J Supercomput (2015) 71:3770–3786
DOI 10.1007/s11227-015-1466-0

A low-cost and latency bypass channel-based on-chip network

Amir Fadakar Noghondar[1] · **Midia Reshadi**[1]

Published online: 8 July 2015

Abstract The number of cores on the chip increases rapidly; therefore, scalability is the most important design choice. Mesh-based Networks-on-Chip (NoC) are the most widely used topologies as a scalable alternative for traditional shared bus in many-core chips today. As the NoCs diameter increases, the low-latency communication between cores is becoming more important to ensure sustained scalability, and higher performance. In the ideal network, the low-load network latency between a source and destination is almost equal to *single cycle*. In this work, we propose a router for network-on-chip called Bypass router that leads to create a single-cycle data path all the way from the source to the destination. We do not use any additional control links in the network; instead the proposed router is compatible with all topologies and deterministic routing algorithm. We also propose a new routing algorithm to use the advantages of our router design. The area consumption is also reduced on $4 \times 4, 8 \times 8, 16 \times 16$ mesh topologies, compared to SMART network (Krishna et al. IEEE 19th international symposium on high performance computer architecture (HPCA2013), 2013). System simulations with Noxim simulators demonstrate at mean 60 % reduction in latencies across synthetic traffic patterns compared to a baseline router.

Keywords Multi-processor System-on-Chip (MPSoC) · Network-on-Chip (NOC) · Interconnection network

✉ Amir Fadakar Noghondar
a.fadakar@srbiau.ac.ir

Midia Reshadi
reshadi@srbiau.ac.ir

[1] Department of Computer Engineering, Science and Research Branch, Islamic Azad university, Tehran, Iran

1 Introduction

The number of cores on chips is increasingly growing with the advancement of chip-manufacturing technology. Therefore, networks on chip have turned into the main communication paradigm of many-core chips due to their scalability and performance. While the dimensions of networks-on-chip are increased, researchers are looking for improvement of parameters such as latency, power and area consumption in networks-on-chip. Generally, networks-on-chip parameters could be divided into the performance and cost. Performance parameters include throughput and average packet latency while cost parameters include power consumption, area overhead and peak temperature [2].

A variety of techniques to improve networks-on-chip parameters are introduced such as applications mapping techniques [3,4], topology level solutions [5–9], routing manners [10–13], router-design techniques [1,14–16] and flow control methods [17]. All of these techniques try to improve cost and performance parameters in the network.

This paper proposes a scheme for reducing the communication latency which is one of the most important topics in networks-on-chip. A packet should be traversed through intermediate routers and connecting links on the network. Thus, the routers and communication links are the most important latency parameters in networks on chip. Latency of a packet is calculated based on clock cycle using the following equation [1]:

$$T = H \,.\, t_r + H \,.\, t_w + T_c + L/b \tag{1}$$

In which H is the number of hops, t_r is the delay of the router, t_w is delay of communication wires between two routers, T_c is delay of congestion in the router and L/b is the serialization delay for the body and tail flits, i.e. time for a packet of length L to cross a channel with bandwidth b equals L/b [1].

Packet latency is increased for one cycle with intermediate node latching. Flit latching in each router leads to increase H in Eq. 1. When dimension of network is extended, consequently H count increases and packet latency grows. A packet should not be required to latch at the intermediate nodes and if intermediate node latching can be avoided, a significant fraction of the packet latency overhead can be removed. By ignoring wire delay of communication links, flits can traverse the path between source and destination through the network without latching according to Fig. 1. In this case, the ideal network latency equals $T = 1 + L/b$ [1].

There are two kinds of path: 1—one-dimensional paths which are along a dimension and without turning. 2—All other paths are referred to as two-dimensional paths. It should be possible that flits could be passed through the intermediate router without latching in both paths. In this work, we propose the bypass router to get to the single-cycle network. Flits can traverse one or two dimensional path without latching through the bypass router. The proposed router is appropriate with different topologies and deterministic algorithm. We also propose a novel routing algorithm to complement our two-dimensional bypass router design.

The rest of the paper is organized as follows. Sect. 2 presents related works, description of work is discussed in Sect. 3. Bypass router is introduced in Sect. 4, one-dimensional single-cycle paths is described in Sect. 5. Section 6 demonstrates

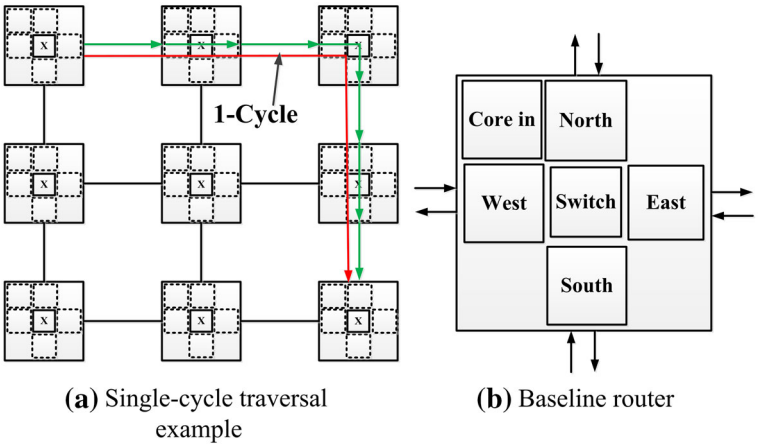


Fig. 1 Single-cycle traversal

two-dimensional single-cycle paths and the routing algorithm is described in Sect. 7. Section 8 introduces the results of simulation and evaluation. Conclusion is presented in Sect. 9.

2 Related work

The best we can do right now is to design *single-cycle* routers, such that a packet travels straight through the router at the intermediate nodes without latching. Therefore, bypass channels has been proposed to cross flits through the network in a single cycle. Two different approaches to use bypass channels are introduced in [1,14].

