Streamlined Network-on-Chip for Multicore Embedded Architectures

Gadi Oxman[1], Shlomo Weiss[1], and Yitzhak (Tsahi) Birk[2]

[1] School of Electrical Engineering,
Tel Aviv University, Tel Aviv, Israel
[2] Faculty of Electrical Engineering,
Technion - Israel Institute of Technology, Haifa, Israel

Abstract. MPSoCs are becoming complex systems incorporating a large number of compute cores as well as various accelerators and application specific units. To handle the communication in MPSoCs, the Network-on-Chip (NoC) concept has been proposed as a versatile and scalable solution. The cost of the communication subsystem may have a major impact on the overall cost of the SoC; hence the need for careful evaluation of NoC design alternatives. Deflection routing, characterized by router simplicity and minimal resources, is an attractive design alternative but is generally viewed as suitable only for NoC with low and medium traffic. In this paper, we propose prioritization and buffering algorithms which improve deflection routing performance to the point it becomes attractive in heavily loaded NoC as well.

Keywords: Multicore embedded systems, network-on-chip, NoC, MPSoC, deflection routing.

1 Introduction

The growing complexity of Multi-Processor System-on-Chips (MPSoCs) and the requirement for scalability underscore the need for efficient on-chip communication. Network-on-Chip (NoC) designs [2] have been proposed to address both complexity and scalability. NoC design involves tradeoffs [15] with respect to performance (latency and throughput), cost (power [6,14] and silicon), and buffer size [7]. Routing has a significant impact on the cost of the NoC [10,9].

Due to its simplicity, in this paper we focus on deflection routing. The merit of deflection routing is that it needs scarce resources. It does not employ routing tables, and it works without buffering, although we show that a limited number of buffers is profitable for improving performance. In deflection routing a packet is either sent towards its target, or during congestion may be temporarily deflected from its path to destination. Deflection routing offers several desired properties including simple router design, low power footprint, congestion leveling, and fault tolerance.

Although these properties make deflection routing an attractive alternative for NoCs in multi-core embedded systems for low and medium traffic, its performance

A. Herkersdorf, K. Römer, and U. Brinkschulte (Eds.): ARCS 2012, LNCS 7179, pp. 238–249, 2012.

under heavy load is thought to be lacking. In this paper, we show how to employ deflection routing in high traffic NoC. Our contribution is as follows.

1. We introduce new prioritized deflection routing algorithms: MULTIPATH, and RADIAL.
2. We introduce two new buffered deflection routing algorithms: CENTRAL, and RING.
3. We demonstrate that deflection routing performance under heavy load can be substantially improved through prioritization and buffering.

The paper is organized as follows. Section 2 discusses related work. Section 3 discusses the MULTIPATH and RADIAL priority based routing algorithms. Section 4 introduces the CENTRAL and RING buffered algorithms. Section 5 presents and discusses our results. Finally, conclusions are drawn in section 6.

2 Related Work

NoC design involves tradeoffs [15] with respect to performance (latency and throughput), cost (power [6,14] and silicon), and buffer size [7].

Bufferless deflection routing has been a popular technique in Optical Burst Switching (OBS) networks [5], because integrated optical buffers do not compare favorably with electronic buffers [17] in terms of area, power, and capacity, and fiber optical buffers are large.

Lu et al. evaluated several priority based criteria for bufferless deflection routing on NoC, including DIMENSION-XY and DELTA-XY [11]. Radetzki and Kohler [16] proposed evaluating and minimizing a cost based routing function over all input-output permutations.

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.

Kim et al. proposed a lightweight router micro-architecture for a NoC based on ring topology [10]. The work suggests adding a single buffer to each router, and utilizing credit-based flow control to manage the buffers, but without requiring virtual channels due to rotary flow control possible in a ring topology. In another work, Kim proposes to add buffers to the 2D mesh using a dimension sliced router, which partitions the crossbar switch into two smaller crossbar switches, one for each dimension, and adds an intermediate buffer for packets switching between dimensions [9].

Moscibroda and Mutlu performed a comprehensive study of bufferless deflection routing [13] and demonstrated its performance, area and power benefits for NoC under light and medium load. In their study, the authors proposed a buffered variant of their *BLESS* algorithm called *BLESS with buffers*, which 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 [12], and also proposed the MAX-XY priority based algorithm.

