J. Parallel Distrib. Comput. 72 (2012) 515–524

Contents lists available at SciVerse ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Making-a-stop: A new bufferless routing algorithm for on-chip network

Jing Lin [a], Xiaola Lin [a,b,*], Liang Tang [c]

[a] Department of Computer Science, School of Information Science and Technology, Sun Yat-sen University, Guangzhou, China

[b] Key laboratory of Ministry of Education for Digit Home, Guangzhou, China

[c] Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai, China

ARTICLE INFO

Article history:
Received 15 January 2011
Received in revised form
30 September 2011
Accepted 3 January 2012
Available online 20 January 2012

Keywords:
Network-on-chip
Wormhole
Bufferless routing
Livelock freedom

ABSTRACT

In the deep submicron regime, the power and area consumed by router buffers in network-on-chip (NoC) have become a primary concern. With buffers elimination, bufferless routing is emerging as a promising solution to provide power-and-area efficiency for NoC. In this paper, we present a new bufferless routing algorithm that can be coupled with any topology. The proposed routing algorithm is based on the concept of making-a-stop (MaS), aiming to deadlock and livelock freedom in wormhole-switched NoC. Performance evaluation is carried out by using a flit-level, cycle-accurate network simulator under synthetic traffic scenarios. Simulation results indicate that the proposed routing algorithm yields an improvement over the recent bufferless routing algorithm in average latency, power consumption, and area overhead by up to 10%, 9%, and 80%, respectively.

1. Introduction

Continued developments in very-large-scale integration (VLSI) technology enable more cores available on a single chip [14]. To meet the increasing complexity and communication requirements arising in such large systems, scalable network-on-chip (NoC) paradigm has been put forth as a potential solution [7,3,15]. Compared to traditional shared-bus on-chip communications, NoC offers lower packet latency, better predictability, and much greater scalability.

Although on-chip network has started to form the backbone of communication between cores, the performance of such interconnection network is bounded by the limited power and area budgets [20]. It has been reported that a substantial portion of system power is dissipated by NoC, e.g., roughly 30% in the Intel TeraFLOPS chip [28] and 36% in the MIT RAW chip [27]. In addition, the on-chip network has to compete with the cores for the real estate of the same chip. Therefore, power consumption and silicon area have been emerging as the most dominant roadblocks for NoC design. It is critical to design power and area efficient on-chip network. Based on different research techniques, there is a broad avenue for research studies of power reduction in NoC, spanning all the way from system level [1,21,26] to microarchitecture level [13,22,18,17]. Among various proposed power efficient NoC designs, network topology and routing are two of the most important aspects. In this paper, we focus on proposing a new routing algorithm.

Routing algorithms can be classified in several ways in the literature. In buffered routing, when packets compete for the same output port, the winning packet advances through the output port while the losing ones can be stored at buffers in the router until the required port becomes available. However, despite of its ability to improve bandwidth efficiency, buffer dominates both the on-chip network power and area [28,11]. For example, router buffers dissipate 22% of the router power [28] and consume 75% of NoC area [11]. In addition, buffering adds the complexity of on-chip network design. Not only extra logics are needed to place packets into and out of buffers, specific buffer management mechanism also needs to be implemented to prevent buffers from overflowing. Last but not least, deadlock may arise when buffers are not allocated properly [6,5].

Based on the above observations, a promising solution is to use the bufferless routing method [24,10,12,23]. In bufferless routing, there is no buffer to house packets when no proper output ports are available for them. Once contention arises, a router must decide how to dispose the packets which fail to get the proper output ports. Two main mechanisms of such disposition, dropping and deflecting, have been proposed. With dropping mechanism, failed packets are dropped and then retransmitted by the source node. The main problem in dropping mechanism is that on one hand, the source node needs to store the issued packet until a positive acknowledgment packet is received; on the other hand, in the event of collision, a negative acknowledgment (NACK) packet is

* Corresponding author at: Department of Computer Science, School of Information Science and Technology, Sun Yat-sen University, Guangzhou, 510006, China.
E-mail address: linxl@mail.sysu.edu.cn (X. Lin).

0743-7315/$ – see front matter © 2012 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2012.01.001

sent back to the source node to trigger the retransmission of the dropped packet. The injection of NACKs and the retransmission of dropped packets obviously increase total network burden especially in a heavy-loaded network. Moreover, as NACKs cannot be dropped, small NACK buffers are required in routers input port [10] or additional separate NACK network is implemented [12], both of which increase the design complexity and do not rigorously prove livelock freedom.

As most on-chip network designs cannot tolerate the dropping of packets [16], in this paper, we adopt deflecting bufferless routing. Deflection routing was first introduced as hot-potato routing in [2]. When deflected, a packet is sent in the non-minimal direction, getting away from its destination. If a packet is deflected too often, it may never get close to its destination. Moscibroda and Mutlu [24], Fallin et al. [9] provide livelock freedom guarantee in a bufferless deflection routing, either with oldest-first arbitration [24] or golden packet [9] scheme. However, when combined with wormhole switching [25], the most widely adopted switching technique for NoC routers, it may not guarantee the livelock freedom any more. With wormhole switching, each packet is divided into several flow control units (flits). The head flit carrying the routing information governs the route. Routers arbitrate only among the head flits. As the head flit advances along the specific route, the body flits simply follow the reserved route and the tail flit will later release the route reservation. Due to this pipeline nature, it is much more likely for a packet to be deflected in wormhole-switched networks. More specifically, since arbitration is performed only for head flits, there is no chance for a packet to compete with another *in-flight* packet from other input port (*in-flight* means packets are being transmitting their body flits); deflection thus occurs. Such a case can happen to a packet in all routers, resulting in it never reaches its destination. That is, livelock happens. Livelock freedom in a wormhole-switched bufferless NoC is not proved in [9].

To deal with the livelock problem of deflecting bufferless routing in wormhole-switched networks, the authors in [24] recently propose worm truncation mechanism named BLESS-Worm. However, the worm truncation mechanism has introduced several potential problems [23]. First, besides the mapping information between output ports and allocated packets, each router needs to store the packet header information in case of truncation. Second, routers have to create head flits out of body flits at the time of truncation. Mean-while, routers have to notify the corresponding neighboring router, triggering the release of the output port mapping of the truncated packet. Third, additional wires between routers are required to transport routing information. Fourth, since flits are serviced one by one in the order of their rank, the truncated packet may be assigned the output port that has been already allocated to another packet, causing the latter one to be truncated. Consequently, a "domino effect" ensues, in which there is a sequence of packets where each packet truncates the next packet in the sequence. Last but most importantly, buffering at the receiver side is increased, which may outweigh the potential area saving. Since packets can be truncated, flits of the same packet may take different paths to reach the destination. Thereby, BLESS-Worm requires additional logic and buffering resources at destinations to reorder the out-of-order arriving flits of the same packet. The number of packets may be unbounded as packets can arrive in an interleaved manner. Thus the buffers should be sized for the worst case.