In [14], an on-chip network has been introduced that lets the flits traverse some of the determined path in the absence of network congestion via bypass channels. The proposed design in [14] is only compatible to its own topologies and it is not suitable for topologies such as Mesh, torus, etc. XY routing algorithm is implemented using source routing (which is used additionally in this approach) is not scalable [18,19]. As the number of bypass channels in the network is limited, it is not possible for all flits to traverse the path via the bypass channels. Restriction in the number of bypass channels leads to competition for obtaining the path [14].

In [1], a network-on-chip named SMART is introduced which lets the flits to traverse a number of hops in a clock cycle between the source and destination. Improvement of latency parameter is the advantage of SMART which forces hardware overhead to the system. When the network diameter is expanded, the number of *SSR* wires increases. Flits require to asynchronous links to traverse in a clock cycle. In each router, additional computation for arbitration among the different values of *SSR* wires is done additional to the flits routing.

The designed bypass router in this paper improves the latency parameter like SMART router while there is no need for *SSR* wires in contrast to SMART. Furthermore, the designed router in this article is adaptable with all topologies and

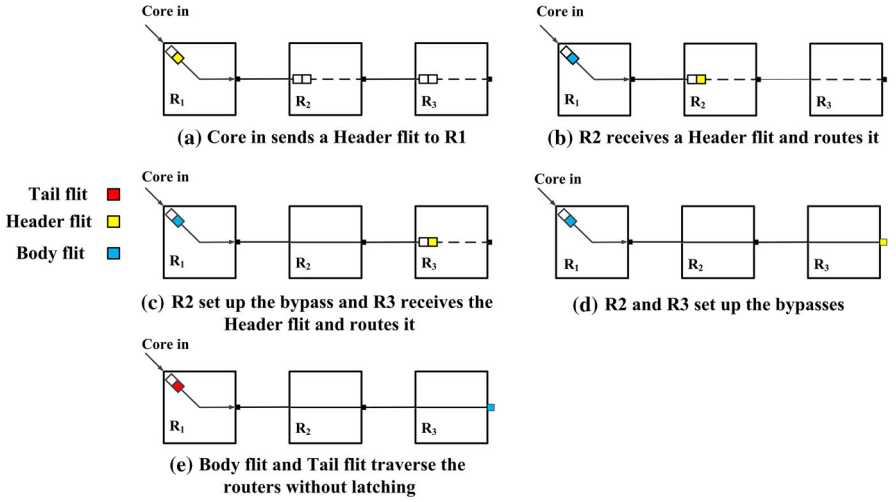


Fig. 2 Set up the bypass path

deterministic or proposed routing algorithms which shows the flexibility of the presented approach.

3 Description of work

SMART needs SSR wires to set up the bypass path that makes overhead for system but our work does not need any control link. The proposed work in this article uses the header flit instead of using control links to create the bypass path. The header flit sets the bypass path in the intermediate routers and then the body flits and tail flit traverse the intermediate routers without latching. Figure 2 shows an example, in Fig. 2a Core_in sends a header flit to Router1 and Router1 routes and sends it to Router2, in Fig. 2b Router2 routes the incoming header flit and sets the bypass path, in Fig. 2c Router3 routes header flit and sets the bypass path as well as Router2 then the body flit traverses through Router1 and Router2 without latching in Fig. 2d. The microarchitecture of bypass router will be described in the next Section.

4 Description of the router

Networks-on-Chip (NoC) consist of routers and shared links that connects all IP cores and each hop consists of a router and a link. According to Fig. 3, the router includes five ports (East, West, North, South, and IP Core) similar to a router in typical two-dimensional mesh. The received flit in each input port should be forwarded to one of the output ports and also the goal of a router is forwarding flits to the links. A router performs the following actions in a pipelined manner [20]:

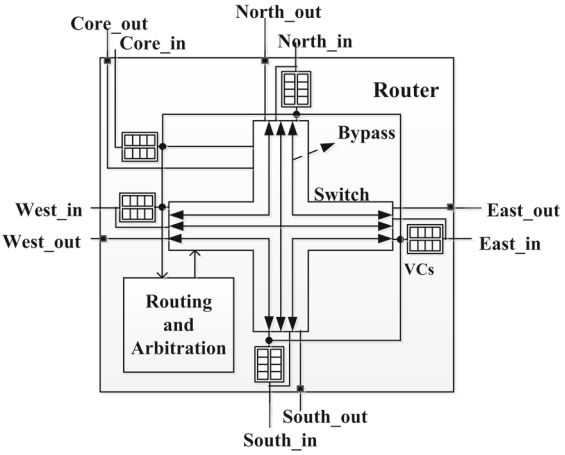


Fig. 3 Bypass router

Stage 1:

Buffer Write (BW): the incoming flit is buffered.

Route Compute (RC): the incoming header flit chooses an output port to depart from.

Stage 2:

Switch Allocation (SA): buffered flits arbitrate among themselves for the input and output ports of the crossbar switch. At the end of this stage, there is at most one winner for every input and output port of the crossbar.

VC Selection (VS): header flits that win SA reserve a VC for the next router, from a pool of free VCs [1].

Stage 3:

Switch Traversal (ST): SA winners use the Xbar.

Link Traversal (LT): each flit incurs 3 or more cycles at each router, followed by one cycle in the link connecting the crossbar to the next router.

In case of contention, Link Traversal (LA) can fail; in this case the flit is buffered and goes through the regular three-stages pipeline. During Switch Allocation (SA), all inter routers arbitrate among the flits that they received; the arbiters guarantee that only one flit will be allowed to access to a particular output port of the crossbar [1].

The proposed bypass router has above pipeline manner and also bypass action that in some cases send flits in less than three-stages. Flits should be latched in the first-stage pipeline but the bypass router allows flits to pass through the router without latching.

Figure 3 shows all possible bypass paths in the bypass router. As shown in Fig. 4a, each bypass router contains four sub-switch blocks corresponding to each direction. Each sub-switch block can be filled with the bypass block in Fig. 4b or the simple block in Fig. 4c. For example, the bypass router in Fig. 5a corresponds to the router block diagram in Fig. 5b.