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 Prioritized Deflection Routing Algorithms

Large packets are often broken into small pieces called flits (flow control digits). In this work, we assume flits travel on their own and we do not guarantee in-order multi-flit packet delivery. A flit is routed in a *productive direction* through a *productive port*, if the distance between the flit position and its destination decreases. Otherwise, the flit is routed in a *non-productive* direction, and is *deflected*. For example in the 2D mesh, a router has a maximum of four ports, and can have up to two productive directions. Each deflection temporarily moves a flit further away from its destination. While on the surface it seems that deflections could be bad for performance, using deflections the network adaptively senses hot spots of congestion, and routes flits in different paths around them.

In each router, each flit i is assigned a *flit priority* F_i, and a *port priority* P_{ij}. F_i describes how important is the delivery of the flit relative to other flits, and P_{ij} describes how desirable it is for flit i to be routed through port j relative to other ports. We use a greedy algorithm to assign flits with higher flit priority to their desired high priority ports before assigning lower priority flits, as described in Algorithm 1.

Algorithm 1. Greedy routing with priorities

```
repeat
    Choose the flit with the highest flit priority F_i.
    Assign the flit i to the port j with the highest port priority P_ij which has
    not been assigned yet.
until all flits routed
```

3.1 MULTIPATH Flit Priority

Deflection routing is safe from a deadlock condition in which no flit can advance in the network, since each router locally decides to send a flit to the next router, and the next router must always accept it. On the other hand, livelock, in which a flit stays in the network without reaching its destination, should be addressed. To that end, the flit age is tracked (time since the flit was injected into the network, starting at 0 and incremented by 1 each cycle the flit is still traveling in the network), and the routing algorithm assigns the age as the flit priority,

$F_i = Age$, to ensure that older flits in the network are prioritized over younger flits and make progress towards their destination earlier. We propose using the following MULTIPATH flit priority criteria instead:

$$F = \begin{cases} Age - C * (N_{productive} - 1), & \text{if } N_{productive} > 0 \\ Age - C * D, & \text{if } N_{productive} = 0 \end{cases} \quad (1)$$

Where $N_{productive}$ is the number of productive directions a flit has to the destination at this router, D is the switch degree (number of input ports), and C is a parameter which provides a tradeoff between throughput and latency. Using the MULTIPATH criteria balances the flits age with the number of available productive paths, allowing flits with only a single productive path to *temporarily* be prioritized over older flits with multiple productive directions. Penalizing flits that have multiple productive paths available by reducing their priority offers the advantage of first routing the flits which have only one productive direction, thereby offering more choice for productive routing, decreasing the overall deflections, and improving network throughput. We differentiate between two versions of the greedy MULTIPATH routing algorithm: the *non-recursive* version calculates $N_{productive}$ once at the start of each clock and doesn't modify it (thus $N_{productive} > 0$), and the *recursive* version recalculates $N_{productive}$ and the flit priority F_i after each flit is routed, therefore $N_{productive}$ can be 0 in case the productive ports were already taken earlier.

3.2 RADIAL Port Priority

We now shift our attention to port priorities. DIMENSION-XY, DELTA-XY and MAX-XY have been proposed earlier [11,12]. We propose a RADIAL algorithm, which may alleviate congestion at the center of the network. We calculate for each router (x, y) in a $N \times N$ 2D mesh its *radial distance* from the center, R, using (2). For higher dimension meshes, we add similar terms with the additional dimensions to the maximum value calculation. This assigns the routers in the 2D mesh to rectangular rings. For example, for the 8×8 2D mesh, the routers will be assigned to one of four rings $R = 0$ (inner ring),1,2,3 (outer ring), as illustrated in Figure 1.

$$R = \lfloor max\{|x - \frac{N-1}{2}|, |y - \frac{N-1}{2}|\}\rfloor \quad (2)$$

The RADIAL port priority algorithm tries to alleviate congestion by routing flits away from the center of the mesh. It prefers productive directions to non-productive directions, but when there is a choice of multiple productive directions, the direction which increases, or at least does not decrease, the radial distance R, is preferred over a productive direction which decreases R. Similarly, if the flit must be deflected since all productive ports are already taken by higher priority flits, and in case there are several deflection directions possible, the direction which increases R will be preferred. The algorithm is described in Algorithm 2.

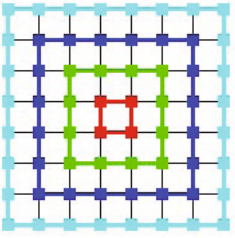


Fig. 1. Assignment of routers to rings based on their distance from the mesh center

Algorithm 2. RADIAL port priority algorithm

1 For each free port (both productive and not), calculate the radial distance R of the next router.
2 Choose the productive port with the highest R, if available.
3 Otherwise, if no productive port is available, deflect the flit to the free non-productive port with the highest R.