Based on the above analysis, we propose a new bufferless routing algorithm for wormhole-switched NoC, in which the notion of making-a-stop (MaS) is introduced. The main objective of MaS is to solve the livelock problem in wormhole-switched bufferless routing algorithm. By making a stop, there is no need to truncate packets, thus removing the large buffering requirements at the receiver and other overheads caused by worm truncation mechanism.

The remainder of the paper is organized as follows: Section 2 describes the MaS bufferless routing algorithm. Section 3 presents the microarchitecture of MaS router and comparisons between MaS and BLESS-Worm. Section 4 shows the simulation results in performance evaluation and a brief conclusion is drawn in Section 5.

2. Making-a-stop bufferless routing algorithm

In this section, we describe the new bufferless routing algorithm. Although the proposed routing algorithm is topology-agnostic, for better description, we consider a wormhole-switched 2D mesh NoC. By means of an example in Section 3.1, we show how our approach works. We then present a formal description of the MaS algorithm. Finally, we prove the proposed MaS are both deadlock- and livelock-free.

2.1. MaS outline

We first outline the basic idea of the proposed MaS algorithm. Suppose that, at time t, moving toward the destination node (1, 3), the head flit of packet P_1 arrives at the router of node (1, 1), requesting the east output port which brings it closer to the destination (see the snapshot in Fig. 1a). However, at this time, the east output port has been allocated to P_2 coming from the north input port and the west output port has been allocated to P_3.

Even though packet P_1 is the highest-ranked packet at this moment, it cannot get the desired east output port as packet P_2 is currently transmitting its body flits. In such case, in order to avoid livelock, packet P_1 makes a stop at the register array of router (1, 1), as shown in Fig. 2a (the corresponding snapshot is depicted in Fig. 1b). It should be emphasized that as the ranking of packets has locality property, the current highest-ranked packet may not be the highest-ranked one at the next moment and may be evicted from register array by the highest-ranked packet at that time.

Continuing with the example in Fig. 2a, at time $(t + 2)$, moving toward the destination node $(1, 0)$, the head flit of packet P_0 arrives at the router (1, 1). At this moment, both packets P_2 and P_3 are in-flight packets, indicating the east and west output port are still not available for other packets (see Fig. 1c). We assume that packet P_0 is the highest-ranked packet at time $(t + 2)$. Since the west output port has been allocated to P_3 and no truncation is allowed, P_0 cannot get the desired west output port. To avoid starvation of the highest-ranked packet, instead of being deflected, packet P_0 makes a stop. In order not to incur large area overhead, P_1 is evicted from the register array to vacate the register for the current highest-ranked packet P_0, and it is deflected to north output port as the east output port is not available (Fig. 2b). As the channel can forward only one flit once one time, the head flit of P_1 is evicted while the current arriving body flit takes over the place, resulting in a dynamic balance (the corresponding snapshot is shown in Fig. 1d).

2.2. MaS algorithm

MaS employs a minimal adaptive bufferless routing algorithm. With MaS, each packet is routed independently of every other through the network. In addition to destination information, each packet contains a priority information indicating its rank when contention happens. In [19], it shows that priority-based deflection policies using global or history-related criterion are beneficial in a deflection routing on-chip network. Based on this idea, we employ a global priority-based deflection policy which assumes there is a total *age* order among packets (e.g., *timestamp*) and ranks packets

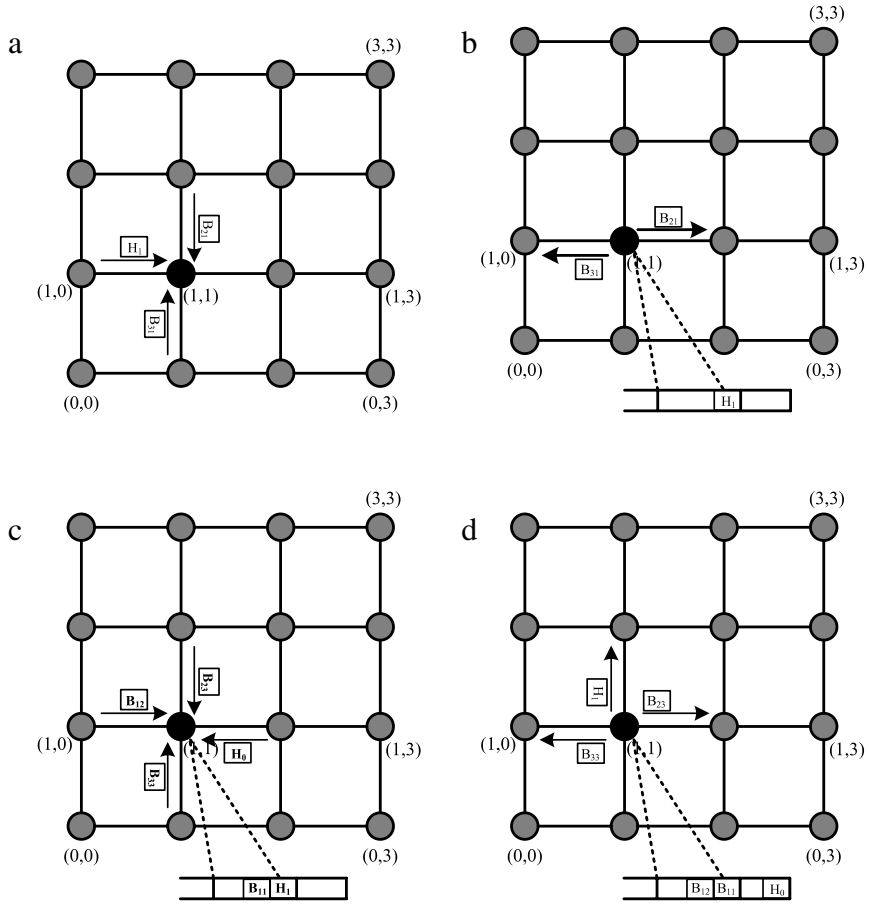


Fig. 1. Snapshots at time t and $(t+2)$. (a) At the beginning of time t. (b) At the end of time t. (c) At the beginning of time $(t+2)$. (d) At the end of time $(t+2)$.