Three simple types of bypass path are shown in Fig. 6:

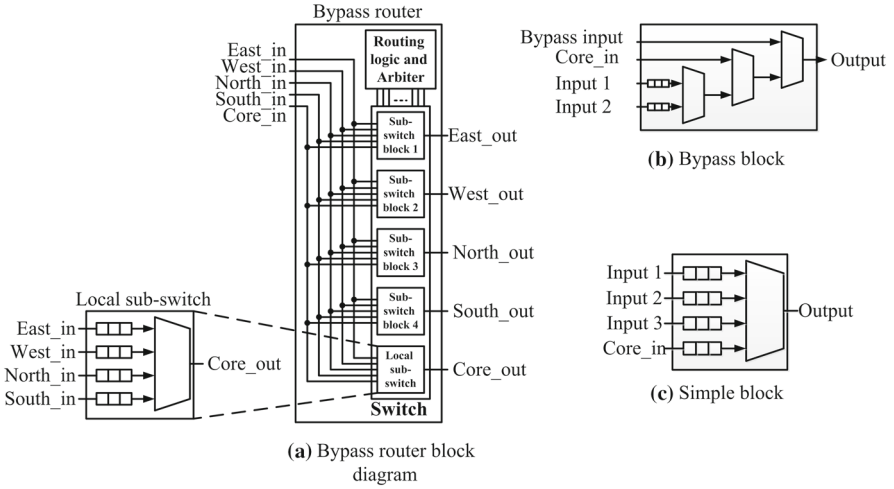


Fig. 4 Bypass router micro architecture block diagram

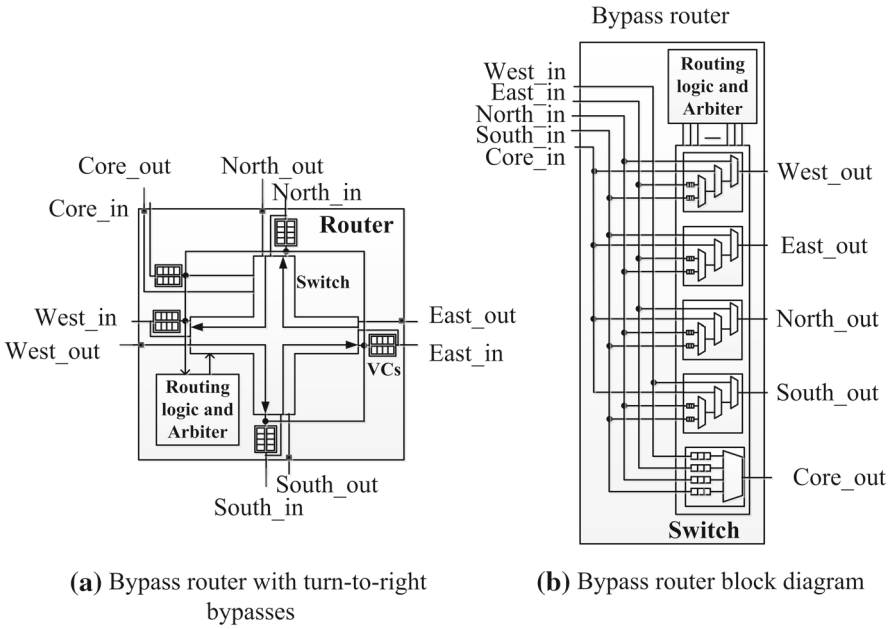


Fig. 5 Example of bypass router with turning bypasses

1—Straight bypass: joining a given input port to the output port in the same dimension.
2—Turning-to-right bypass: refers to the bypass which turns right for connecting input port to output port in the router.
3—Turning-to-left bypass: refers to connect input port to output port that makes left turn.

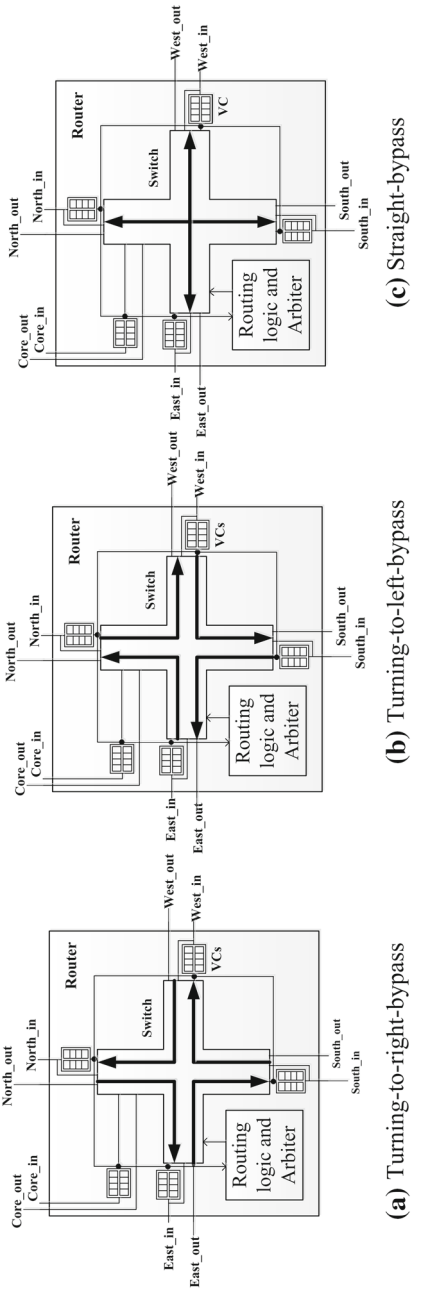


Fig. 6 Example of permitted bypass

In our router, the arbiter operates so that the flits served requestor for bypass path has the highest priority than other flits. First, a bypass path is determined and reserved for the other flits of a packet. The header flit of the packet makes its way from the sender to receiver reserving bypass paths along the way.

The flit that could be forwarded through the bypass path is called bypass flit. The bypass flit has the highest priority to win the output in Switch Allocation (SA) stage. In the case of bypassing, only the header flit incurs router pipeline and body and tail flits traverse routers without latching.