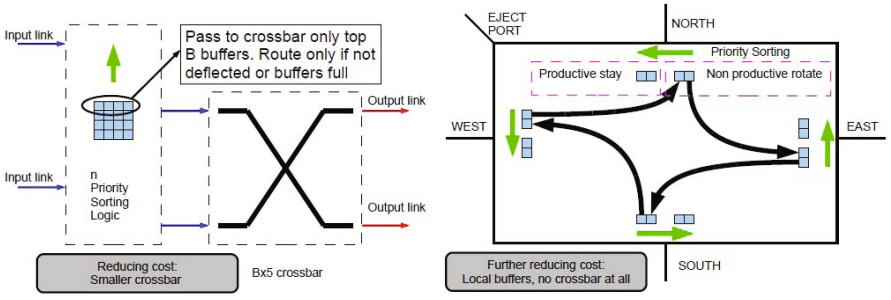
4 Buffered Deflection Routing

Having discussed flit priority methods as a vehicle to improve NoC performance under heavy load, we now turn our attention to the use of buffers for the same purpose. Priority methods reduce deflections by smarter routing decisions, while buffering reduce deflections by holding flits which couldn't be routed to a productive direction in buffers instead of deflecting them, while hoping that productive ports will soon be available in the next clock cycles. We propose two buffered algorithms which extend bufferless deflection routing: CENTRAL, and RING.

4.1 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. 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 which can not 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.

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 number of crossbar inputs from $N_b + I$ down to B inputs, which is



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

Fig. 2. 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. In the ring algorithm, the 16 buffers are divided to groups of 4 buffers per port where each group is local to 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.

more competitive, in terms of implementation area and power, 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, $D <= B <= N_b + I <= N_b + D$, which have the highest flit priority out of the (N_b+I) total flits. In case all available flits (both buffered and input) are candidates for crossbar traversal without limitation, we refer to the algorithm as CENTRAL(N_b,ALL), and we mean that $B = N_b + I$. The CENTRAL algorithm is described in Algorithm 3.

Algorithm 3. 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.

4.2 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

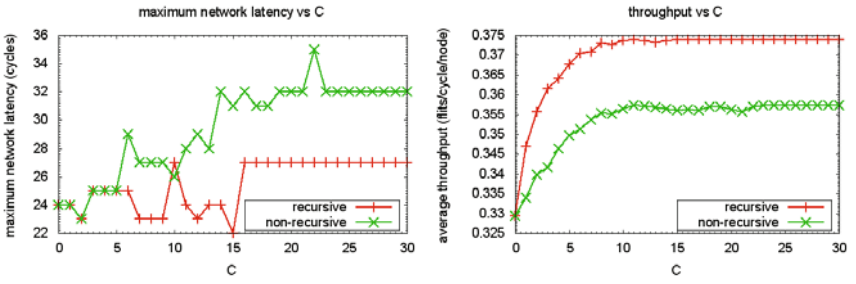
arranged in a ring structure, and each cycle half of the buffers in each group shift across the ring in a clockwise direction, while half of the buffers in the group stay in the same port. On each cycle, we consider two parameters for each flit: the *flit priority*, and whether this port is *productive* 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. The algorithm is described in Algorithm 4.

Algorithm 4. 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.

5 Experimental Method and Results

We developed a cycle accurate NoC simulator in the C programming language, and used it to simulate deflection routing performance with the priority and buffering algorithms. The simulator contains simulation modules for flits, routers, router-router links, processors, and processor-router links. Each router is connected to a processor through processor-router links. By default two such links are used, one to inject flits, and one to eject flits. The processor-router links model an infinite queue used to achieve an open-loop simulation, in which a specific network load can be simulated by the processors even if the network can not sustain it, in which case the flits will be kept in the FIFO waiting for injection into the network. Routers are interconnected using router-router links, which are implemented as dual pointers to the flit data structures, one "old" for the status of the link at the beginning of the simulated clock cycle, and another "new" for the status of the link at the end of the cycle. During initialization, the simulator reads a configuration file which describes the simulation parameters such as the

