A Flit-level Speedup Scheme For Network-on-Chips Using Self-Reconfigurable Bi-directional Channels

Zhiliang Qian, Ying Fei Teh and Chi-Ying Tsui
VLSI Research Laboratory, Department of Electronic and Computer Engineering
The Hong Kong University of Science and Technology, Hong Kong, China
Email: {qianzl, yingfei, eetsui}@ust.hk

***Abstract*—In this work, we propose a flit-level speedup scheme to enhance the network-on-chip(NoC) performance utilizing bidirectional channels. In addition to the traditional efforts on allowing flits of different packets using the idling internal and external bandwidth of the bi-directional channel, our proposed flit-level speedup scheme also allows flits within the same packet to be transmitted simultaneously on the bi-directional channel. For inter-router transmission, a novel distributed channel configuration protocol is developed to dynamically control the link directions. For the intra-router transmission, an input buffer architecture which supports reading and writing two flits from the same virtual channel at the same time is proposed. The switch allocator is also designed to support flit-level parallel arbitration. Simulation results on both synthetic traffic and real benchmarks show performance improvement in throughput and latency over the existing architectures using bi-directional channels.**

***Index Terms*—Bidirectional channel, flit-level speedup, NoC**

I. Introduction and Related Work

With technology scaling down, hundreds of cores can be integrated on a single chip and future System-on-Chip (SoC) will be predominantly communication-centric [1], [2]. Network-on-Chip (NoC) has emerged as a reliable and modular solution to support the high bandwidth communication requirement [1].

The bandwidth provided by NoC and its utilization have a significant impact on the overall performance of the system [3]. The NoC bandwidth can be divided into two types: the inter-router bandwidth (channel bandwidth) and the intra-router bandwidth (switch bandwidth). In a typical NoC design, the channel bandwidth is determined by the unidirectional links between the routers. For the intra-router bandwidth, it is dictated by the datapath in the router which comprises of the input buffers, the $N \times N$ crossbar fabric (N is the number of input ports) and the output registers. In [3], NoC with input speedup is proposed by providing some excess bandwidth in the crosspoint ($2N \times N$ crossbar for a 2X input speedup) to avoid inefficient usage of the internal bandwidth and idling of the channel resources [3].

Recently, motivated by the observation that the traffic distribution across NoC is heterogeneous [4] and quite often one uni-directional link is overflowed with heavy traffic while the opposite link remains idle, NoC designs employing bidirectional channels have been proposed [2][5][6]. In [2], the link directions are configured in the architecture level with route re-allocation to meet the bandwidth and QoS requirement. In [5][6], NoC architectures with dynamically reconfigurable bi-directional links were proposed. The direction of each link is reconfigured at run time depending on the traffics of the two routers involved using either an external bandwidth arbiter [6] or a channel direction control (CDC) protocol [5]. For both approaches, the link direction is reversed if there are more than two packets requesting to traverse along one direction and no packet requesting in the opposite direction.

978-3-9810801-8-6/DATE12/©2012 EDAA

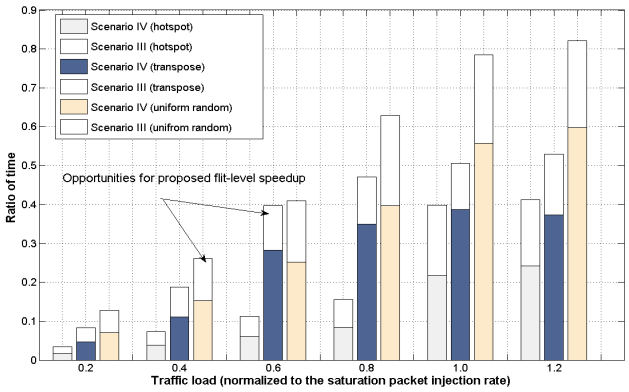


Figure 1: The occurrence of scenario III and IV in profiling

In this work, we further improve the NoC performance by adding flexibility in using the bi-directional links. Our proposed design is based on the following observations:

- For a channel between two routers, when sending a packet from one end to the other, there is a significant portion of time that no packet is coming from the opposite direction [5]. Through our profiling on a 8×8 mesh NoC architecture, the average percentage of time is as high as 67.50% for most traffics even when the injection rate is 80% of the saturation rate.
- Under the condition that there is no packet coming from the opposite direction on a particular link, we can further distinguish into four scenarios for the router. They are: I) the router has no packet to send through the link, II) the router has one packet with a single flit to be sent through the link, III) the router has one packet with multiple flits to be sent through the link, IV) the router has multiple packets requesting the output link. For the last two scenarios, we can increase bandwidth and hence the performance if we can reverse the direction of the opposite link and send two flits through the two links at the same time. Figure 1 shows the portion of time that scenario III and IV occur among all of the above four scenarios under various injection rates and traffic patterns