One-dimensional bypass routers only have a straight bypass and routers along each dimension can be bypassed, so flits need to stop at turn router. The straight and turning-to-right or left bypass paths can be mixed together for making two-dimensional bypass routers which one turn can be bypassed. The goal of proposing bypass router is reducing or removing latching at the intermediate nodes. All communications are carried out in the form of packets, subdivided into flits, which are sent through the NoC using wormhole flow control.

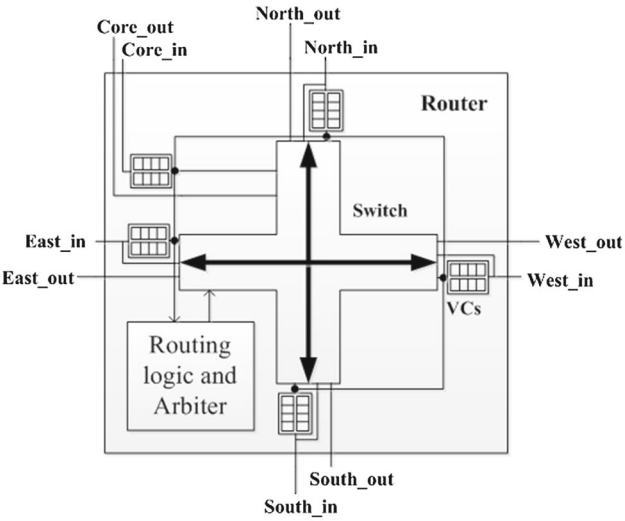
5 One-dimensional single-cycle paths

The SMART network uses *SSR* wires, which causes hardware overhead to create SMART path. The proposed router of this paper needs no control wires for making the bypass path. We start with a design where we do not allow bypass at turns. As seen in Fig. 7a, the bypass path in the router lets flits to traverse without latching. In the other words, if a flit travels through the straight, it is desirable that the flit be forwarded without latching. Based on Fig. 7b, flits can virtually fly through straight paths without getting latched using proposed bypass router.

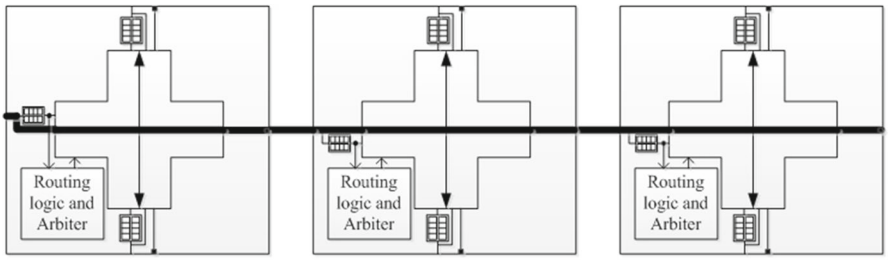
Firstly, the bypass router routes the header flit. Providing a flit could be departed from the router through the bypass path, a flit is allowed to access to the particular output even if it was obtained by other input port. The bypass input port has the highest priority to access to the output port and other bypass flits including body and tail flits are forwarded to the output port without latching. When the output port is obtained by the bypass input port, it is unreachable until passing the tail flit.

If a flit cannot be forwarded through the bypass path, then it will be buffered and reserves the output port. In this case, buffered flits start to traverse after releasing the output port. In the case of bypassing, it should be noted that the header flit only routed in the intermediate routers and other bypass flits including body and tail flits traverse the routers without latching and tail flit frees the output port.

Algorithm 1 shows the pseudocode of bypassing function and Fig. 8 is an example about functionality of a bypass router. In Fig. 8a, South and North input ports send flits to West port. In Fig. 8b, c, West input port requires East output port and because of connecting West port to the East through a bypass path, West input port obtains East output port and incoming flits to North input port will be buffered. In Fig. 8d, e, forwarding flits from South to East is ended and North could resend its flit to East port. In Fig. 8f, sending West flits is ended and East port is released.



(a) Straight-bypass in the Bypass router



(b) Example of one-dimensional single-cycle traversal

Fig. 7 One-dimensional single-cycle path

6 Two-dimensional single-cycle paths

Two-dimensional bypass routers can be made up through a combination of simple bypass paths and two-dimensional paths can be traversed without latching. For example, in Fig. 9a two-dimensional bypass router consists of straight and turning-to-right bypass paths. Flits are not latched on the turning path and they can traverse both X and Y dimensions based on Fig. 9b.

If incoming header flit of an input port could be passed through a bypass path toward an output port, then input port obtains the output port and other flits including body and tail flits are propagated along a bypass path toward the output port. When input port starts sending flits to the output port through the bypass path, the output port is unreachable since passing tail flit. It should be noted that the header flit obtains the output port and body and tail flits traverse the router without latching.

```
Input: Incoming flit Output: Bypass flit
for each direction of router do
    if incoming flit is header flit then
        Route incoming flit;
        if incoming flit is bypass flit then
            Send flit to the output;
            The output port is unreachable;
        end
        else
            Buffer the incoming flit;
        end
    end
    if incoming flit is body flit then
        if incoming flit is bypass flit then
            Send flit;
        end
        else
            Route and buffer the flit;
        end
    end
    if incoming flit is tail flit then
        if incoming flit is bypass flit then
            Send flit and release the output port;
        end
        else
            Route and buffer the flit;
        end
    end
end
```

Algorithm 1: Bypass algorithm

7 Routing algorithm

We propose a new semi-adaptive routing algorithm for using the advantage of lower latency in two-dimensional path. The proposed bypass router can be used in both deterministic routing algorithm such as XY routing and the semi-adaptive routing algorithm.

We developed a modified version of standard adaptive routing algorithm that is deadlock free. The proposed algorithm works as follows: in the first, router routes the header flit with adaptive routing and if the output is accessed through the bypasses then the output port is assigned to the input and the other incoming flits will be propagated to the output and the output port is unreachable since the tail flit does not traverse. If header flit could not be passed through the bypasses in router, then body and tail flits are routed according to adaptive routing algorithm. Algorithm 2 shows the function of proposed routing algorithm.