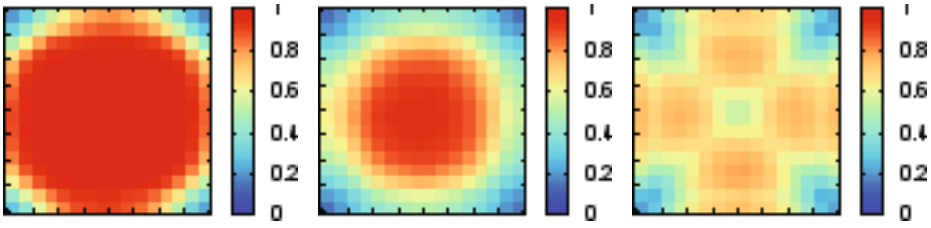


(a) maximum network latency (b) throughput

Fig. 3. Effect of C on the multipath flit priority algorithm for a 8×8 mesh under uniform random traffic

network topology and routing algorithm. Next, the topology is constructed by allocating routers and interconnecting them using links. Cycle by cycle simulation is then started. At each cycle, the simulator cycles through each router, invoking an algorithm specific function to simulate the router behavior, based on the configured parameters. The router will process flits from input links, and route new flits into output links. At the end of the cycle, each link copies the "new" flit set at this cycle into the "old" flit data structure. The cycle counter is then advanced and a new simulation cycle begins. The processor model is a synthetic traffic generator, capable of generating uniform random, transpose, and tornado patterns [4]. In uniform random traffic, the destination router is randomly chosen. In the transpose traffic, each processor at address $\{x, y\}$ generates flits to the diagonally opposite router at destination $\{y, x\}$. In the tornado traffic for a $N \times N$ network, each processor at address $\{x, y\}$ sends flits half-way across the network to address $\{(x + \frac{N}{2} - 1)\ mod\ N, (y + \frac{N}{2} - 1)\ mod\ N\}$. Each processor generates flits at a specified injection rate – each cycle the processor uniformly rolls a number between 0 to 1, and generates a new flit if the number is bigger than the specified rate. The simulation period is split to three phases: warmup phase, evaluation phase, and drain phase. During the warmup phase, the simulator waits for a steady network state and does not monitor flit statistics. During the evaluation phase, flit statistics such as injection time, ejection time, number of deflections and congestion are being monitored. During the drain phase, flit generation is still enabled but newly injected flits are no longer monitored for statistics, and the simulator waits till all flits monitored during the evaluation phase are received at the destination router.

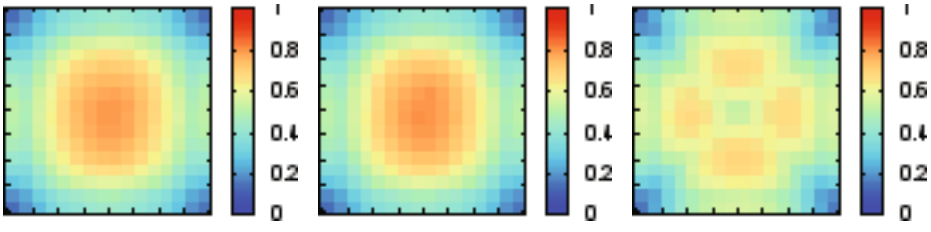
Figure 3 shows the effect of the parameter C of the MULTIPATH priority algorithm. We simulated a 8×8 mesh with bufferless deflection routing and DIMENSION-XY port priority, under uniform random traffic. Saturation throughput was measured by calculating the number of flits ejected from all the routers in the NoC during the evaluation period, divided by the number of cycles and number of routers, while using a flit injection probability of 0.5 per router. Maximum network latency was calculated by monitoring the



(a) Bufferless, AGE, DIMENSION-XY: average: 0.87, standard deviation: 0.19

(b) Bufferless, MULTIPATH, DIMENSION-XY: average: 0.64, standard deviation: 0.22

(c) Bufferless, MULTIPATH, RADIAL: average: 0.61, standard deviation: 0.14



(d) Buffered, AGE, DIMENSION-XY: average: 0.52, standard deviation: 0.16

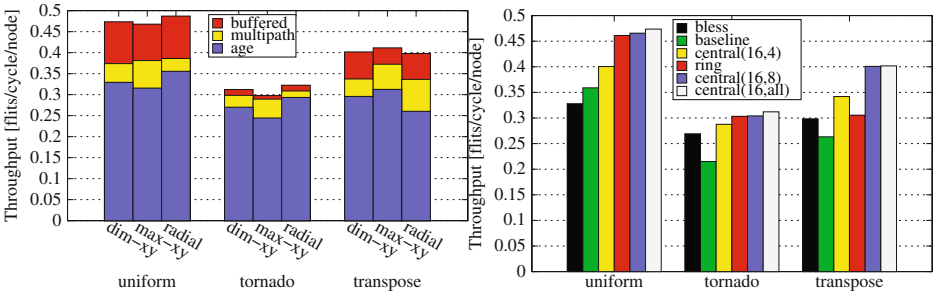
(e) Buffered, MULTIPATH, DIMENSION-XY: average: 0.52, standard deviation: 0.17

