. Becker, B. Towles, and W. Dally. Booksim interconnection network simulator. `https://nocs.stanford.edu/cgi-bin/trac.cgi/wiki/Resources/BookSim`.

[8] Z. Lu, M. Zhong, and A. Jantsch. Evaluation of on-chip networks using deflection routing. In *GLSVLSI '06: Proceedings 16th ACM Great Lakes symposium on VLSI*, pages 296–301, 2006.

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

[10] T. Moscibroda and O. Mutlu. A case for bufferless routing in on-chip networks. In *ISCA '09: Proceedings of the 36th annual international symposium on Computer architecture*, pages 196–207, 2009.