The proposed routing algorithm behaves like the adaptive routing except that flits are propagated along the bypass path and they are forwarded through the router without latching. In this case flits are not routed by routing algorithm mechanism.

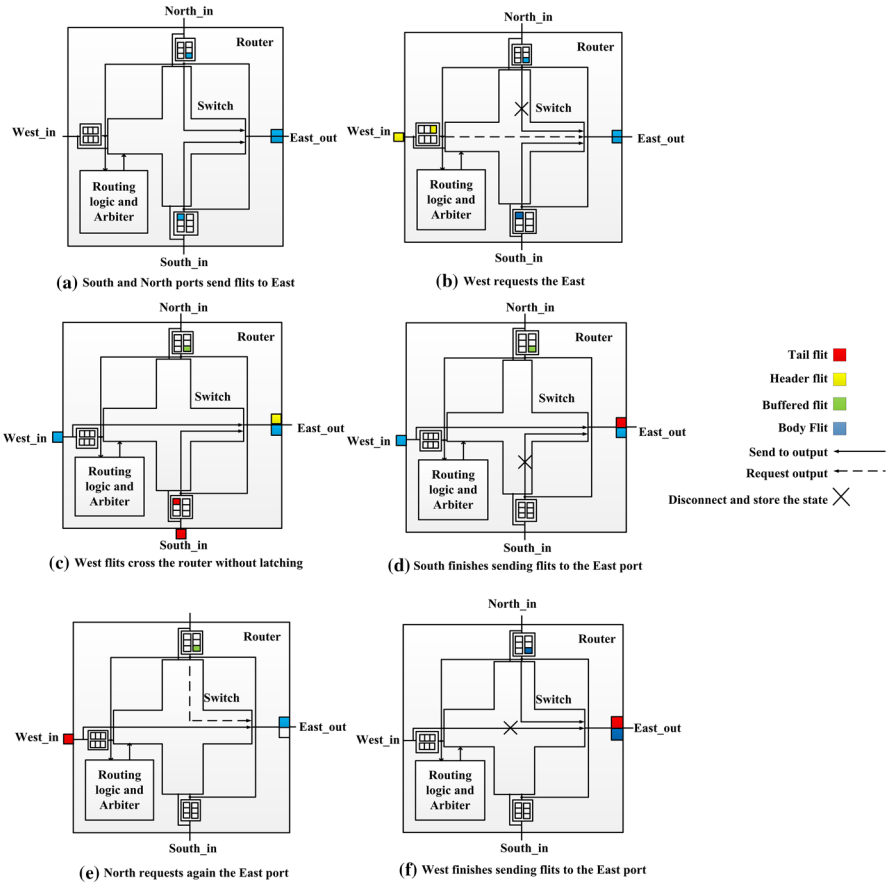
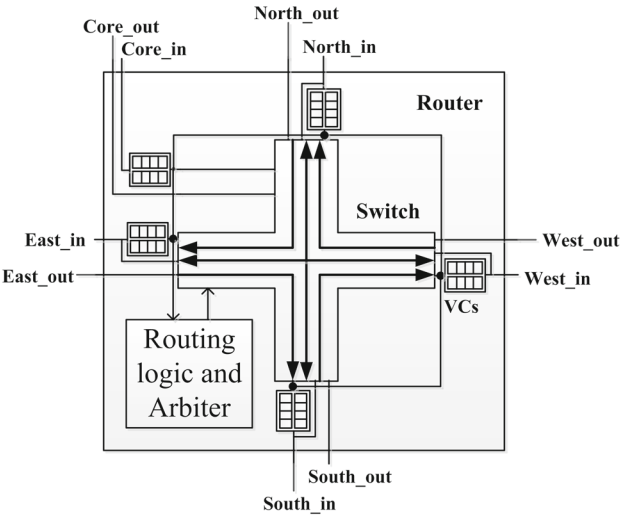


Fig. 8 Example for the function of bypass router

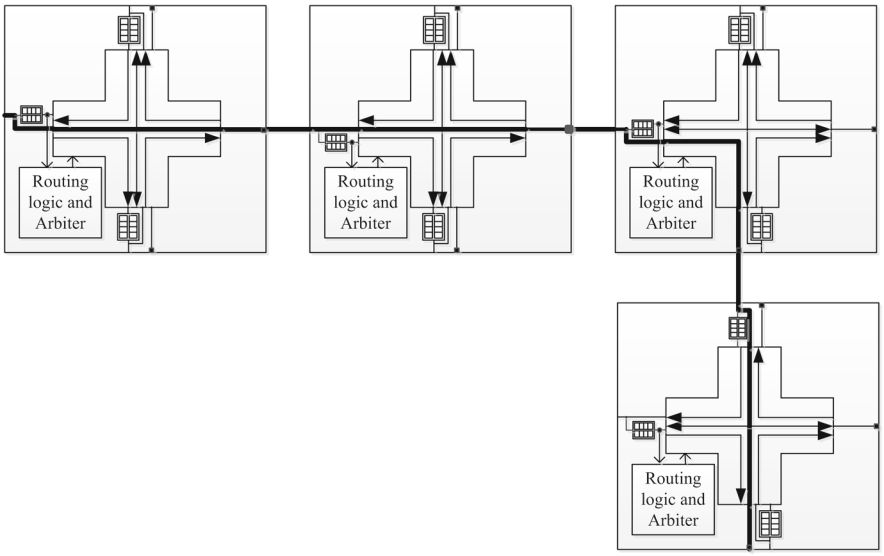
8 Evaluation

The proposed router in this article is simulated by Noxim [21] simulator. We extend the Noxim for supporting the proposed router. The configuration parameters are shown in Table 1. We start running one-dimensional bypass router in 4×4, 8×8, 16×16 mesh topologies with different traffic patterns and XY routing algorithm. The latency of one-dimensional bypass router is compared with baseline router in Fig. 10. Proposed router simulations demonstrate reduction in average packet latency. This is because flits are not latched in each router and it decreases number of hop count (H) in Eq. 1.