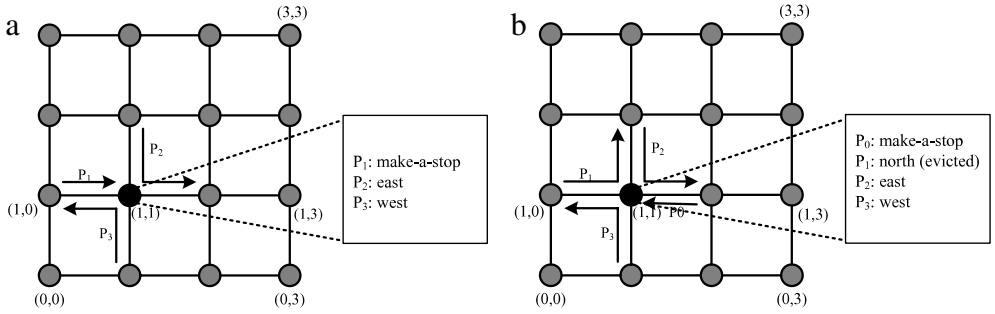


Fig. 2. An outline of MaS algorithm. (a) Arbitration at time t. (b) Arbitration at time $(t+2)$.

based on their age. The elapsed time that a packet has traversed in the network is regarded as the age of the packet. The *older* packets have higher priority than the *younger* ones in output port assignment. Once the head flit of a packet is assigned a specific output port, all subsequent flits simply follow the preceding flit. When the productive port(s) which results in the shortest path is not available, the highest priority packet makes a stop at the current router while the lower priority packets are deflected to any free output ports.

MaS algorithm distinguishes between the non-head flits and head flits. For a non-head flit, it is assigned the specific output port that has already assigned to its packet. For head flits, MaS algorithm ranks all head flits based on their age. Head flits are then serviced in the order of their ages. For a head flit, the router assigns it to the productive output port if the output port is available. If there are more than one productive output ports available, MaS algorithm randomly picks one. If no productive output port is free, MaS algorithm further distinguishes between the oldest head flit and non-oldest head flit. For the oldest head flit, it is allowed to make a stop at register array, which may force the current occupying flits to leave the register array. For a non-oldest head flit, it is assigned any free output port. In case no output port is free, the non-oldest head flit makes a stop (note that in such a case, the oldest head flit must have been assigned the productive output port). Formal description of MaS routing algorithm is given in Algorithm 1 below.

Note that in Algorithm 1, (c_x, c_y) is the current node address, and (d_x, d_y) is the destination node address carried in head flit f. e, w, n, s, and l represents east, west, north, south, and local output port, respectively.

2.3. Deadlock and livelock

When designing routing algorithm, it is critical to guarantee the absence of deadlock and livelock.

Algorithm 1 MaS algorithm

Input: Flit f.
Output: Port assignment p.

```
    function Δ = D(f)
 1: Δ ← Ø
 2: if (d_x > c_x) then
 3:     Δ ← Δ ∪ {n}
 4:     if (d_y > c_y) then
 5:         Δ ← Δ ∪ {e}
 6:     else if (d_y < c_y) then
 7:         Δ ← Δ ∪ {w}
 8:     end if
 9: else if (d_x < c_x) then
10:     Δ ← Δ ∪ {s}
11:     if (d_y > c_y) then
12:         Δ ← Δ ∪ {e}
13:     else if (d_y < c_y) then
14:         Δ ← Δ ∪ {w}
15:     end if
16: else
17:     if (d_y > c_y) then
18:         Δ ← Δ ∪ {e}
19:     else if (d_y < c_y) then
20:         Δ ← Δ ∪ {w}
21:     else
22:         Δ ← Δ ∪ {l}
23:     end if
24: end if
25: return Δ
    MaS algorithm procedure:
26: if f is head flit then
27:     productive port(s) Ω ← D(f)
28:     if ∃ p ∈ Ω, p unallocated then
29:         assign f to p
30:     else
31:         if f is the oldest flit then
32:             allow f to make a stop
33:             set p to empty
34:             if f' is currently occupying the register array then
35:                 f' is evicted
36:             end if
37:         else
38:             if ∃ p, p unallocated then
39:                 assign f to p
40:             else
41:                 allow f to make a stop
42:                 set p to empty
43:             end if
44:         end if
45:     end if
46: else
47:     assign f to the port p which is granted to its packet
48: end if
49: return p
```

Proposition 1. *The proposed MaS routing algorithm is deadlock-free.*

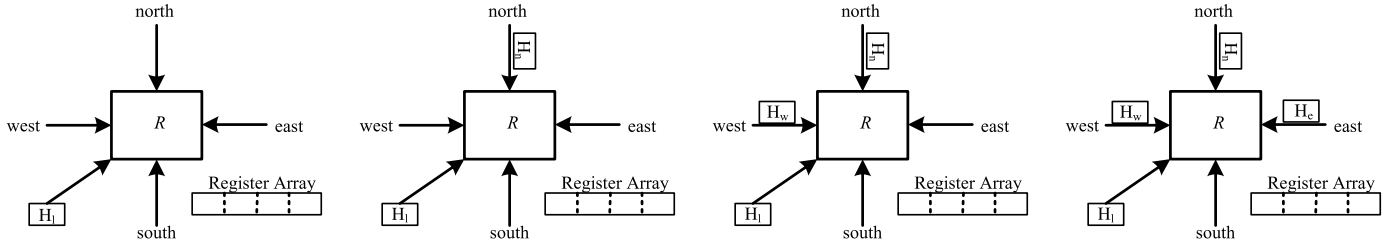
Proof. Since a node can inject a packet into its router if and only if at least one incoming channel is free and the register array is empty, we analyze the deadlock issue from two aspects.

- *Case* 1: the node injects a packet: Consider that $i(i < 4)$ packets enter the router at cycle 0 and the register array is empty. In this situation, the node is allowed to inject a packet into the router, as shown in Fig. 3. Fig. 4 indicates the port mapping in cycle 0. If the productive port is not available, a non-oldest packet is deflected to any other free output port while the oldest packet makes a stop at the register array. For simplicity, we assume there is no need to make a stop. Since packets can always find an output port to leave the router, no cyclic dependency can exist among the channels, and therefore the MaS is deadlock free. At cycle 1, the proofs for deadlock freedom with different number of incoming packets $i(i \in \{0, 1, 2, 3\})$ are similar to cycle 0. We need to pay special attention to the situation when all the incoming channels are busy, as shown in Fig. 5. In this situation, since there is no available output port for the new incoming packet, the new packet makes a stop in order not to be dropped. Again, no cyclic dependency can exist among the channels.
- *Case* 2: the node does not inject a packet: In this case, there are at most four incoming packets. Due to the deflecting policy, packets can always find an output port to leave the router. Thus, deadlock is free.