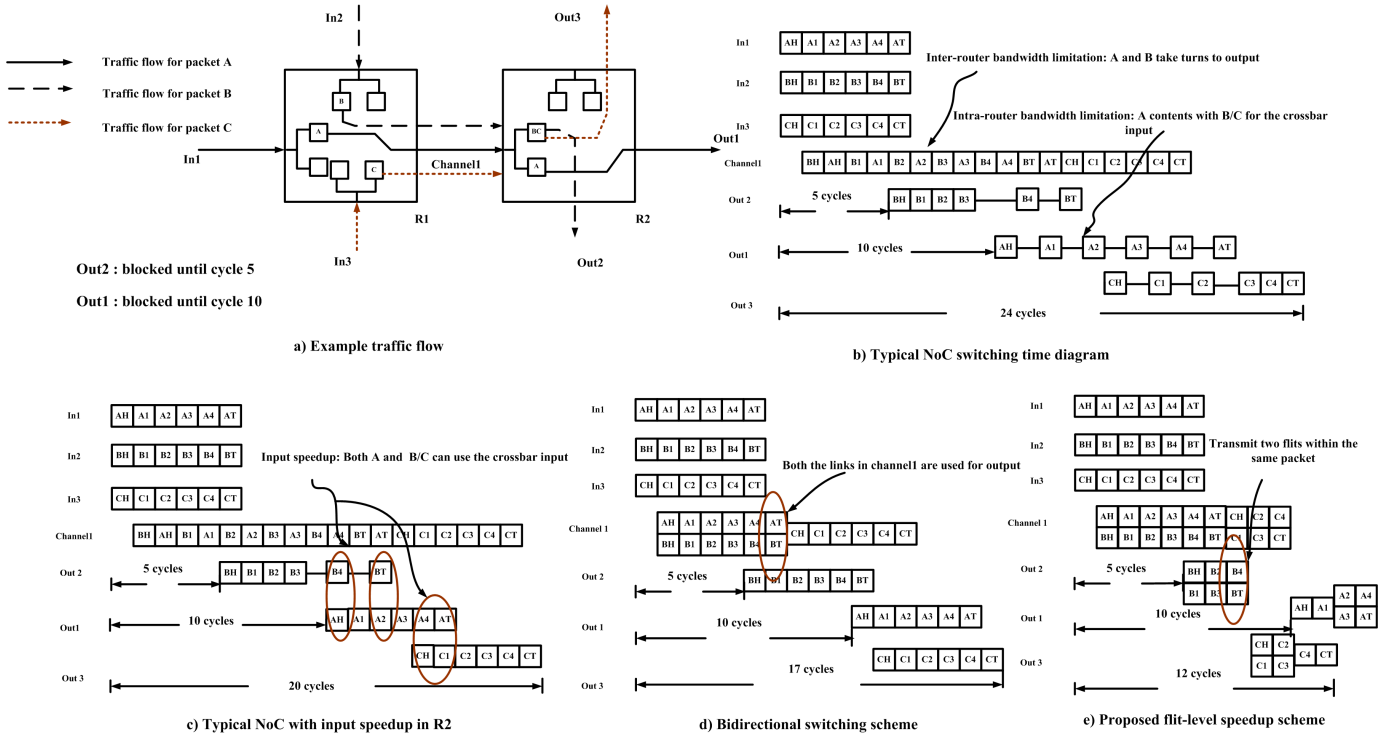


Figure 2: A motivational example

for a 8×8 mesh NoC. As shown in the figure, scenario III occurs frequently at run time. However the state-of-the-art bidirectional switching schemes will only work for scenario IV since the routers only support parallel transmission of two flits from different packets [5][6].

The observations above show that there is room for performance improvement by utilizing the double capacity bi-directional channels for scenario III. In this work, we propose a flit-level speedup scheme for improving the NoC performance. For the inter-router transmission, bidirectional links are employed to provide a double bandwidth at run time. For the intra-router transmission, we propose a novel input buffer architecture and a switch allocator design to allow flits within the same packet to participate in the routing pipelines.