One-dimensional bypass router is simulated in 4×4, 8×8, 16×16 mesh with proposed routing algorithm and Butterfly traffic distribution. The results of simulation are compared with baseline router in the same topology and adaptive routing algorithm in Fig. 11. As shown, latency is reduced in one-dimensional bypass router with deterministic routing and proposed routing algorithms. The bypass paths lead to flit traverse the intermediate routers without latching and it leads to reduce number of



(a) Combination of Straight-bypass and Turning-to-right-bypass in the Bypass router



(b) Example of two-dimensional single-cycle traversal

Fig. 9 Two-dimensional single-cycle path

hops in the traversing between source and destination and also reducing in average packet latency.

The two-dimensional bypass router in Fig. 12a is simulated in 8×8 mesh with semi-adaptive routing algorithm and latency improvement is shown in Fig. 12b. In this case, flits can bypass both routers along a dimension and turn router(s). The

```
Input: Incoming flit Output: the output port
Routing (incoming flit)
if incoming flit is bypass flit then
    Forward it without latching;
end
if incoming flit is header flit then
    Route according adaptive routing algorithm;
    if the output is reached through the bypass then
        Free the output and make it available to the bypass flit;
    end
    else
        Reserved the output;
    end
end
else
    Rout incoming flit according adaptive routing algorithm;
end
if incoming flit is tail flit then
    Free the output;
end
```

Algorithm 2: Semi-Adaptive Routing Algorithm

Table 1 Target system and configuration

Technology	45 nm	Topology	4×4, 8×8, 16×16 Mesh
Vdd	1.0	Router ports	5
Frequency	1.0 GHz	Routing	*XY*, proposed routing
Link length	1 mm	Flit width	128-bit

simulation result of two-dimensional bypass router is compared with baseline router which is simulated in 8×8 mesh with adaptive routing algorithm and butterfly traffic. In baseline router, flits are latched in each router and it increases the latency; but in bypass router, flits can be forwarded through the bypass path and it decreases hop count (H) and latency.

The energy consumption of SMART is compared with bypass router and baseline router for 8×8 mesh topology in Fig. 13a. The energy consumption for the SMART network is more than baseline and proposed network and this is because the SMART network uses *SSR* wires, which force hardware overhead to the network, to create SMART path. Our proposed network has less energy consumption than baseline and it is because flits bypass intermediate nodes and it leads to reduce energy consumption; we illustrate this reason with an example.

For instance in SMART, if a flit wins $SA - L$ and $SA - G$ and traverses a SMART-hop of length 4 in an $\text{HPC}_{\text{max}} = 8$ design, the consumed energy will be $E_{\text{SA-L}} + 8.E_{\text{SSR}} + 4.E_{\text{SA-G}} + E_{\text{buf-rd}} + 4.E_{X\text{bar}} + 4.E_{\text{Link}} + E_{\text{buf-wr}}$ [1]. But in proposed network, the consumed energy for a body or tail flit traverses 4 hop in a single-cycle will be $4.E_{\text{SA}} + E_{\text{buf}} + 4.E_{X\text{bar}} + 4.E_{\text{Link}}$ and consumed energy for a header flit will be $4.E_{\text{SA}} + 4.E_{\text{buf}} + 4.E_{X\text{bar}} + 4.E_{\text{Link}}$. The consumed energy in baseline network for a flit traverse 4 hop will be $4.E_{\text{SA}} + 4.E_{\text{buf}} + 4.E_{X\text{bar}} + 4.E_{\text{Link}}$.

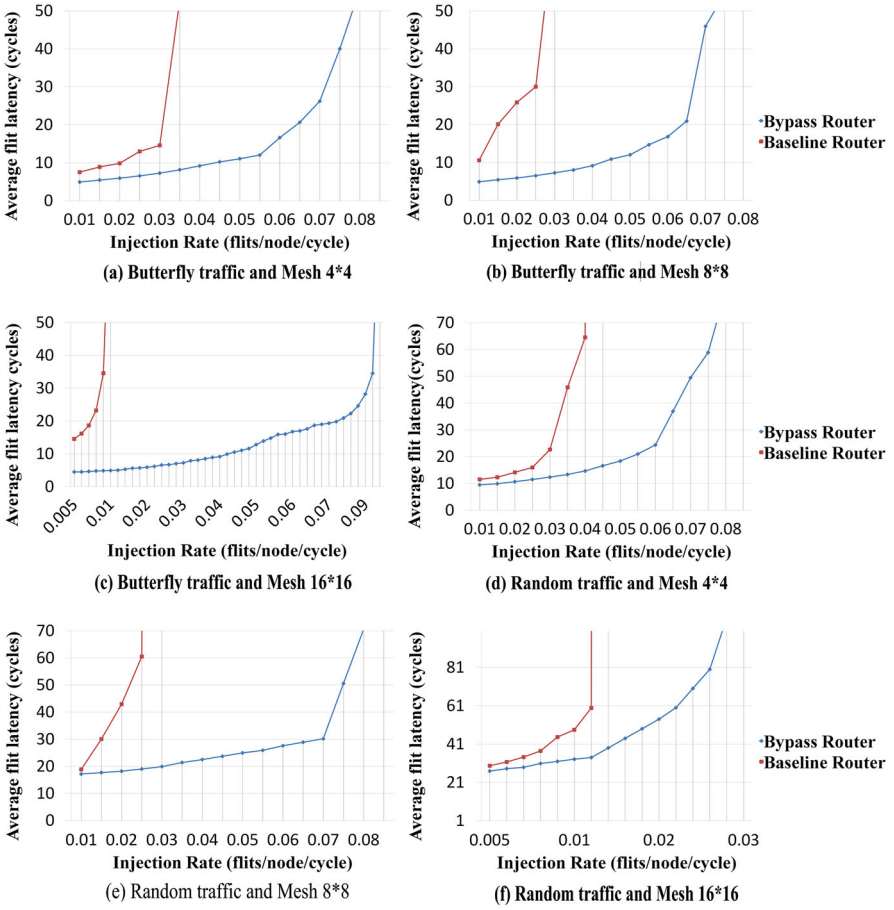


Fig. 10 Bypass router with different synthetic traffic and *XY* routing algorithm

We use Orion 2.0 [22] for area consumption evaluation. For 4×4, 8×8, 16×16 mesh, the area consumption of one-dimensional bypass router is compared with bypassing routers along dimension method in SMART-2D in Fig. 13b. The area consumption is reduced as compared with SMART and this is because we do not use the control link, i.e. *SSR* wires.

