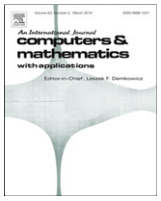
Computers and Mathematics with Applications 64 (2012) 1301–1310

Contents lists available at SciVerse ScienceDirect

Computers and Mathematics with Applications

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



Flexible router architecture for network-on-chip

Mostafa S. Sayed[a,*], Ahmed Shalaby[a], Mohamed El-Sayed[a], Victor Goulart[a,b]

[a] *ECE Department, Egypt–Japan University of Science and Technology (E-JUST), Alexandria, Egypt*

[b] *Center for Japan–Egypt Cooperation in Science and Technology, Kyushu University, Fukuoka, Japan*

ARTICLE INFO

Keywords:
Network-on-Chip
Virtual channels (VCs)
Traffic patterns

ABSTRACT

The growing complexity of systems-on-chip (SoCs) pushes researchers to propose replacing the bus architecture by Networks-on-Chip (NoCs). The key advantages of NoCs are efficient exploitation of performance and scalability. Nowadays NoCs are a well established research topic and several implementations have been proposed. Some techniques are proposed to improve NoC performance in terms of latency and throughput while others are proposed to improve area utilization and power consumption. An important research in NoC design is the tradeoff between area/power and performance. In order to improve performance some techniques tend to increase the number of buffers. However this method increases area and power consumption. This paper introduces new router architecture, called the Flexible Router, which improves the performance of the overall network using the same amount of available buffers but in more efficient way. Therefore there is no need to increase the size of buffers or to use extra virtual channels (VCs) which cause high power consumption, area overheads, and complex logic. The Flexible Router provides a way to handle the requests to a busy buffer by other buffers in the router. It is observed that the Flexible router can achieve better performance in terms of increasing the saturation rate for Hotspot, Uniform, and Nearest-Neighbor traffic patterns, especially Hotspot with an 11.4% increase. Discussion about area overhead compared to the Base router and analysis of arriving out of order packets (side-effect) are also provided.

1. Introduction

Although NoC performance generally increases with the number of buffers used per input port (VCs) and their sizes (FIFO depth) as well, there are two main problems resulting from router buffers in NoCs: area [1,2] and power consumption overheads [3–6].

In [1,2] it is observed that the area of the router increases linearly with buffer size; for example in [2], when the FIFO depth increases from 2 to 3 flits, the area of the router increases by 30%. Concerning power consumption, FIFO buffers alone can consume about one fifth of the total power consumed within the router [3]. According to these problems, it is important to reduce the numbers and sizes of buffers.

Such a tradeoff between performance and area has been studied in [2,7] where they tried to get higher performance by increasing the amount of buffering resources just at the bottlenecks, i.e. input ports with higher congestion. Both papers present static algorithms that identify the bottlenecks and then increase the buffer size [2], or the number of VCs [7] at these bottlenecks. Although they got better performance with huge optimization in NoC buffers (sizes or numbers), a problem still exists, where both of them use static algorithms which give optimization for specific traffic pattern or application. So if the

* Corresponding author. Tel.: +20 1003163829; fax: +20 3 459 9520.
E-mail addresses: mossaied2@gmail.com, mostafa.saied@ejust.edu.eg (M.S. Sayed), ahmed.shalaby@ejust.edu.eg (A. Shalaby), m.ragab@ejust.edu.eg (M. El-Sayed), victor.goulart@acm.org (V. Goulart).

0898-1221/$ – see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.camwa.2012.03.074

application changes, the locations of the bottlenecks could change as well and we will need to repeat the whole operation again.

In [8], a low-cost (LC) router is proposed in which the complexity of the router is decreased by decoupling the routing in the *X* direction from the routing in the *Y* direction for deterministic *XY* routing. This decoupling results in large optimization in the size of the switch crossbar and the number of buffers used and hence both area and power overheads are decreased. However, such architecture is only optimized for the dimension ordered *XY* routing algorithm used in mesh topology therefore it must be modified to run adaptive routing algorithms or even to work under other topologies. Finally, according to some experiments done under some traffic conditions, the proposed router was a bottleneck and the performance was degraded compared to the conventional router architecture.

Véstias and Neto [9] analyze the buffer resizing and propose a dynamic buffer resizing technique for FPGA. This technique uses a floating buffer which can be assigned to any output port buffer to increase its buffering size. The work aims to reduce the fixed buffer size to a minimum and then adaptively compensate the shortage of buffering by assigning the floating buffer to the output port. However this technique targets the FPGA platform only and also is less efficient with small buffers implemented as it uses higher area. It depends on the FPGA architecture which FIFO areas are practically the same for a set of sizes; not like ASIC in which the area of the FIFOs is proportional to the size. Hence there are limitations on buffer size and platform.

A new router architecture called Virtual Channel Regulator (ViChaR) was proposed in [10]. ViChaR uses the concept of unified buffer structure (UBS) [11], which enables each input port to dynamically allocate the available VCs according to the incoming traffic, so it appears to have a variable number of the VCs. Therefore as the traffic increases the router will have more VCs with less depth and vice versa. Although this kind of flexibility increases the overall performance of the system, still the implementation of VCs results in an area and possibly power overheads, besides the extra logic used to dynamically use them as ViChaR proposes.