In any case, no cyclic dependency can exist among the channels, and then the MaS algorithm is deadlock free. □

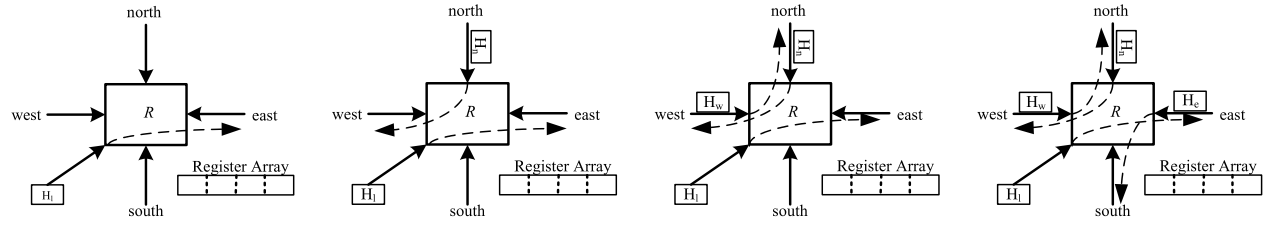
Proposition 2. *To avoid livelock, the size of register array should be K flits, where $K \in \mathcal{Z}^+$ is the maximum length of packet.*

Proof. For ease of presentation, we assume that all packets have the same length. Recall that a packet is evicted from the register array when the current highest priority packet does not get its productive output port. Let us consider the worst case: a packet P arrives at the router at cycle t, being the highest priority packet in the current router. Unfortunately, the desired output port has been allocated to other packet at cycle $(t-1)$; packet P thus makes a stop. However, at next cycle $(t+1)$, the packet P has to be evicted from register array as P is not the highest priority packet



(a) $i = 0$. (b) $i = 1$. (c) $i = 2$. (d) $i = 3$.

Fig. 3. With different numbers of incoming packets at cycle 0.



(a) $i = 0$. (b) $i = 1$. (c) $i = 2$. (d) $i = 3$.

Fig. 4. Port mapping at cycle 0.

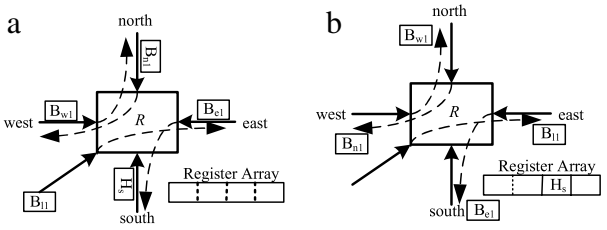


Fig. 5. (a) Snapshot at the beginning of cycle 1. (b) Snapshot at the end of cycle 1.

at cycle $(t+1)$. We denote $P_i > P_j$ to indicate that P_i has higher priority than P_j.

(a) Cycle 0: Without loss of generality, we first assume the head flit H_w of packet P_w from west input port arrives at the router R. It is assigned the north output port as no other packets compete for the port (Fig. 6a).

(b) Cycle 1: Coming from east input port, packet P_e arrives at router R, with higher priority than P_w (i.e., $P_e > P_w$). Assume packet P_e desires north output port. Since the north output port has been assigned to P_w at cycle 0 and P_w still has body flits (B_{w1}) not left the router R, P_e makes a stop at the register array (Fig. 6b). In this cycle, B_{w1} is forwarded through the north output port.

(c) Cycle 2: Coming from south input port, packet P_s arrives, with $P_s > P_e > P_w$, and requests the north output port. As P_w does not release the north output port, P_s, being the highest priority packet at this cycle, makes a stop at register array, forcing P_e to move out of the register array. Hence, P_e is deflected to the south output port (Fig. 6c). Note that as the channel capacity is one flit per cycle, the head flit H_e of P_e is forwarded through the south output port while new arriving body flit B_{e1} takes over the space of H_e, keeping a dynamic balance.

(d) Cycle 3: Coming from north input port, packet P_n arrives, with higher priority than P_s and desiring the south output port. As the south output port has been allocated to P_e at cycle 2, P_n makes a stop, causing P_s be evicted from the register array and deflected to west output port. Just as described at the previous cycle 2, B_{e1} is forwarded through the south output port and the new arriving body flit B_{e2} takes over the space of B_{e1}; head flit H_s is forwarded through the west output port and the new arriving body flit B_{s1} takes over the space of H_s (Fig. 6d).

(e) Cycle K: We assume P_n has the highest priority within the network. Thus, no packets can force P_n to get out of the register array. At this cycle, the register array contains K flits (Fig. 6e). After that, the register array will not maintain more that K flits (Fig. 6f).

With eviction policy, in any case, the number of flits residing in the register array will not exceed K. □

Based on the above analysis, two observations can be made as follows.

Observation 1. With deflecting strategy, instead of implementing virtual channels or turn forbidden [8], all arriving flits are enabled to find an output port to advance or make a stop, thus no cyclic dependency among the channels is formed. With oldest first ranking policy and introduction of making-a-stop, the oldest packet can always be assigned a productive direction immediately or in a few cycles, hence makes forward progress and eventually reaches its destination. A packet will become the oldest packet within the network eventually. Hence, freedom of deadlock and livelock is guaranteed.

Observation 2. Via eliminating router buffers, MaS enables power efficiency. By making a stop, no packet is truncated. Hence, flits of the same packet are transmitted in order. As a result, no large buffering is required to reorder the flits to enable in-order delivery. In this way, combining router buffers elimination, MaS achieves area efficiency.

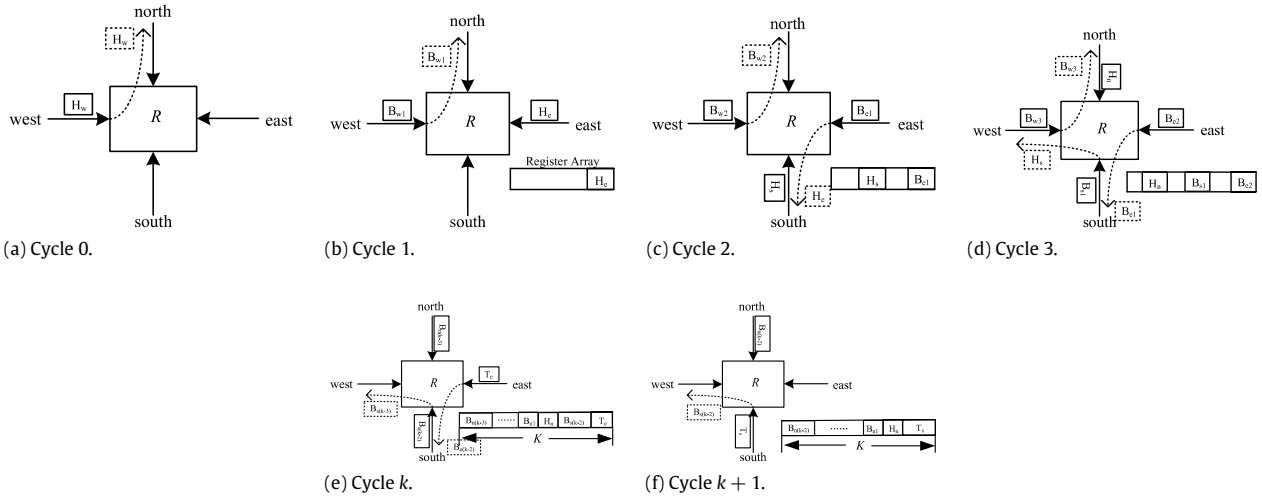
3. MaS router

The MaS router is designed to eliminate router buffers, limit additional receiver-side buffering requirements, and to provide livelock freedom for wormhole-switched deflecting bufferless routing.