The *SSR* wires make hardware overhead for network and increase area consumption, especially when the network diameter is expanded and also number of *SSR* wires increases. For example, we just show that *SSR*s have a significant impact in area consumption.

SMART-2Ds control path is able to achieve an $\mathrm{HPC}_{\max}$ of 9. The total number of SSR-bits entering an input port are $\mathrm{O}(\mathrm{HPC}_{\max}.\log_2(\mathrm{HPC}_{\max}))$ and $O(\mathrm{HPC}^2_{\max}.\log^2(\mathrm{HPC}_{\max}))$ in SMART-1D and SMART-2D, respectively [1]. The number of *SSR* wires in SMART-2D in 8×8 mesh is 1256 and width of each *SSR* is 4 bit ($\mathrm{HPC}_{\max} = 9$) and length of each *SSR* is 9 hops that leads to increase the number of repeater. So we can see a huge increasing amount of area consumption.

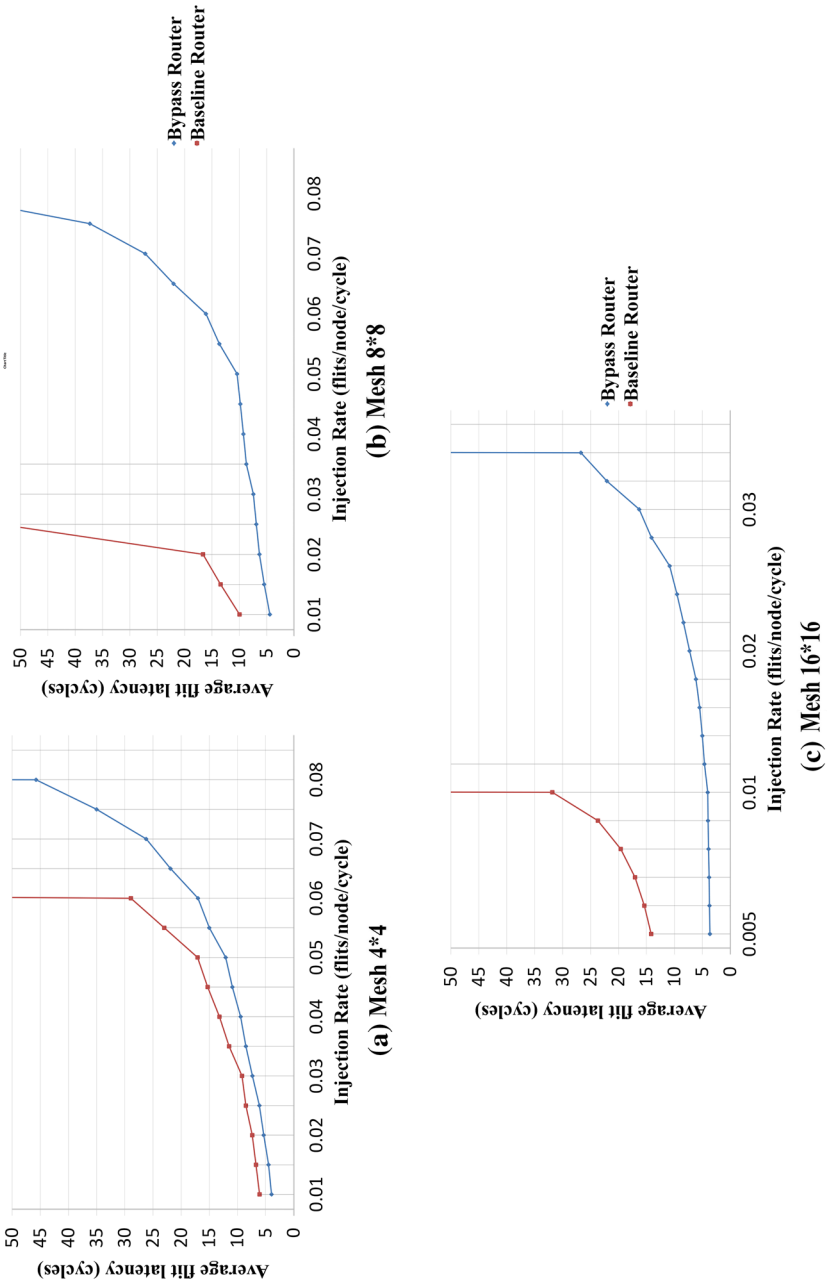
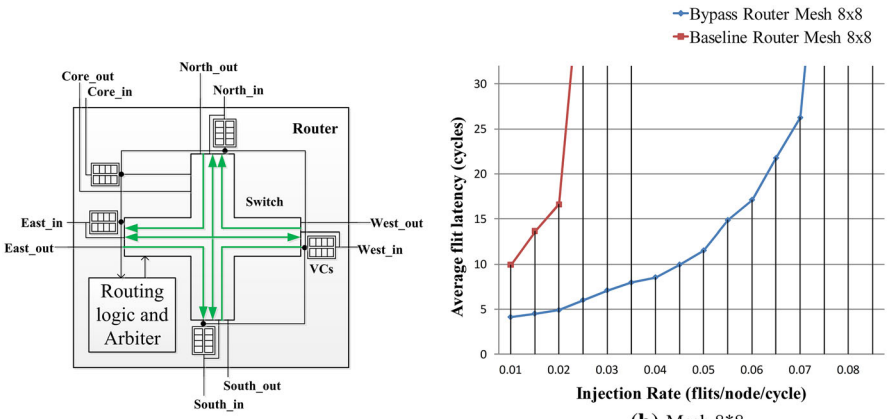