(f) Buffered, MULTIPATH, RADIAL: average: 0.52, standard deviation: 0.13

Fig. 4. Comparison of 16×16 NoC congestion uniform traffic with injection rate 0.18. Top line, left to right: bufferless with DIMENSION-XY, MULTIPATH and MULTIPATH+RADIAL. Bottom line: same with buffered.

maximum age of each flit when it reached its destination and exited the network. The figure demonstrates throughput is improved when C is increased, however the maximum network latency is increased as well, and therefore C provides a tradeoff between throughput and latency. The figure also demonstrates that the recursive variant further increases the throughput and lowers the latency. In that particular mesh configuration, the effective range of C is up to about $C = 25$, at which point both the latency and throughput reach stable values.

We measured the effect of prioritization and buffering on deflection routing network congestion under heavy load. Congestion maps were generated by displaying a map of average congestion per router during the simulation evaluation period. Average congestion is calculated by calculating the total number of incoming flits into the router, divided by the number of cycles and the number of router input links. This provides a normalized number between 0 and 1, where 0 is "not congested" and 1 is "very congested". We used a 16×16 network under uniform random traffic. We used an injection rate of 0.18 which is very close to the saturation bandwidth for bufferless deflection routing with Age flit priority and DIMENSION-XY, without our proposed enhancements. Then, in turn, we



(a) deflection routing with multipath prioritization and buffering

(b) buffered algorithms comparison

Fig. 5. Saturation throughput of deflection routing on 8×8 mesh. (a): Saturation throughput with deflection routing with prioritization and buffering. (b) Performance comparison of different buffered algorithms with 16 total router buffers each.

added MULTIPATH ($C = 25$, recursive), next we added both MULTIPATH and RADIAL but still bufferless. Next, we tested CENTRAL buffering ($B = ALL$) with MULTIPATH and DIMENSION-XY, and finally we enabled all three and test MULTIPATH, RADIAL and CENTRAL together. Fig 4 shows the resulting congestion maps. We see that both priority and buffering alleviate congestion. The best version with all enhancements enabled improved throughput to 0.246, an improvement factor of 1.36. For the original saturation traffic of 0.18, average congestion, was reduced from 0.87 to 0.52, a factor of 1.67. In the buffered case all three schemes resulted in the same average congestion of 0.52, and therefore the prioritization was less important there, although the standard deviation of the MULTIPATH RADIAL was slightly lower.

Figure 5a illustrates the saturation throughput of deflection routing under bufferless routing, bufferless multipath routing, and buffered routing (CENTRAL algorithm with N_b=16 buffers and B=ALL), with three port priority algorithms: DIMENSION-XY, MAX-XY and RADIAL, and three traffic patterns: uniform random, tornado, and transpose, on the 8×8 mesh. In all cases, multipath and buffering improves performance. Radial port priority improves performance in uniform random and tornado traffic, but is a below max-xy performance under transpose traffic.

Figure 5b compares the proposed prioritized buffered algorithms to previously proposed router architectures: the baseline VC router [4], and BLESS with buffers [13]. The baseline virtual channel router was simulated using the booksim 2.0 NoC simulator [8]. The deflection, and *BLESS with buffers* algorithms were modeled and simulated using our own NoC simulator described earlier. We simulated a 8×8 mesh with the same number of total buffers, 16, in all buffered algorithms, and used DIMENSION-XY and MULTIPATH in the CENTRAL and RING algorithms. The simulation was performed with three traffic patterns: uniform random, transpose and tornado. Simulation shows that the

proposed algorithms offer best network performance under heavy load, with the cost reduced BUFDR-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 since 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 additional buffers are candidates for routing. *CENTRAL(N_b,ALL)* performs best since all buffered flits are candidates for routing without restriction, however it requires a big crossbar with $N_b + D$ inputs.