In this paper we propose a new router that gets the benefit of all router buffers whether there are VCs or not, so we can just use the available input buffers but in a flexible way. If there is a contention (blocking to a request because the requested buffer is busy) at any input port, the Flexible router will try to allocate any suitable free buffer in other input ports in the router to store the incoming packet, thus the contention problem could be solved without waiting for the original requested busy input port buffer to be free. Using such an architecture we could improve the performance over a standard Base router under three different traffic patterns. We could also mitigate some problems associated with flexibility with a small side-effect which is receiving some packets out of order.

There are three main differences between the Flexible and the ViChaR routers. First, the ViChaR idea is applicable only if there are VCs implemented, while the Flexible router idea can be applied if there are VCs or even if there is just a single buffer per input port. Second, in ViChaR the contention can be solved using the VCs of the requested input port under consideration only, while the Flexible router can use all the available free buffers or VCs of the router not only the buffers or VCs of the requested input port to solve contention. Third, although our proposed router has a side-effect of receiving packets out of order in opposite to ViChaR which does not suffer from such side-effect. However, this side-effect is found to be limited and small as it will be analysed later in the simulation section.

2. Background

To understand well the operation of the Flexible router, we should first explain the Base router architecture and its operation.

2.1. Base router architecture

As shown in Fig. 1, it consists of five input ports and five output ports, connected together using the intermediate crossbar. The Flow_ctrl represents the request and grant signals, while Data_E,W,N,S are the packets from the upstream routers.

In a mesh topology, each input and output port is associated with a specific direction: East (E), West (W), North (N), South (S), and Local (L). The local input and output ports are connected to the network interface which is connected to the processing element (PE).

The structure and the functionality of the input and output ports is explained as follows:

Input port: Any input port consists of three main components (Fig. 2):

- *Input controller*: It is responsible for receiving requests (req_UP) from upstream router and giving the grants (gnt_US) if the FIFO is not full. Also it sends the internal requests (req_int) to the output ports and waits for their grants (gnt_int) in order to transfer the received packets to their downstream routers according to the routing logic used.
- *FIFO buffer*: To store the incoming packets (pkt_US) from the upstream router.
- *Routing logic*: Applies the routing algorithm to the header packet of the FIFO to determine the packet direction according to the destination address within the packet in order to choose the appropriate output port.

Output port: As shown in Fig. 3, it also consists of three main components:

- *Arbiter*: Receives all incoming requests (req_int_E,W,N,S,L) for the output port and then grants one of them (gnt_int_E,W,N,S,L) according to a prespecified logic. There are many arbitration logic functions stated in [12]. We chose the Round Robin logic because of its simplicity and fairness.

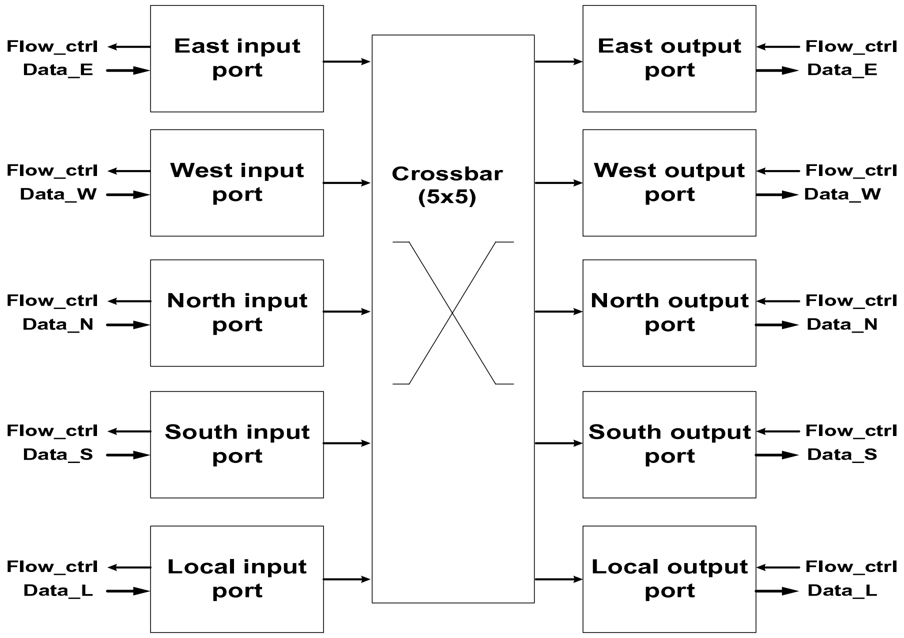


Fig. 1. Base router architecture.