3.1. Router microarchitecture

Fig. 7 shows the major components of MaS router. The MaS router consists of an arbitration unit, a register array, and a switch crossbar. Arbitration unit determines the output port to which a flit can be forwarded. Register array which in fact is a dynamic, multi-queue implementing the making-a-stop mechanism. It allows packets traveling within the network to make a stop when necessary. Switch crossbar is responsible for forwarding the flit to the appropriate output port. In order to avoid large switch crossbar which consumes large area, muxes are added to connect the register array to switch crossbar. The inputs to the muxes can be classified as either register array inputs (e.g., inputs from register array) or direct inputs (e.g., inputs from the neighboring routers).

To avoid deflecting packets to ejection ports, a node can inject a flit into its router only when the register array is empty and at least one incoming channel is free.



(a) Cycle 0. (b) Cycle 1. (c) Cycle 2. (d) Cycle 3.

(e) Cycle k. (f) Cycle $k+1$.

Fig. 6. Snapshots at different cycles.

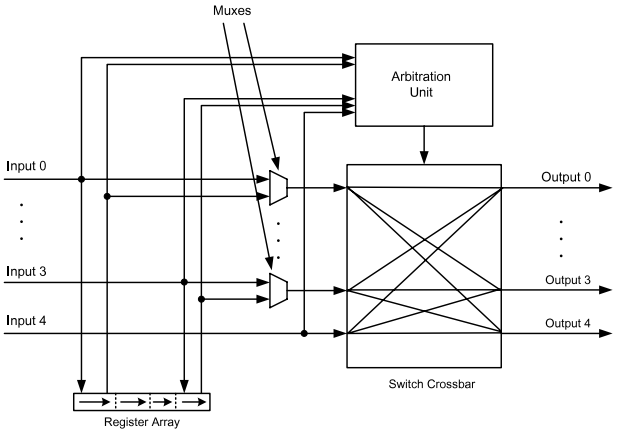


Fig. 7. Microarchitecture of MaS router.

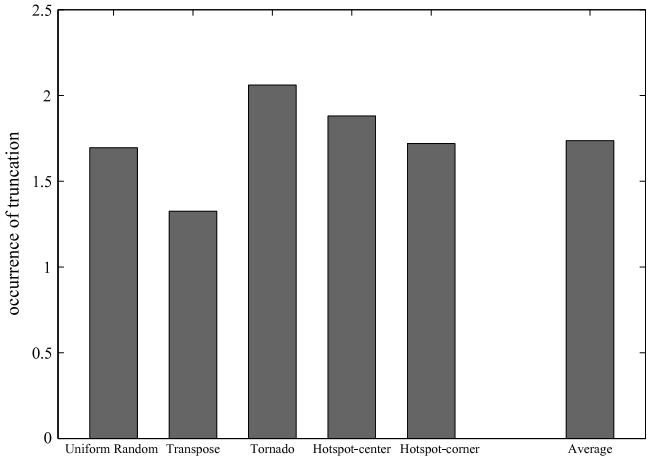


Fig. 8. Occurrence of truncation under different traffic scenarios.

3.2. Comparison between MaS and BLESS-Worm

As far as we know, BLESS-Worm is the recently proposed bufferless routing algorithm which considers both bufferless routing and wormhole switching. In this subsection, we compare the proposed MaS with BLESS-Worm in terms of channels, buffering requirements at receiver-side, design complexity, and worst-case execution time.

Channels: With BLESS-Worm, since worm truncation technique is adopted, packets may have to be split under collision. In the worst case, BLESS-Worm has to route all flits of the same packet independently. Therefore, every flit needs to contain routing information. BLESS-Worm uses additional wires between routers to transport routing information, which consume dynamic energy when wires are activated and static energy when wires are idle. Moreover, each node is subject to a fixed pin limitation, thus increasing the number of channels leads to narrow channel width. Thus, serialization latency is induced by squeezing a large packet through a narrow channel. In contrast, in MaS, packets are never truncated and can always be routed in their entirety. Hence, there is no need to narrow channel width for the separate header bits.

Buffering requirements: Fig. 8 shows the occurrence of truncation before network saturates in a 2D 10 × 10 NoC, with routers implementing the BLESS-Worm routing algorithm. In particular, with the term "occurrence of truncation", we hereby mean the relationship between the number of worm truncation happens and the total number of packets, i.e., the average truncation occurrence per packet. As can be observed, on average, the occurrence of truncation is over 1.7 (i.e., on average, a packet needs to be truncated twice before reaching its destination). As a result, unbounded extra logic and buffers are needed to reorder the out-of-order arriving flits of the same packet for BLESS-Worm. However, in MaS, as packets are not divided, flits of the same packet can be guaranteed to remain in order. Thus, no additional buffering is required at receiver side to reorder the flits.

Design complexity: The BLESS-Worm router needs to create head flits out of body flits at the time of truncation. As a result, extra logic is needed to maintain not only allocation information but also the packet header information from the original head flit of a packet. In addition, a sequence number is needed to reorder the flits of a packet if they get out of order in transit. While in MaS, a fixed-size register array is implemented in a dynamical, multi-queue fashion at each router. In addition, since packets will arrive in order, no sequence number is required.

Worst-case execution time: Here, we simply consider an "unlucky" packet to analysis the worst-case execution time. An "unlucky" packet is the highest-rank packet with its productive output port(s) is always allocated to other packet just before the "unlucky" packet arrives at the router. In BLESS-Worm, the "unlucky" packet is able to get the desired output port immediately by truncating other packet; while in MaS, the "unlucky" packet needs to making-a-stop at every intermediate router, thus it may need more time to reach destination at such worst-case. However, the "unlucky" packet in MaS does not impact other packets so much as in BLESS-Worm. In the latter, more packets may be involved into truncation, as the truncated packet may truncate other packets, spreading the truncation scope. While in MaS, only the "unlucky" packet is involved.

Table 1
Network configuration.