II. Motivational Case Study

We consider a case of three packets A, B, C traversing from router R1 to router R2 as shown in Figure 2-a. Each packet is six flits long including the header and the tail flit. We adopt a fair round-robin arbitration policy and assume at the beginning, packets A and B successfully acquire the two virtual channels in router R2. We assume packet stalling [3] occur in R2 and the blockage times for packets A and B in ports Out1 and Out2 are 10 and 5 cycles, respectively. Figure 2-b is the timing diagram if we use a traditional NoC architecture. In the figure, we can observe two kinds of bandwidth limitations. The first is the inter-router bandwidth limitation, where in the traditional NoC, only one unidirectional link is available. Packets A and B have to alternatively use this single capacity link. The second is the intra-router bandwidth limitation where in R2, the two virtual channels in the west input need to compete in the switch allocation phase, which means that packet A and B/C can not be transmitted at the same time. The intra-router bandwidth limitation can be overcome by adopting input speedup in R2 as shown in Figure 2-c. Both virtual channels in R2 can use the crossbar and thus the latency of packet C is reduced to 20 cycles. If bidirectional NoC switching scheme is applied as shown in Figure 2-d, both packets A and B are transmitted at the same time using the double capacity bi-directional link and the delay of packet C can be further reduced to 17 cycles. Figure 2-e is our proposed flit-level speedup scheme which allows flits within the same packet (A,B,C) to be transmitted simultaneously. In R2, the latency of packets A and B are reduced due to a shorter transmission time while the latency of packet C is improved because its waiting time is greatly reduced since packet B in front of it leaves earlier. In this example, these two effects further improve the latency of the three packets by 12.5%, 27.2% and 29.4%, respectively, compared with the existing bidirectional switching scheme.

III. Flit-level Speedup NoC

In this section, we present the details of our proposed flit-level speedup scheme. The channel direction control protocol will be described first. Then we discuss the modification of the input buffer and the switch allocator to support flit-level parallel transmission and arbitration in the router. For the issue of deadlock and starvation avoidance as well as other datapath components design, the readers can refer to [5][6] for details.

A. Inter-router channel direction control

1) Master and slave link : Both links connecting to a router can be reconfigured as sending or receiving. However for each

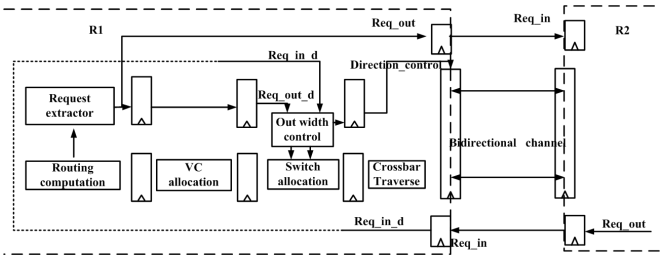


Figure 3: Channel direction control module

router we define one link as the master and the other as the slave. If there are traffics in both direction, the master and slave links act as the sending and receiving links, respectively. Under flit-level speedup, when writing two flits from the same VC into the output channel, the output controller will always put the first flit on the master link and the second flit on its slave link. On the receiving side, we always assemble the flit appeared on the slave link ahead of the flit on the master link into the virtual channel.

2) Channel direction control protocol: The channel direction control protocol is used to configure the bidirectional link at run time based on the traffic condition while at the same time guarantee there is no writing conflicts i.e. writing into the link from both ends. There are two different ways to do the control. [6] employs a central control while [5] uses a distributed control mechanism. In this work, we also employ a distributed architecture since it is more modular and easier to implement. Therefore we mainly compare our architecture with that used in [5].

In [5], at each router there are two Finite State Machines (FSMs), one for the master link and the other for the slave link. The direction of each link depends on the state of the corresponding FSM. Since the incoming request signal from the neighboring router takes two cycles to arrive at the current router, a "wait" state is required when the FSM transits from a "receiving" state to a "sending" state which will introduce at least one dead cycle in the channel direction reversal process.