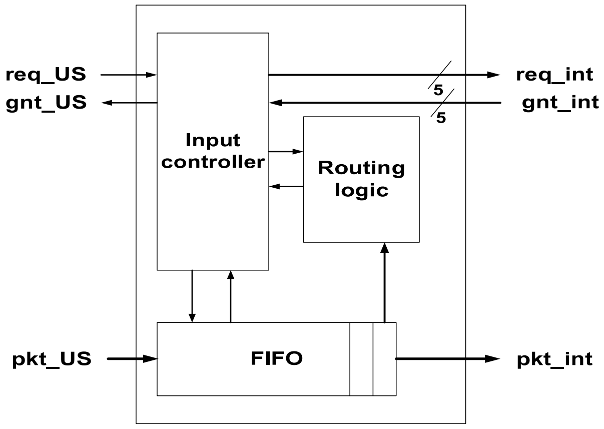


Fig. 2. Input port of the Base router.

- *Output controller*: Handles the communication with the downstream router (using req_DS and gnt_DS signals).
- *MUX*: Selects which packet goes to the downstream router according to the arbiter selection.

2.2. Basic operation of the Base router and its limitations

When a request (req_US) arises from one of the neighboring upstream routers, then the input controller will check the FIFO buffer for an available space. If the FIFO is full then the input controller will wait until at least one slot (one slot carries one packet) becomes free. At this moment the controller will set the grant (gnt_US) signal to the upstream router announcing that the packet is stored in the FIFO. After receiving the packet, the packet waits until it becomes the head of the FIFO then the destination address field in the packet is checked by the routing logic function to determine the appropriate output port. After determining a suitable output port, the input controller initiates a request (req_int) to that specific output port and waits to be granted.

Suppose that the output port has a number of simultaneous requests (req_int_E,W,N,S,L) from other input ports other than the request under consideration; it will select one of them according to the arbitration logic used. Now the arbiter will select one of these requests and trigger the output controller to initiate a request (req_DS) to the downstream router and wait to be granted.

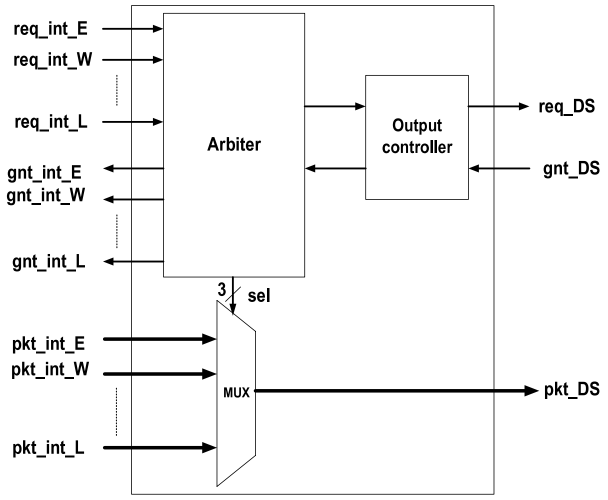


Fig. 3. Output port of the Base router.

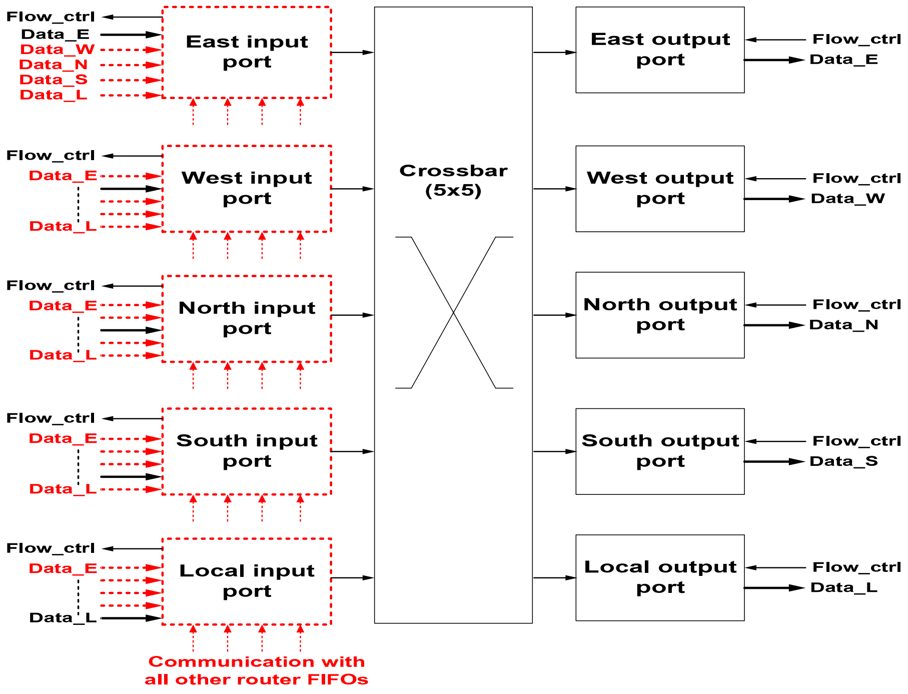


Fig. 4. Flexible router architecture.

Upon receiving the grant signal (gnt_DS), the arbiter raises the grant signal to the selected input port, so now this input port can check the head of its FIFO again and repeat the previous steps to transfer another packet.