Network size	10 × 10 mesh
Clock frequency	2 GHz
Link length	128 bits
Switching technology	Wormhole
Router latency	2 cycles
Channel latency	1 cycle
Flit size	128 bits
Packet size	8 flits

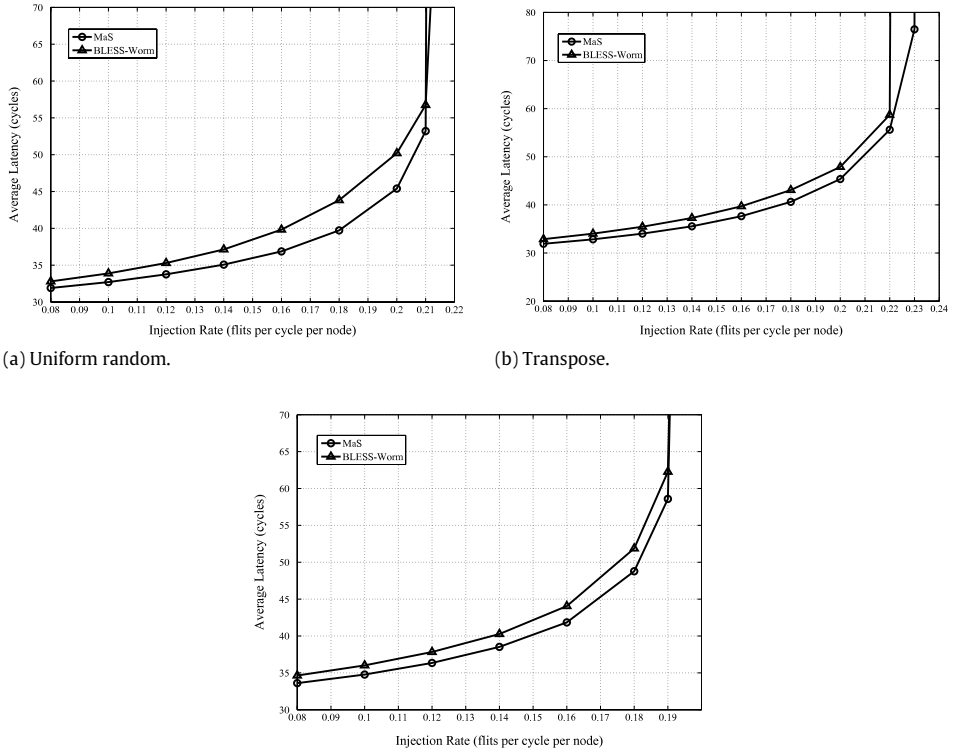
4. Performance evaluation

4.1. Evaluation methodology

The performance of MaS has been evaluated using a flit-level, cycle-accurate on-chip network simulator based on the booksim [4]. The modeled on-chip network configuration is 2D mesh topology with single-cycle channels. A baseline router has five input ports and five output ports; while an MaS router has five input ports, five output ports, and a dynamic, multi-queue register array. Within each simulation for a synthetic traffic there is a warm-up period of 100,000 cycles. Thereafter, 10,000 packets are injected per node and the simulation continues until these packets in the sample space are all received. We define that on-chip network saturates when average packet latency is over twice of zero-load latency. Table 1 specifies the configuration parameters in our performance evaluation.

We use three different traffic patterns: uniform random (UR), transpose (TR), and hotspot (HS). UR traffic assumes each node uniformly injects packets to randomly distributed destinations in the network. In TR traffic, the node (i, j) only communicates with node (j, i). In HS traffic, four hotspot nodes are located at the center of the network. The 80% of traffic is sent to randomly distributed destinations in the network, while the remaining traffic is to the hotspot nodes.

Packet latency is calculated from the time when the first flit of a packet is generated, to the time when its last flit is ejected at



(a) Uniform random.

(b) Transpose.

(c) Hotspot.

Fig. 9. Average packet latency under different traffic patterns.

the destination, including source queueing time. In BLESS-Worm, as packets may have to be truncated when congestion arises, flits of the same packet may follow different paths to reach the destination. That is, one flit may be deflected at current router and be deflected again at the next router. From statistics point of view, it is more likely that one flit is delayed or takes a detour, thereby delaying the entire packet. Hence, BLESS-Worm can have a negative impact on packet latency. However, a packet is treated as an entirety in MaS. Thus once the head flit of the packet is allocated to a productive output port, all subsequent flits follow the head flit and make forward toward the destination with no flits deflecting at the current router.

The power model is based on the 32 nm process technology similar to [1,23]. More precisely, the power consumption, measured using the P_{flit} metric, is defined as the power needed to transmit one flit from the source to the destination. P_{flit} is given by $P_{flit} = P_{C_{flit}} + P_{S_{flit}}$, where $P_{C_{flit}}$ represents the power consumed by flits traversing through the channels while $P_{S_{flit}}$ represents the power needed by the switches in the router. In MaS, we further model the power consumed by register array. While in BLESS-Worm, additional wires are dedicated for header transmission, which incurs extra overhead such as power consumption. However, this overhead is not modeled in our evaluation, giving BLESS-Worm a small advantage over MaS.

Since bufferless NoC is considered, we roughly estimate the area overhead of buffers at the receiver in terms of flits using a simple model: $S_{buf} = N_{buf} \times N_R$, where N_{buf} is the number of receiver side buffers per node and N_R is the number of routers. BLESS-Worm eliminates input buffers of router at the cost of increasing receiver side buffers. MaS also gets rid of input buffers and only needs a (first-in first-out) FIFO buffer per input port at receiver as well as a register array.

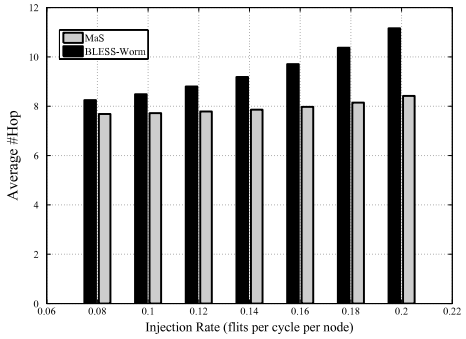
In addition, average maximum buffering requirement is the average over several maximum buffering requirements observed in different traffic patterns, which aims to avoid biasing the workload in favor of one architecture or traffic pattern. Recall that a packet can be truncated when necessary in BLESS-Worm, flits from the same packet may take independent routes and arrive at significantly different points in time at the destination. Thus flits of the same packet can arrive in out-of-order, which increases buffering requirements at the receiver to support in-order delivery of packets. Since packets can arrive interleaved, the buffering resources which are needed to reorder the out-of-order flits of the same packet can be unbounded. The potential cost savings by eliminating input buffers may be outweighed by the increasing buffering requirements at receivers. However, a packet is routed as a complete entity in MaS, thus reducing buffering requirements at receiver side. The receiver in MaS approach only needs an FIFO buffer per input port, which effectively reduces buffering requirements at the receiver.