In this work, we propose a novel channel control scheme based on a four-stages pipeline NoC which does not use FSMs. The direction of the bi-directional link is determined by the pressure signal of the local router (pressure signal represents the number of flits that the router requests to send out at the channel) and that of the neighboring router. The block diagram of the overall architecture is shown in Figure 3. A request extractor is added at the routing computation stage to generate the pressure signals and an output width controller is added to work with the switch allocator to process the pressure signals and determine the channel direction accordingly. At the routing computation stage, the request extractor records the number of packets that have won the neighbor's virtual channels. The signal is generated at the routing computation stage but not the VC allocation stage because the pressure signal needs two cycles to arrive at the neighboring router (assuming all interconnections between two adjacent routers are doubly registered [5]) and it is required to be used in the switching allocation stage. The effect of generating the signal in the routing computation stage is that the VC allocation result is updated one clock cycle later when the packet first enters the VC. However it only affects the performance when the neighboring router requests to send two flits, while the newly arrived packet wins the VC allocation but it is not yet updated in the routing computation stage. In this case, the local router assumes there is no request for sending and the link will then be configured as receiving for both channels while it should be configured as 1 sending and 1 receiving. The correctness of the router operation is not compromised, only some of the priority of sending out a flit in the local router is lost in this special case. From the simulation results, it is shown that there is no significant difference in the latency and throughput when the pressure signal is generated at the routing computation stage comparing with that generated at the VC allocation stage. The request extractor calculates the local pressure signals "Req_out" for each output channel, passes them to the local switch allocators and at the same time sends them out to the neighboring routers as their "Req_in" signals. Two cycles later, at the switch allocation stage, this local pressure signal and the pressure signals received from the neighboring router are used by the output width controller to decide the allocation of the bandwidth of the corresponding channel. Together with the switch allocator, the bi-directional links are re-configured accordingly.

The operation of the request extractor is described in Algorithm 1. It tries to acquire a two flits channel when either there are multiple virtual channel buffers having flits to be sent (line 9-10 in Algorithm 1) or there is one packet with multiple flits in a VC channel buffer requesting to be sent out at the output channel (line 13-15 in Algorithm 1).

In our scheme, since the output width is determined every cycle in the switch allocation phase, no dead cycle is needed. However, we need to guarantee the flits, after winning at the switch allocations, can traverse the crossbar and link successfully without any stall to guarantee no writing conflict occurring. Hence, we need to ensure there are enough buffer slots in the downstream virtual channel to hold the flits. Since our scheme supports transmitting two flits into the same virtual channel at one time and the VC buffer full information of the adjacent router takes 3 cycles to reach the local router, the minimum buffer slots required is thus equal to 6.

Algorithm 2 describes the operation of the output width controller. It dynamically configures the data width of the switch output (0, 1 or 2 flits) and the input switch arbitration mode. Input switch arbitration mode indicates whether the two flits participating in the arbitration are coming from the same VC buffer. For example, if the local pressure signal indicates that a 2-flit channel for the output is requested while the neighboring pressure signals indicates that there is no flow in the opposite direction, the controller will enable both the master and the slave link channels to participate in the switch arbitration (line 9-11). Under this condition, if the 2-flit channels are requested by more than one packet, the input switch arbitration mode for these packets is set to "0"

Algorithm 1 Operation of request extractor

Input : Buffer status $BF\_Status[N][V]$ and Credits info $Credit\_in[N][V]$
Container : Request bank on each output $Req[N]$
Output: Local pressure information $Req\_out[N]$
1 For each input port $i \in N$
2 $Req[i].clear()$ // Initialize the local pressure counter
3 For each VC buffer $j \in V$
4 If $BF\_Status[i][j].VC\_status = Assigned$
5 $k = BF\_Status[i][j].Get\_output\_port()$;
6 $t = BF\_Status[i][j].Get\_output\_vc()$
7 $Req[k].Add\_request([i, j])$ // Adding requests to the output port
// output the pressure signal to the neighbor
8 For each output port$o$ in N
9 If $Req[o].Size() >= 2$
10 $Req\_out[o] = 2$
11 Else If $Req[o].Size() == 1$
12 $(i, j) = Req[o].Get\_request()$
13 If $BF\_Status[i][j].buffer\_occupancy > 1$
14 If $BF\_Status[i][j].head\_flit.flit\_type! = Flit\_Tail$
15 $Req\_out[o] = 2$
16 Else $Req\_out[o] = 1$
17 Else $Req\_out[o] = 0$
18 Write_out($Req\_out[o]$)