The main limitation during the Base router operation is the contention problem. Contention occurs when a request at some input port is blocked because the requested FIFO of this port is full. Such contention may lead to other blockings in the network and hence congestion occurs [13], which degrades the performance of the overall network.

3. The proposed Flexible router architecture

3.1. Flexible router architecture

As shown in Fig. 4, the design is similar to the Base router except for some added functionality and modules to the input ports (the modified modules are shown with dashed frames and the added signals with dashed arrows). As shown there are also five input and output ports but now the input ports are different.

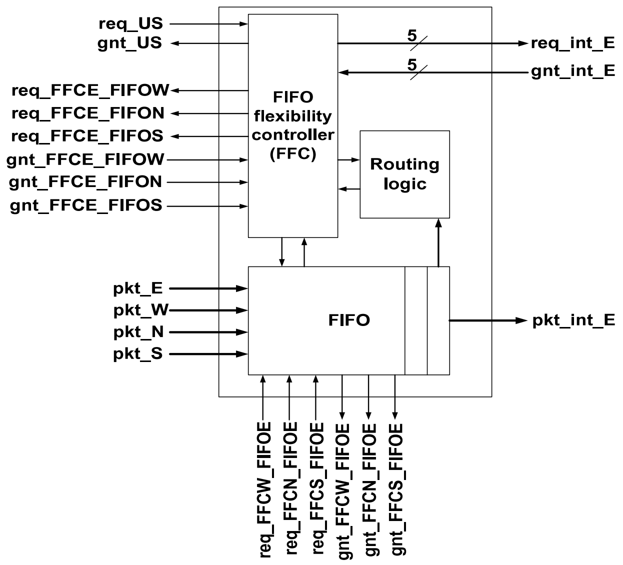


Fig. 5. The East input port of the Flexible router.

Input port: Fig. 5 shows the East input port as an example. It consists of three basic modules:

- *FIFO Flexibility Controller (FFC)*: This new added module is responsible for finding any suitable free FIFO in the router in order to store the incoming packet (using req_FFCE_FIFOW,N,S and gnt_FFCE_FIFOW,N,S request and grant signals respectively shown in Fig. 5). Finally, it is also responsible to communicate with the output ports (using req_int_E and gnt_int_E as shown in Fig. 5) to transfer the received packets to their downstream routers according to the routing algorithm used.
- FIFO buffer: The FIFO here can receive packets not only from the directly connected upstream router as the FIFOs of the Base router, but also from other input ports (by granting req_FFCW,N,S_FIFOE requests shown in Fig. 5 originating from other FFCs in the router).
- *Routing logic*: Applies the routing algorithm to determine the packet direction in order to choose the appropriate output port.

Output port: it is exactly the same as the output port of the Base router shown in Fig. 3.

3.2. Basic operation of the Flexible router

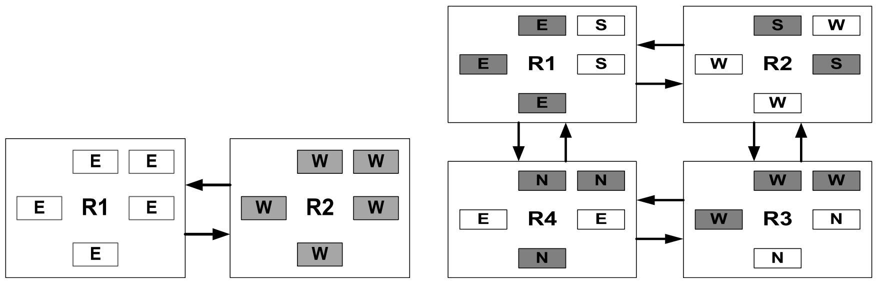
The operation of the Flexible router is the same as the operation of the Base router but in case of contention it will be different. Regarding contention, the Flexible router will not wait until the requested full FIFO to have one or more free slots as the Base router does, but now the FFC will search for a free slot in any suitable not full FIFO in the router (including the FIFO of its input port; because it may have one or more free slots while searching) by requesting (req_FFCE_FIFOW,N,S) the FIFOs that are not full in other input ports, and once it finds a free slot it grants back the request to the upstream router (gnt_US). Then the packet is transferred (one of the pkt_E,W,N,S in Fig. 5) to the selected FIFO. After that, the operation of the Flexible router will be exactly the same as the Base router.

3.3. Deadlock problem

If the Flexible router is designed to store any packet in any buffer regardless of its direction, deadlock may occur. For example, in Fig. 6(a), all packets stored in R1 are going to East and at the same time all packets in R2 are going to West, hence a deadlock occurs. Also a deadlock may occur between four routers as shown in Fig. 6(b).

To avoid the deadlock some restrictions must be made in terms of where the router can store the incoming packets. These restrictions are created based on the direction of the incoming packets. The idea is explained in more detail as follows:

An analogy was made between the buffers of the Base router and the buffers of the Flexible router in terms of what packet directions can each buffer have in the Base router while working under the *XY* routing algorithm. For example, in the Base router the East buffer can contain packets directed to North, South, West, or Local. Now according to that analogy, the East buffer of the Flexible router will be designed to accept packets directed to the same directions (North, South, West, or Local), and the same concept will be applied to the rest of the buffers.