4.2. Simulation results

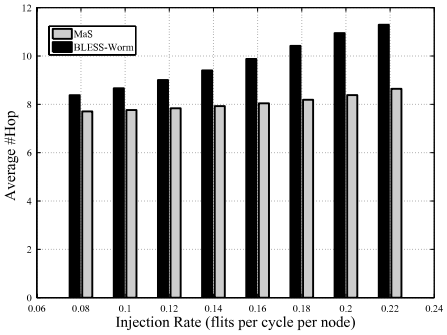
In this section, we compare MaS with the recent proposed deflecting bufferless routing BLESS-Worm [24] in terms of average packet latency, average hop count, power consumption, and average maximum buffering requirement at the receiver side.

Fig. 9 depicts average packet latency as a function of injected rate under different synthetic traffic patterns. Clearly, MaS outperforms BLESS-Worm in average packet latency under all levels network traffic loads. The latency reduction of MaS as compared to BLESS-Worm is up to 10% (UR), 6% (TR), and 6% (HS), respectively. The reason is that packets may be segmented under congestion with BLESS-Worm. Therefore different segments of the same packet may take different paths reaching the destination. Obviously, with more segments, it is more likely that one segment takes detour thereby delaying the whole packet.

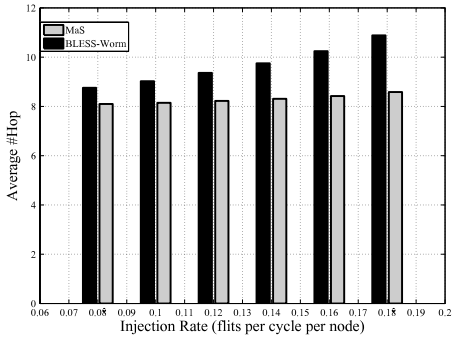
Fig. 10 shows the average hop count per packet as a function of injected rate under different synthetic traffic patterns before network saturates. It can be observed that the average hop count



(a) Uniform random.

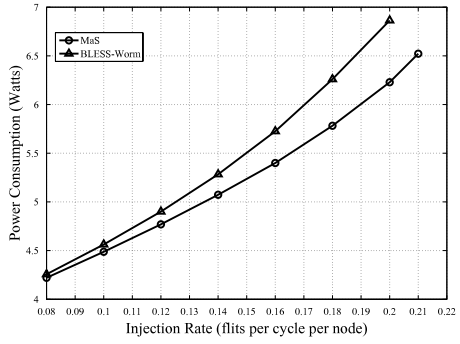


(b) Transpose.

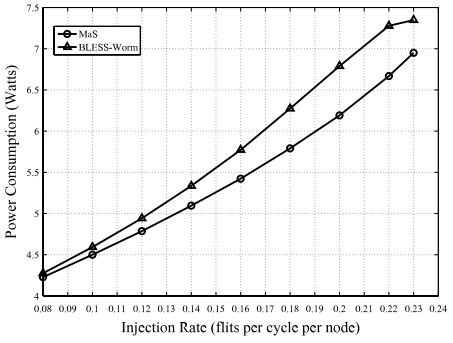


(c) Hotspot.

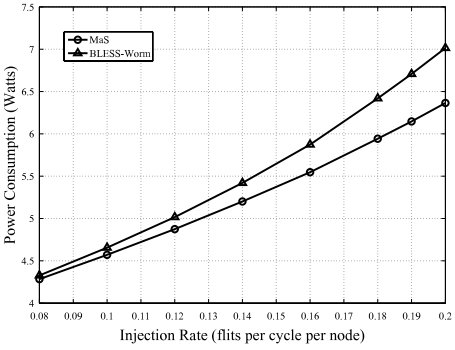
Fig. 10. Average hop count under different traffic patterns.



(a) Uniform random.



(b) Transpose.



(c) Hotspot.

Fig. 11. Power consumption under different traffic patterns.

is quite a constant with MaS as traffic increases while it climbs quickly with BLESS-Worm as network approaches saturation. More precisely, the average hop count reduction of MaS over BLESS-Worm is up to 25% (UR), 24% (TR), and 23% (HS), respectively. Due to the existence of register array, packets are always routed in their entirety which reduces the deflecting probability. Note that at least two extra hops are induced by one deflection. As a result, lower deflecting results in lower hop count.

Fig. 11 indicates power consumption as a function of injected rate under different synthetic traffic patterns. It can be observed

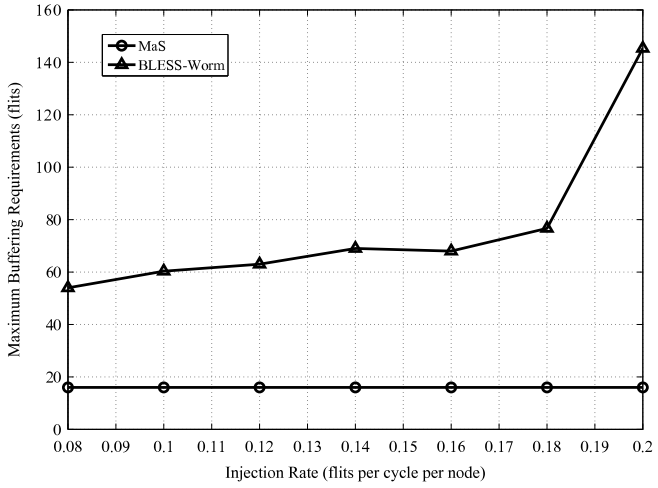


Fig. 12. Maximum buffering requirements at receiver side per input port.

that power consumption in MaS approach is lower than that in BLESS-Worm. The reduction of power consumption of MaS against BLESS-Worm is up to 9% (UR), 9% (TR), and 8% (HS), respectively. As routers and links are contributors to power consumption, lower hop count results in lower power consumption.

We plot the average maximum buffering requirement in flits at receiver as traffic increases in Fig. 12. BLESS-Worm increases buffering requirements at the receiver due to truncations. The worm truncation mechanism enables a packet be truncated into several parts. Each part of a packet arrives at different time, which increases buffering requirements to support in-order delivery. It can be observed from Fig. 12 that in BLESS-Worm the average maximum buffering requirement arises rapidly as network is approaching saturation. In the case where injection rate is 0.08, we find that MaS reduces the buffering requirements by about 70% compared to BLESS-Worm, while the reduction of buffering requirements arise up to 80% when network gets close saturated.

5. Conclusion and future work

To address the issues of area and power in on-chip network, in this paper, we have proposed an area-benefit approach MaS for bufferless on-chip network, which enables in-order delivery without large buffering requirements at receiver side while maintaining the energy efficient. Compared to recent proposed deflecting bufferless routing BLESS-Worm, MaS reduces the increasing buffering requirements at receiver side which is caused by out-of-order arriving. In addition, the deadlock and livelock freedom of MaS algorithm have been proved. Extensive cycle-accurate simulations have been conducted to show that MaS delivers a better performance compared to BLESS-Worm. Results show that MaS lowers average packet latency by 10% and power consumption by 9%, respectively. Moreover, it reduces buffer requirements at the receiver by up to 80%. For future work, the analysis of real traffic will be conducted. In addition, the routing algorithm should be evaluated in other priority ranking policies such as most deflecting-first method.