Algorithm 2 Operation of the Out width controller

Input : Local pressure $Req\_out\_d[N]$ and neighbor pressure $Req\_in\_d[N]$
Container: Request bank on each output $Req[N]$ (same in Algorithm I)
Output: Master link allocation control $MLA[N]$, slave link allocation control $SLA[N]$, Input switch allocation mode $SA\_in[N][V]$
1 //Initialize the switch allocation input mode
2 For each in port$i$ in N
3 For each VC channel v in V
4 $SA\_in[i][v] = 0$
5 For each output port o in N
6 $MLA[o] = enable$
7 $SLA[o] = disable$
8 Switch $Req\_out\_d[o]$:
9 Case "2":
10 If $Req\_in\_d[dir] = 0$ Then
11 $SLA[dir] = enable$
12 If $Req[dir].Size() = 1$
13 For request (i, j) in $Req[dir]$
14 $SA\_in[i][j] = 1$ // set input SA request mode to 1
15 Case "0":
16 If $Req\_in\_d[dir] = 2$ Then
17 $MLA[dir] = disable$

which means that two separate VCs are requesting a one-flit channel each. Otherwise, the switch arbitration mode of the requesting VC is set to "1" (line 14) which allows two flits within the same packet to participate in the arbitration for both the master and slaver links. This constitutes the proposed flit-level speedup.

B. Router datapath design to support flit-level speed-up

1) Input buffer organization: In flit-level speedup scheme, it is possible that the two incoming flits are for the same VC buffer. Therefore it is required to support reading/writing two flits in the same VC buffer at the same time. One way is to adopt multiple ports memory. However, this causes significant overhead in terms of area and memory access time [3]. Here we propose a novel input buffer organization, which is shown in Figure 4, to satisfy this requirement.

Figure 4-a shows the input port configuration with V virtual channels. Two 1-to-V demux and V flit assemblers are needed

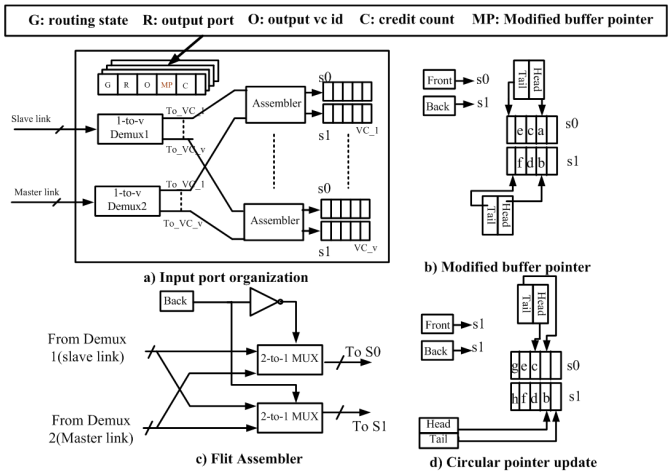


Figure 4: Proposed buffer organization

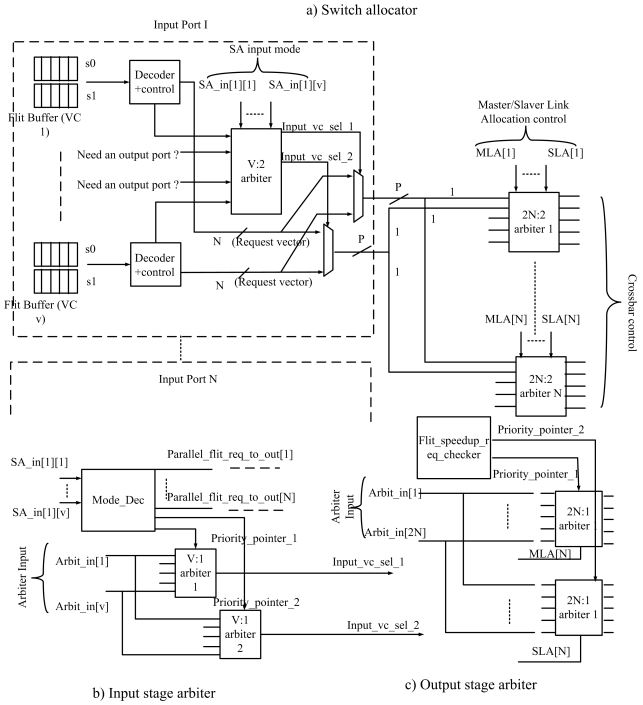


Figure 5: Switch allocator logic

for each port. The original virtual channel buffer is splitted into two sub-buffers, s0 and s1, respectively. Both sub-buffers share the same buffer state information (the GROPC vector in Figure 4-a). The incoming flits are written into s0 and s1 alternatively so that two consecutive flits are stored in different sub-buffers. Figure 4-b shows the modified buffer pointer in each virtual channel. For s0 and s1, each has a head-tail pointer pair pointing to the start and end addresses of the flits of the packets. In addition, two 1-bit registers, denoted as the Front and Back registers, are added to indicate which sub-buffer holds the first and the last flit (e.g. flits "a" and "f" in Figure 4-b) of the VC channel.