(a) Deadlock between two routers. (b) Deadlock between four routers.

Fig. 6. Some deadlock situations in the full Flexible router under *XY* routing.

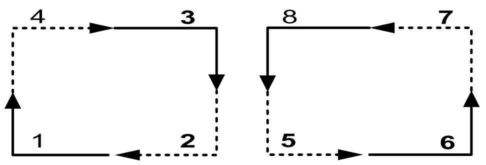


Fig. 7. Possible and forbidden turns in *XY* routing algorithm.

This solution is based on the turn model [14]. Some turns are prevented to avoid complete cycles and hence to prevent deadlock. Basically, the turn model is used to determine whether the routing algorithm under consideration is deadlock free or not. For example, if we apply this model to *XY* routing, all what we should do is to determine all possible turns that could occur in *XY* routing. Possible turns in *XY* routing are from East to North or South (turns 1 and 8 of Fig. 7 respectively) and from West to North or South (turns 6 and 3 of Fig. 7 respectively).

Now there are two forbidden turns in counterclockwise direction (turns 5 and 7 of Fig. 7), and another two turns in clockwise direction (turns 2 and 4 of Fig. 7) and hence complete cycles cannot occur in both directions making *XY* routing a deadlock free algorithm.

3.4. The Flexible router is a deadlock free router under XY routing

By applying the turn model on the Flexible router working under *XY* routing we can prove that it becomes deadlock free. The same steps will be followed, so we will exclude all possible turns that can occur in the Flexible router working under *XY* routing algorithm.

From the analogy with the Base router under *XY* routing, possible packet directions that each buffer can store in the Flexible router are as follows:

- *North buffer*: Can contain packets directed to Local or South. For packets directed to the Local port they reach their destination and are absorbed directly with the Local port. For packets with South direction this is not a turn because it is a vertical movement, therefore packets in the North buffer cannot make turns.
- *South buffer*: Can contain packets directed to Local or North. This is similar to the North buffer, and hence there are no turns originating from this buffer.
- *East buffer*: Can contain packets directed to Local, North, South, or West. Also here both Local and West do not make turns because the Local will be absorbed and the West direction is just a horizontal movement, but for North and South movements these directions represent two turns; turns number 1 and 8 respectively in Fig. 7.
- *West buffer*: Can contain packets directed to Local, North, South, or East; using the same concept as for the East buffer, this buffer contributes with turns 6 and 3 in Fig. 7 for directions North and South respectively.

According to the above discussion, the Flexible router could avoid four turns, 2 and 4 in clockwise and 5 and 7 in counterclockwise in Fig. 7. Therefore the Flexible router is a deadlock free router under *XY* routing.

3.5. Buffer-to-buffer deadlock problem

Another kind of deadlock can occur. We call this kind *buffer-to-buffer (b2b)* deadlock to differentiate it from the previous discussed one.

To explain the problem, suppose the example in Fig. 8, where R1 and R2 are two neighbor routers. Suppose also that packet P1 in FIFO B1 is requesting R2 and FFC2 chooses B2 for it. At the same time packet P2 in FIFO B2 of R2 was requesting R1 and FFC1 accidentally chose B1 for it. Both requests now enter an infinite waiting loops that will be broken only if the grant arises.

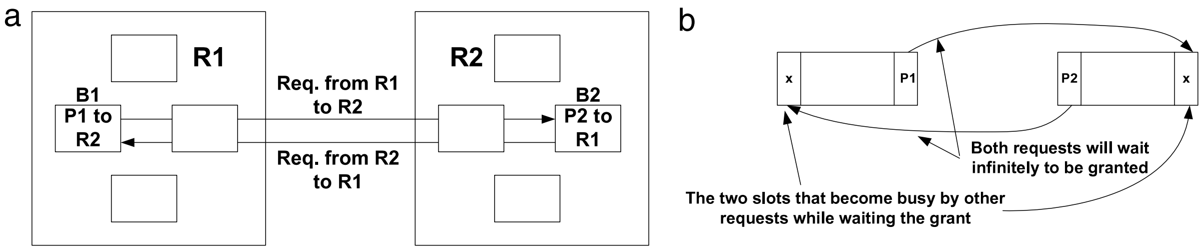

a
R1
R2
B1
P1 to R2
B2
P2 to R1
Req. from R1 to R2
Req. from R2 to R1
b
x
P1
P2
x
Both requests will wait infinitely to be granted
The two slots that become busy by other requests while waiting the grant

Fig. 8. Buffer-to-buffer deadlock problem.