Acknowledgments

This work is supported in part by the National Natural Science Foundation of China under Grant No. 61073055 and 211-III fund under Project 3226301.

References

[1] J. Balfour, W.J. Dally, Design tradeoffs for tiled CMP on-chip networks, in: Proceedings of the International Conference on Supercomputing, 2006, pp. 187–198.
[2] P. Baran, On distributed communications networks, IEEE Transactions on Communications (1964).
[3] L. Benini, G.De. Micheli, Networks on chips: a new SoC paradigm, IEEE Transactions on Computer 35 (1) (2002) 70–78.
[4] URL: http://nocs.stanford.edu/cgi-bin/trac.cgi/wiki/Resources/BookSim.
[5] W.J. Dally, Virtual-channel flow control, Parallel and Distributed Systems, IEEE Transactions on 3 (2) (1992) 194–205.
[6] W.J. Dally, C.L. Seitz, Deadlock-free message routing in multiprocessor interconnection networks, IEEE Transactions on Computers 100 (5) (1987) 547–553.
[7] W.J. Dally, B. Towles, Route packets, not wires: on-chip interconnection networks, Proceedings of Design Automation Conference (2001) 684–689.
[8] W.J. Dally, B.P. Towles, Principles and Practices of Interconnection Networks, Morgan Kaufmann, 2004.
[9] C. Fallin, C. Craik, O. Mutlu, CHIPPER: a low-complexity bufferless deflection router, in: Proceedings of the 17th International Symposium on High Performance Computer Architecture (HPCA), 2011, pp. 144–155.
[10] C. Gomez, M.E. Gomez, P. Lopez, J. Duato, Reducing packet dropping in a bufferless NoC, in: Proceedings of International Euro-Par Conference on Parallel Processing, 2008, pp. 899–909.
[11] P. Gratz, C. Kim, R. Mcdonald, S.W. Keckler, D. Burger, Implementation and evaluation of on-chip network architectures, in: Proceedings of International Conference on Computer Design, 2006, pp. 477–484.
[12] M. Hayenga, N.E. Jerger, M. Lipasti, SCARAB: a single cycle adaptive routing and bufferless network, in: Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture, 2009, pp. 244–254.
[13] J. Hu, R. Marculescu, Application-specific buffer space allocation for networks-on-chip router design, in: Proceedings of IEEE/ACM International Conference on Computer Aided Design, 2004, pp. 354–361.
[14] International Technology Roadmap for Semiconductors Interconnect,
[15] A. Jantsch, H. Tenhunen (Eds.), Networks-On-Chip, Kluwer Academic Publishers, 2003.
[16] N.E. Jerger, Li-Shiuan Peh, On-Chip Networks, Morgan & Claypool Publishers, 2009.
[17] J. Kim, Low-cost router microarchitecture for on-chip networks, in: Proceedings of the 42nd IEEE/ACM International Symposium on Microarchitecture, 2009, pp. 255–266.
[18] A.K. Kodi, A. Sarathy, A. Louri, iDEAL: inter-router dual-function energy and area-efficient links for network-on-chip (NoC) architercure, in: Proceedings of the 35th International Symposium on Computer Architecture, 2008, pp. 241–250.
[19] Z. Lu, M. Zhong, A. Jantsch, Evaluation of on-chip networks using deflection routing, in: Proceedings of ACM Great Lakes Symposium on VLSI, 2006, pp. 296–301.
[20] R. Marculescu, U.Y. Ogras, L.-S. Peh, N.E. Jerger, Y. Hoskote, Outstanding research problems in NoC design: system, microarchitecture, and circuit perspectives, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD) 28 (1) (2009) 3–21.
[21] T. Meincke, et al. Globally asynchronous locally synchronous architecture for large high-performance ASICs, in: Proceedings of IEEE International Symposium on Circuits and Systems, vol. 2, 1999, pp. 512–515.
[22] G. Michelogiannakis, J.D. Balfour, W.J. Dally, Elastic-buffer flow control for on-chip networks, in: Proceedings of the 15th International Symposium on High Performance Computer Architecture, 2009, pp. 151–162.
[23] G. Michelogiannakis, D. Sanchez, W.J. Dally, C. Kozyrakis, Evaluating bufferless flow control for on-chip networks, in: Proceedings of the 4th ACM/IEEE International Symposium on Networks-On-Chip, 2010, pp. 9–16.
[24] T. Moscibroda, O. Mutlu, A case for bufferless routing in on-chip networks, in: Proceedings of the 36th International Symposium on Computer Architecture, 2009, pp. 196–207.
[25] L.M. Ni, P.K. McKinley, A survey of wormhole routing techniques in direct networks, IEEE Computer 26 (2) (1993) 62–76.
[26] L. Shang, L.-S. Peh, N.K. Jha, Dynamic voltage scaling with links for power optimization of interconnection networks, in: Proceedings of the 9th International Symposium on High-Performance Computer Architecture, 2003, pp. 91–102.
[27] M.B. Taylor, et al. Evaluation of the raw microprocessor: an exposed-wire-delay architecture for ILP and streams, in: Proceedings of the 31st Annual International Symposium on Computer Architecture, 2004, pp. 2–13.
[28] S.R. Vangal, et al., An 80-tile sub-100-W TeraFLOPS processor in 65-nm CMOS, IEEE Journal of Solid-State Circuits 43 (1) (2008) 29–41.

Jing Lin received the B.S. and M.S. degrees in computer science and electrical engineering from Nanjing University of Posts and Telecommunications, Nanjing, China, and the Ph.D. degree in computer software and theory from Sun Yat-sen University, Guangzhou, China. Her research interests include on-chip networks as well as interconnection networks.

Xiaola Lin received the B.S. and M.S. degrees in computer science from Peking University, Beijing, China, and the Ph.D. degree in computer science from Michigan State University, East Lansing, Michigan. He is currently a professor of computer science in the School of Information Science and Technology, Sun Yat-sen University, Guangzhou, China. His research interests include parallel and distributed computing and computer networks.

Liang Tang received the B.S. and M.S. degrees in computer science and electrical engineering from Nanjing University of Posts and Telecommunications, Nanjing, China, and the Ph.D. degree in control engineering from University of Science and Technology of China, Hefei, China. He is now a research assistant in Broadband Wireless Mobile Communications Research Laboratory, Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences. His current research interests include multimedia communications and wireless network optimization techniques.