6 Conclusions

Deflection routing allows streamlined router implementation, which makes it an attractive routing policy in NoCs for multi-core embedded systems. In this paper we have shown that deflection routing may also provide good performance. We demonstrated that performance can be improved using prioritized and buffered deflection routing algorithms, and that using even a small number of buffers provides substantial performance gains. The CENTRAL algorithms demonstrates the best performance, but may have high implementation cost with a large number of buffers. We proposed simplified versions of the algorithms which provide a tradeoff between cost and performance, and plan to evaluate their area and power benefits in future research.

Acknowledgements. We thank Erik Maehle and the reviewers for their remarks and suggestions.

References

1. Abad, P., Puente, V., Gregorio, J.A., Prieto, P.: Rotary router: an efficient architecture for cmp interconnection networks. SIGARCH Comput. Archit. News 35, 116–125 (2007)
2. Bjerregaard, T., Mahadevan, S.: A survey of research and practices of network-on-chip. ACM Comput. Surv. 38 (2006)
3. Bononi, A., Forghieri, F., Prucnal, P.R.: Analysis of one-buffer deflection routing in ultra-fast optical mesh networks. In: Proc. IEEE INFOCOM 1993, pp. 303–311 (1993)
4. Dally, W., Towles, B.: Principles and Practices of Interconnection Networks. Morgan Kaufmann Publishers Inc., San Francisco (2003)
5. Hsu, C.-F., Liu, T.-L., Huang, N.-F.: Performance analysis of deflection routing in optical burst-switched networks. In: INFOCOM 2002: Proceedings 21st Annual Joint Conference of the IEEE Computer and Communications Societies, pp. 66–73 (2002)

6. Hu, J., Marculescu, R.: Energy-aware mapping for tile-based noc architectures under performance constraints. In: Proceedings of the 2003 Asia and South Pacific Design Automation Conference, ASP-DAC 2003, pp. 233–239 (2003)
7. Jafari, F., Lu, Z., Jantsch, A., Yaghmaee, M.H.: Buffer optimization in network-on-chip through flow regulation. Trans. Comp.-Aided Des. Integ. Cir. Sys. 29, 1973–1986 (2010)
8. Jiang, N., Michelogiannakis, G., Becker, D., Towles, B., Dally, W.: Booksim interconnection network simulator, `https://nocs.stanford.edu/cgi-bin/trac.cgi/wiki/Resources/BookSim`
9. Kim, J.: Low-cost router microarchitecture for on-chip networks. In: Proc. 42nd Annual IEEE/ACM Int'l Symp. on Microarchitecture, MICRO 42, pp. 255–266 (2009)
10. Kim, J., Kim, H.: Router microarchitecture and scalability of ring topology in on-chip networks. In: Proc. 2nd Int'l Workshop on Network on Chip Architectures, NoCArc 2009, pp. 5–10 (2009)
11. Lu, Z., Zhong, M., Jantsch, A.: Evaluation of on-chip networks using deflection routing. In: GLSVLSI 2006: Proceedings 16th ACM Great Lakes Symp. on VLSI, pp. 296–301 (2006)
12. Michelogiannakis, G., Sanchez, D., Dally, W.J., Kozyrakis, C.: Evaluating bufferless flow control for on-chip networks. In: NOCS 2010: Proc. 2010 Fourth Int'l Symp. on Networks-on-Chip, pp. 9–16 (2010)
13. Moscibroda, T., Mutlu, O.: A case for bufferless routing in on-chip networks. In: ISCA 2009: Proc. 36th Annual Int'l Symp. on Computer Architecture, pp. 196–207 (2009)
14. Palesi, M., Holsmark, R., Kumar, S., Catania, V.: Application specific routing algorithms for low power NoC design. In: Silvano, C., Lajolo, M., Palermo, G. (eds.) Low Power Networks-on-Chip, pp. 113–150. Springer, Heidelberg (2011)
15. Pande, P.P., Grecu, C., Jones, M., Ivanov, A., Saleh, R.: Performance evaluation and design trade-offs for network-on-chip interconnect architectures. IEEE Trans. Comput. 54, 1025–1040 (2005)
16. Radetzki, M., Kohler, A.: An intelligent deflection router for networks-on-chip. In: 2009 Seventh Workshop on Intelligent Solutions in Embedded Systems, pp. 57–62 (June 2009)
17. Tucker, R.S.: The role of optics and electronics in high-capacity routers. Journal of Lightwave Technology 24(12), 4655–4673 (2006)