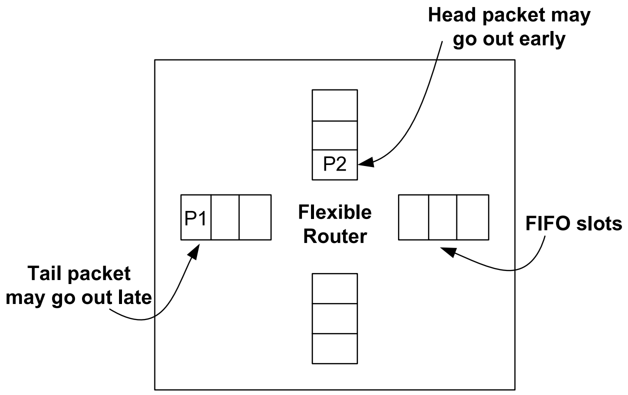


Fig. 9. Two packets (P1,P2) become out of order in the Flexible router.

According to our design, the FIFO responds to the requests in a sequential order like the following pseudo code:

```
if (Req1 & FIFO is not full)
  {store the packet and check if full or not;}
if (Req2 & FIFO is not full)
  {store the packet and check if full or not;}
if (Req3 & FIFO is not full)
  {store the packet and check if full or not;}
if (Req4 & FIFO is not full)
  {store the packet and check if full or not;}
```

Now, suppose that: a) Each of B1 and B2 has just one free slot; b) Requests from P1 and P2 are in lower order than the other requests from other ports; 3) There are requests to B1 and B2 from other ports.

In this situation, B1 and B2 will respond to the other requests and then both of them become full. Hence neither the request from P1 nor P2 is served causing deadlock as both of them depend on each other, i.e. to store P1 in B2, P2 must leave B2 while P2 can leave B2 if P1 leaves B1 to a free slot in B2. Hence deadlock dependency occurs.

To recover from this problem, one way is to check the status of the FIFO while waiting the grant and if the FIFO becomes full the FFC will break the loop and search for a new FIFO, like the following pseudo code.

```
While (grant = 0)
{
    if (FIFO becomes full) {Cancel the request; Break the loop; }
}
```

3.6. Out of order received packets

Due to the flexibility of the Flexible router, packets can reach their destinations out of order. For the example in Fig. 9, we have two packets, packet 1 and packet 2 generated in this order from the same source and going to the same destination. Due to flexibility, these two packets can be in the same router at the same time but in different FIFOs. Therefore, according to the arbiter decision packet 2 may go out from the required output port before packet 1, especially if it was in advanced FIFO position closer to the head position than packet 1. It is worth to say that this side-effect is limited and practically insignificant at low injection rates as we will see later in Section 4.4.

Destination X	**Destination Y**	**Source X**	**Source Y**	**Packet sequence**	**Time of transmission**	**Random information**

Fig. 10. Packet structure.

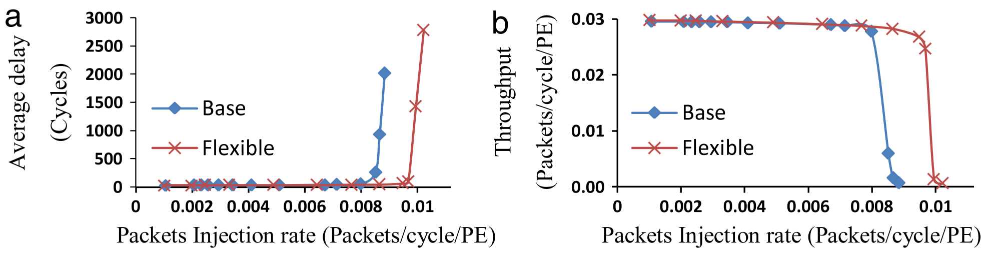


Fig. 11. Delay (a) and throughput (b) characteristics versus injection rate under HS traffic.

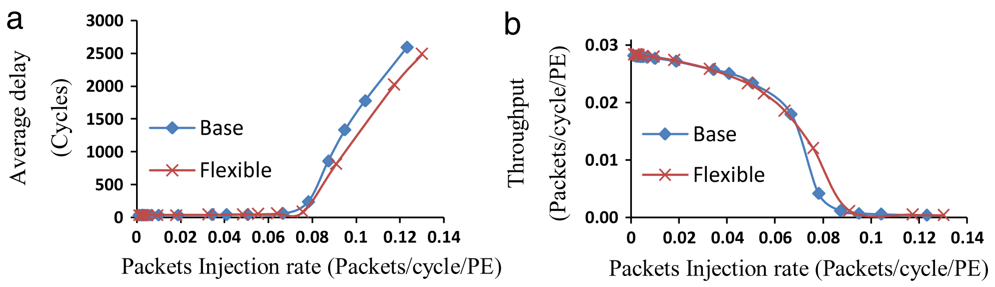


Fig. 12. Delay (a) and throughput (b) characteristics versus injection rate under UNI traffic.

4. Simulation platform and results

4.1. Simulation setup

To perform the simulation, we implemented a NoC simulator using systemC. We use a 5×5 mesh topology to perform the simulation. The FIFO size used was 5 packets except the local FIFO which is implemented large enough (assumed to be infinite) to facilitate the calculation of the injection rate. The data link width between any two routers is equal to one packet, so a complete packet can be transferred in one cycle.