The details of the flit assembler is described in Figure 4-c. It is responsible for assembling the flits into the sub-buffer of the VC in the right order. When two flits are sent in the reconfigurable links, the first flit is always connected to the

master link of the sender (i.e. the slave link of the receiver). To ensure the correct assembling of the flits, the slave link is always connected to the sub-buffer that is not pointed by the Back register. Figure 4-b shows an example. The Back register points to s1 since the last flit "f" is stored in s1. The assembler will connect the slave link to s0 and the master link to s1 in the next operation. The read operation is similar to the write operation based on the value of the Front register. Figure 4-d illustrates the updated pointers after reading a flit "a" from the buffer and writing flits "g" and "h" into it.

2) Switch allocator design : To support flit-level speedup in the NoC, the switch allocator should not only support granting two requests from different VCs, but also allow a single VC to arbitrate for both the master and slave links. Figure 5-a shows the building block of our switch allocation module. For each input port, a V-to-2 arbiter is employed in the input stage to select 2 out of many requests from the V virtual channels. The SA input switching arbitration mode vector (SA[N][V]) determines whether the two winning requests are from different VCs (normal arbitration mode) or from the same VC (flit-level speedup mode). At the output side of the switch allocator, N 2N-to-2 arbiters are utilized to decide the actual number of requests that is finally granted.

Figure 5-b shows the detail design of the V-to-2 arbiter for the input allocation. It consists of two V-to-1 arbiters and a mode decision module (Mode_Dec). Mode_Dec reads in the SA input mode switching arbitration vectors (SA_in[N][V]) and determines the selection mode for the current input port. If all the entries of SA vector equal to 0, it means there is no request that requires two flits to be transmitted from the same VC and hence the two v-to-1 arbiters will work independently to select two VCs using different priority pointers. Otherwise, the Mode_Dec will select one VC winner and generate the priority pointer exactly pointing to that VC for both arbiters.

Figure 5-c shows the 2N-to-2 arbiter used in the output side. It consists of two 2N-to-1 arbiters. The link allocation control signals (MLA[N] and SLA[N]) enable/disable the arbiters accordingly. A module named Flit_speedup_req_checker shown in Figure 5-c is used to grant two requests to the same input port so as to allow parallel transmission of two flits from the same VC.

IV. Simulation results

A. Simulation setup

We evaluate the proposed flit-level speedup scheme using a cycle-accurate NoC simulator modified from Noxim [7]. A 8×8 mesh-based NoC architecture is used. We assume each input port of the router has 4 virtual channels and the buffer depth of each VC is 16 flits. The flit size is 64 bits and each packet is made up of 16 flits. Dimension ordered XY routing is implemented in the simulation to guarantee deadlock free routing. We compared the performance of the proposed scheme with three different architectures, namely the unidirectional NoC, unidirectional NoC with 2X input speedup, and the bidirectional NoC. Various traffic patterns were used in the simulation, including synthetic traffic and real benchmarks. Real benchmarks include MWD (Multi-Window Display) [8], MMS (Multimedia system) [9] , MPEG4 (MPEG4 codec) [8] and DVOPD (Dual Video Object Plane Decoder) [10] as well as three applications named auto_indust, telecom and consumer from E3S [11] benchmark. For these application-specific traffic benchmarks, we use the energy-aware mapping [9] to map the task graphs onto the mesh architecture first.

B. Simulation results on synthetic traffics

For synthetic traffic evaluation, five traffic patterns were used: uniform random, transpose, hotspot, shuffle and bit-reversal [7]. Figures 6 a-e summarize the comparison results on the latency. For all the cases, the proposed flit-level speedup scheme out-performs the other three schemes. As observed from the figure, the improvement highly depends on the traffic patterns. For most non-uniform traffics such as the transpose traffic, our proposed scheme can effectively utilize the links and hence improve the performance. Compared with the bi-directional switching scheme, the flit-level speedup further improves the throughput by 5%-30%.