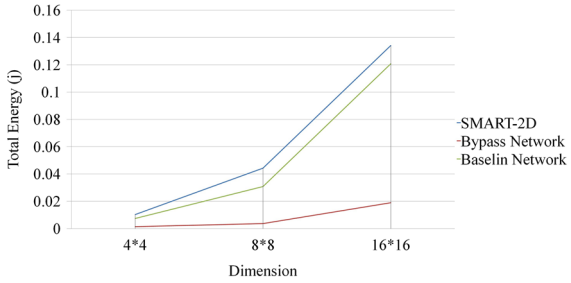
Fig. 11 Bypass router with Butterfly traffic and proposed routing algorithm



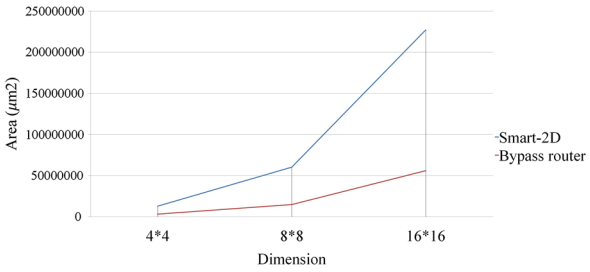
(a) Turning-to-right-bypass **(b)** Mesh 8*8

Fig. 12 Two-dimensional bypass router with Butterfly traffic and semi-adaptive routing algorithm

Fig. 13 Energy and area consumption of bypass router



(a) Energy consumption of Bypass router



(b) Area consumption of Bypass router

9 Conclusion

When dimension of network is expanded, packet latency is increased. For reducing packet latency through the network, it is not required to latch flits in each intermediate router. Therefore, the proposed bypass router in this article is providing a path for traversing flits without latching at the routers. One proposed approach to reduce this latency is SMART [1], which presents a solution to traverse multi-hop paths within a single-cycle by adding physical control links on the network. The proposed bypass

router in this article improves the latency parameter without adding any physical control link on the network. We also propose a new routing algorithm to use the advantage of lower latency in two-dimensional path over the turning-bypass in the proposed router. According to the simulation results in Sect. 8, configuration of single-cycle paths leads to significant reduction of latency while power and area consumption is reduced compared with models presented in earlier works.

References

1. Krishna T et al (2013) Breaking the on-chip latency barrier using SMART. In: IEEE 19th international symposium on high performance computer architecture (HPCA2013)
2. Marculescu R et al (2009) Outstanding research problems in NoC design: system, microarchitecture, and circuit perspectives. Comput-Aided Des Integr Circ Syst IEEE Trans 28(1):3–21
3. Murali S, De Micheli G (2004) Bandwidth-constrained mapping of cores onto NoC architectures. In: Proceedings of the conference on Design, automation and test in Europe, vol 2. IEEE Computer Society
4. Hu J, Marculescu R (2003) Energy-aware mapping for tile-based NoC architectures under performance constraints. In: Proceedings of the 2003 Asia and South Pacific Design Automation Conference, ACM
5. Grot B et al (2009) Express cube topologies for on-chip interconnects. High Performance Computer Architecture, 2009. HPCA 2009. IEEE 15th international symposium on IEEE
6. Kim J, Dally WJ, Abts D (2007) Flattened butterfly: a cost-efficient topology for high-radix networks. ACM SIGARCH Comput Archit News 35(2):126–137
7. Kao Y-H et al (2011) CNoC: high-radix clos network-on-chip. Comput-Aided Des Integr Circ Syst IEEE Trans 30(12):1897–1910
8. Hoskote Y et al (2007) A 5-GHz mesh interconnect for a teraflops processor. IEEE Micro 27(5):51–61
9. Howard J et al (2010) A 48-core IA-32 message-passing processor with DVFS in 45nm CMOS. Solid-state circuits conference digest of technical papers (ISSCC), 2010 IEEE international, IEEE
10. Glass CJ, Ni LM (1992) The turn model for adaptive routing. ACM SIGARCH Comput Archit News 20(2):278–287
11. Chiu G-M (2000) The odd–even turn model for adaptive routing. Parallel Distrib Syst IEEE Trans 11(7):729–738
12. Boura YM, Das CR (1994) Efficient fully adaptive wormhole routing in n-dimensional meshes. Distributed computing systems, 1994. In: Proceedings of the 14th international conference on IEEE
13. Li M, Zeng Q-A, Jone W-B (2006) DyXY: a proximity congestion-aware deadlock-free dynamic routing method for network on chip. In: Proceedings of the 43rd annual design automation conference, ACM
14. Jain TNK et al (2010) Asynchronous bypass channels: improving performance for multi-synchronous NoCs. In: 2010 Fourth ACM/IEEE international symposium on Networks-on-Chip (NOCS). IEEE
15. Rodrigo S et al (2009) Efficient implementation of distributed routing algorithms for NoCs. IET Comput Digit Tech 3(5):460–475
16. Matsutani H et al (2009) Prediction router: yet another low latency on-chip router architecture. In: IEEE 15th international symposium on high performance computer architecture. HPCA 2009. IEEE, pp 367–378
17. Kumar A, Peh L-S, Jha NK (2008) Token flow control. In: Proceedings of the 41st annual IEEE/ACM international symposium on Microarchitecture. IEEE Computer Society
18. Jerger NE, Peh L-S (2009) On-chip networks. Morgan and cLaypool, Cambridge
19. Badri S, Holsmark R, Kumar S (2012) Junction based routing: a scalable technique to support source routing in large NoC platforms. In: Proceedings of the 5th international workshop on Network on chip architectures, ACM
20. Dally WJ, Towles BP (2004) Principles and practices of interconnection networks. Elsevier, Amsterdam
21. Fazzino F, Palesi M, Patti D (2008) Noxim: network-on-chip simulator. http://sourceforge.net/projects/noxim
22. Kahng AB et al (2009) ORION 2.0: a fast and accurate NoC power and area model for early-stage design space exploration. In: Proceedings of the conference on Design, automation and test in Europe. European Design and Automation Association