We simulate under *XY* routing algorithm and Store-and-Forward (SAF) switching technique. Three different traffic patterns were used: Hotspot (HS), Uniform (UNI), and Nearest-Neighbor (NN). For HS, 90% of the traffic is directed to the hotspot node and the rest of the traffic is equally distributed between all other nodes, while in UNI, all the traffic is equally distributed between all nodes. In NN, any node sends only to its neighbor nodes. Each router is connected to a PE which injects packets (typically 1000) to other PEs according to the traffic pattern used, and also receives packets from other PEs. The intermediate time between any two successive transmitted packets is chosen to have a uniform distribution.

Finally, any packet is composed of seven fields as shown in Fig. 10:

- Destination X (Destination Y): The destination node X (Y) coordinate.
- Source X (Source Y): The source node X (Y) coordinate.
- Packet sequence: The sequence number of the packet.
- Time of transmission: The time at which the packet is transmitted (in cycles).
- Random information: A random number that models the information within the packet.

4.2. Simulation results and analysis

Average delay and throughput comparisons between the Flexible and the Base routers under HS, UNI and NN traffic patterns are shown in Figs. 11(a), (b), 12(a), (b) and 13(a), (b) respectively.

For low injection rates both Base and Flexible routers have nearly the same average delay and throughput and the benefit of flexibility just appears at higher rates. This is because at low rates the number of packets injected into the network per

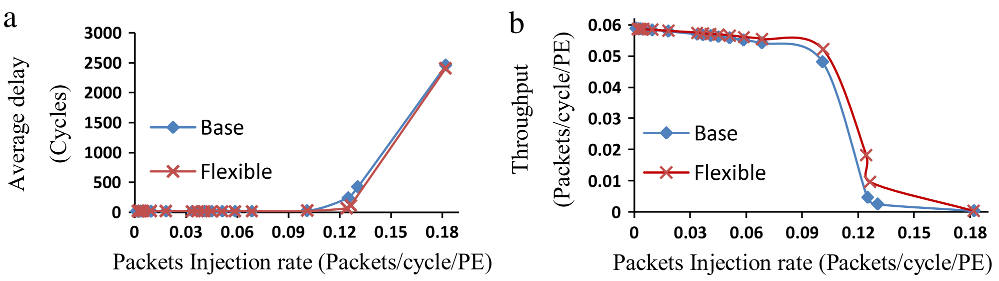


Fig. 13. Delay (a) and throughput (b) characteristics versus injection rate under NN traffic.

Table 1
Area comparison.

FPGA resources	Number of resources used		Percentage increase%
	Base	Flexible	
LUTs	506	596	17.8
FFs	308	344	11.7

cycle is small, therefore the number of contentions will be small as well, and therefore the flexibility advantage will not be used a lot. At higher injection rates, the congestion increases therefore the average delay increases as well and the throughput decreases. At some point the router saturates and the delay becomes very high, so we call this point the saturation point. The Flexible router saturates at higher injection rates than the Base router case. For HS traffic there is 11.4% increase in the saturation rate while for UNI and NN it is a small improvement.

We think that both UNI and NN are not well improved for two reasons:

1. The communication overhead due to FFC; the upstream router must send the request to the FFC and the FFC by turn will search for the free slot and give the grant back to the upstream router. However, if the upstream router could search by itself directly, this communication overhead could be decreased and the performance could be improved.
2. The second reason is the b2b deadlock problem which causes some congestion because when it occurs; the packet spends some cycles with no value then a new request is initiated to another FIFO. This may affect other packets behind this packet and results in more delay in many packets exactly as the congestion due to blocking does.

4.3. Area overhead

Both Base and Flexible routers were designed in Verilog HDL, simulated by ModelSim from MentorGraphics and synthesized using Xilinx ISE 13.2 compiler. The target platform is Virtex-5 FPGA (65 nm), device number xc5vfx70t-1ff1136. The resources comparison in terms of Flip Flops (FFs) and Lookup Tables (LUTs) is shown in Table 1. It can be observed that the area is increased by 17.8% in (LUTs) and 11.7% in (FFs). Hence the area overhead is reasonably small.

4.4. Out of order packets analysis

We have studied this side-effect to know how much it affects the Flexible router operation. The HS traffic pattern was used because for HS the probability of having two successive packets generated from the same source and going to the same destination is high. This experiment is made under the saturation injection rate and the total number of sent packets is 25 000. In this experiment, different lagging distances (the difference in position between the correct and the incorrect order of the packets) are determined along with the number of out of order packets reached with each lagging.

In Fig. 14, the number of out of order packets for each lagging distance is written at the top of each column. As shown, most of the out of order packets (402) reached with lagging 1 and just 10 packets reached with lagging 2 while there is just one packet reached with maximum lagging 3. Totally, there are 413 out of order packets out of 25 000 the total number of sent packets, which means just 1.65%. Also most of them reach with the minimum lagging 1, while there is just one packet reaches with maximum lagging 3. According to such results we can conclude that this side effect is limited and can be easily corrected at the destination node.