In Figure 6-f, we evaluate the saturation throughput under different packet length (4-16 flits) and buffer depth (8-20 flits per VC) for the uniform traffic. As observed from the figure, the throughput improvement increases when a longer packet length is adopted which is the case in many existing NoC designs [5], [3].

C. Simulation results on real benchmarks

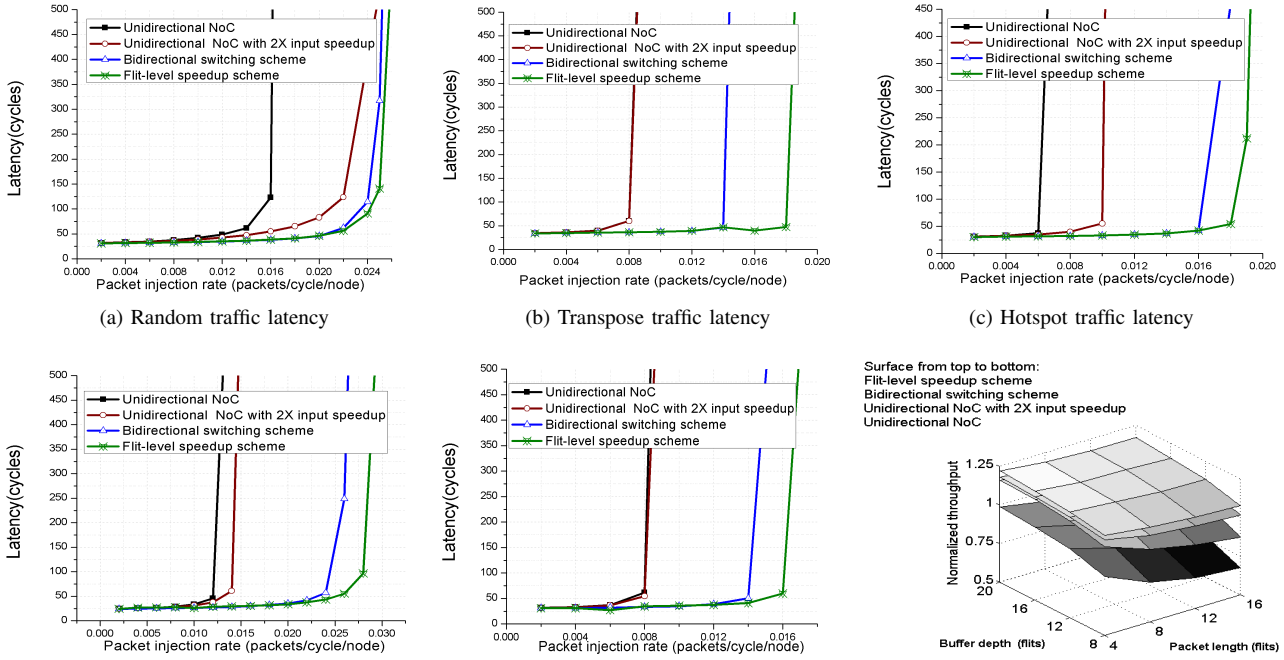
Figure 7 compares the saturation throughput of several real benchmarks. As shown from the figure, while the bi-directional switching scheme out-performs the two unidirectional schemes, our proposed flit-level speedup scheme consistently further improves the performance. The improvement ranges from 2% to 17%. In Figure 8, we show the histogram of the packet delivery time under 80% saturation injection factor of the typical NoC for the telecom application. As shown in the figure, not only the average latency but also the maximum delay are significantly reduced which provides a good prospective to satisfy the QoS requirements.

D. Implementation overhead

Each scheme is modeled in Verilog and synthesized using Synopsys Design Vision based on Nangate 45nm library [12]. From the synthesis results, the area overhead of the input speedup, bidirectional switching and the proposed flit level speedup scheme is 1.87%, 4.41% and 8.08%, respectively. Compared with a typical NoC design, the additional area is mainly due to the larger crossbar size (3.14% overhead) and the modified buffer organization (3.87% overhead) to support flit-level speedup.