4.5. Comparison

Table 2 contains a comparison summary between the Base and the Flexible routers.

5. Conclusion and future work

We have proposed a new Flexible router architecture which can mitigate the contention problem by the efficient use of free buffers in the router. This router shows higher performance for Hotspot, Uniform and Nearest-Neighbor traffic patterns,

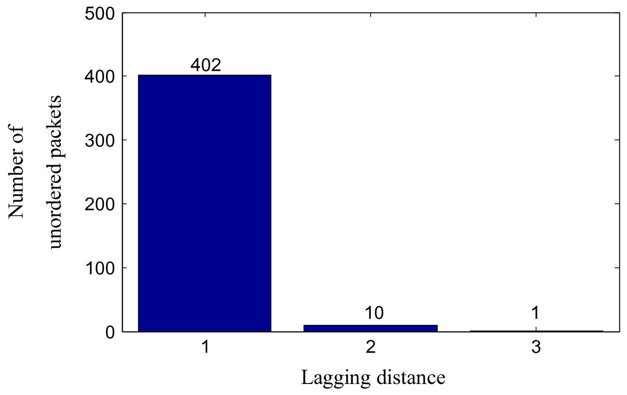


Fig. 14. Lagging distances under saturation injection rate under HS traffic.

Table 2
A comparison summary between the Base and Flexible routers.

Router	Performance			Area		Packets reordering
	HS	UNI	NN	FF	LUTs	
Flexible	Better	Better	Better	344	596	Might be required
Base	Worse	Worse	Worse	308	506	Not required

increasing the saturation rate by 11.4% for HS, with a reasonable area overhead (17.8% increase in LUTs and 11.7% increase in FFs). Some packets reach out of order but this side-effect was not critical. For Hotspot traffic, the out of order packets are just 1.65% of the total number of packets at the saturation injection rate with minimum lagging (1) in most cases.

As a future work, we intend to improve the Flexible router design to decrease the overhead due to the FFC and the b2b problem in order to improve the performance. Also we intend to evaluate the performance of the Flexible router under other routing algorithms and traffic patterns. Moreover ways to eliminate or control the lagging distance of the unordered packets will also be considered.

References

[1] I. Saastamoinen, M. Alho, J. Nurmi, Buffer implementation for Proteo network-on-chip, in: Proceeding of the International Symposium on Circuits and Systems, ISCAS'03, vol. 2, 2003, pp. II-113–II-116.
[2] H. Jingcao, R. Marculescu, Application-specific buffer space allocation for networks-on-chip router design, in: Proceeding of the IEEE/ACM International Conference on Computer Aided Design. ICCAD, 2004, pp. 354–361.
[3] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, S. Borkar, A 5-GHz mesh interconnect for a teraflops processor, in: Proceeding of the IEEE MICRO, vol. 27, September–October 2007, pp. 51–61.
[4] H. Wang, X. Zhu, L. Peh, S. Malik, Orion: a power-performance simulator for interconnection networks, in: Proceeding of the 35th IEEE/ACM Annual International Symposium on Microarchitecture, MICRO-35, November 2002, pp. 294–305.
[5] N. Banerjee, P. Vellanki, K. Chatha, A power and performance model for network-on-chip architectures, in: Proceeding of the Design, Automation and Test in Europe Conference and Exhibition, DATE, vol. 2, February 2004, pp. 1250–1255.
[6] X. Chen, L. Peh, Leakage power modeling and optimization in interconnection networks, in: Proceeding of the 2003 International Symposium on Low Power Electronics and Design, ISLPED'03, August 2003, pp. 90–95.
[7] T.C. Huang, U.Y. Ogras, R. Marculescu, Virtual channels planning for networks-on-chip, in: Proceeding of the 8th International Symposium on Quality Electronic Design, ISQED'07, March 2007, pp. 879–884.
[8] J. Kim, Low-cost router microarchitecture for on-chip networks, in: Proceeding of the 42nd IEEE/ACM International Symposium on Microarchitecture, MICRO-42, December 2009, pp. 255–266.
[9] M.P. Véstias, H.C. Neto, A dynamic buffer resize technique for networks-on-chip on FPGA, in: Proceeding of the VII Southern Conference on Programmable Logic, SPL, April 2011, pp. 227–232.
[10] C.A. Nicopoulos, et al. ViChaR: a dynamic virtual channel regulator for network-on-chip routers, in: Proceeding of the 39th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO'06, December 2006, pp. 333–346.
[11] Y. Tamir, G.L. Frazier, High-performance multiqueue buffers for VLSI communication switches, in: Proceeding of the 15th Annual International Symposium on Computer Architecture, ISCA, 1988, pp. 343–354.
[12] W.J. Dally, B. Towles, Principles and Practices of Interconnection Networks, Morgan Kaufmann, 2004.
[13] L. Benini, G. De Micheli, Networks on Chips: Technology and Tools, Morgan Kaufmann, 2006.
[14] C.J. Glass, L.M. Ni, The turn model for adaptive routing, in: Proceeding of the 15th Annual International Symposium on Computer Architecture, 1992, pp. 278–287.