V. Conclusions

In this work, we propose a flit-level speedup scheme for improving the NoC performance using self-reconfigurable bi-directional links. In order to support transmitting two flits within the same packet at the same cycle, a novel channel direction control protocol is proposed to dynamically configure the link directions. The corresponding design of the input



(a) Random traffic latency (b) Transpose traffic latency (c) Hotspot traffic latency

(d) Shuffle traffic latency (e) Bit-reversal traffic latency (f) Impacts of buffer depth and packet length

Figure 6: Simulation results for synthetic traffics

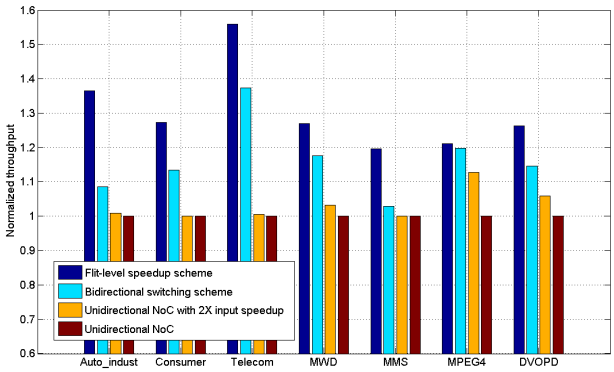


Figure 7: Benchmark throughput comparison

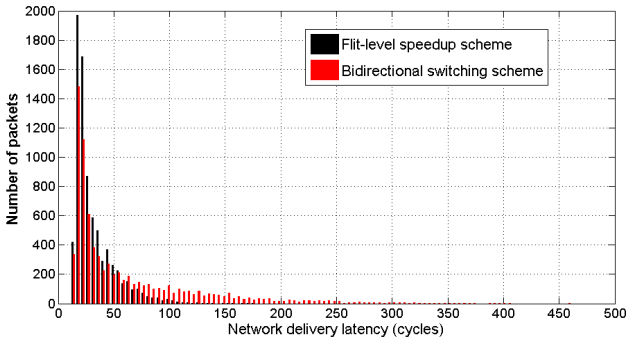


Figure 8: Histogram of the delivery time for Telecom

buffer organization and the switch allocator are also proposed. From the simulation results, significant improvement in latency and throughput are achieved for both synthetic traffic and the real benchmarks.

REFERENCES

[1] L. Benini and G. De Micheli, "Networks on chips: a new soc paradigm," *Computer*, vol. 35, no. 1, pp. 70 –78, jan 2002.

[2] M.A. Al Faruque, T. Ebi, and J. Henkel, "Configurable links for runtime adaptive on-chip communication," in *Design, Automation Test in Europe Conference Exhibition, 2009. DATE '09.*, april 2009, pp. 256 –261.

[3] William Dally and Brian Towles, *Principles and Practices of Interconnection Networks*, Morgan Kaufmann, 2003.

[4] P. Bogdan and R. Marculescu, "Quantum-like effects in network-on-chip buffers behavior," in *Design Automation Conference, 2007. DAC '07. 44th ACM/IEEE*, june 2007, pp. 266 –267.

[5] Ying-Cherng Lan, Hsiao-An Lin, Shih-Hsin Lo, Yu Hen Hu, and Sao-Jie Chen, "A bidirectional noc (binoc) architecture with dynamic self-reconfigurable channel," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 30, no. 3, pp. 427 –440, march 2011.

[6] Myong Hyon Cho, M. Lis, Keun Sup Shim, M. Kinsy, T. Wen, and S. Devadas, "Oblivious routing in on-chip bandwidth-adaptive networks," in *Parallel Architectures and Compilation Techniques, 2009. PACT '09. 18th International Conference on*, sept. 2009, pp. 181 –190.

[7] *Noxim simulator, http://noxim.sourceforge.net*, 2011.

[8] D. Bertozzi, A. Jalabert, Srinivasan Murali, R. Tamhankar, S. Stergiou, L. Benini, and G. De Micheli, "Noc synthesis flow for customized domain specific multiprocessor systems-on-chip," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 16, no. 2, pp. 113 – 129, feb. 2005.

[9] Jingcao Hu and R. Marculescu, "Energy- and performance-aware mapping for regular noc architectures," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 24, no. 4, pp. 551 – 562, april 2005.

[10] A. Pullini, F. Angiolini, P. Meloni, D. Atienza, Srinivasan Murali, L. Raffo, G. De Micheli, and L. Benini, "Noc design and implementation in 65nm technology," in *Networks-on-Chip, 2007. NOCS 2007. First International Symposium on*, may 2007, pp. 273 –282.

[11] *E3S benchmarks, http://ziyang.eecs.umich.edu/dickrp/e3s/.*

[12] *Nangate 45nm library, http://www.nangate.com.*