A Novel Priority-Driven Arbiter for the Router in Reconfigurable Network-on-Chips

Hung K. Nguyen, Xuan-Tu Tran
SISLAB, VNU University of Engineering and Technology, Vietnam National University, Hanoi
144 Xuan Thuy road, Cau Giay district, Hanoi, Vietnam
Email: {kiemhung, tutx}@vnu.edu.vn

***Abstract*—The Network-on-Chip (NoC) paradigm has been emerging as new communication solution for the Ultra Large-Scale Integration System-on-Chips (SoCs). Many real-time embedded SoCs support multi-rate applications that requires the NoC has to offer different Quality-of-Service level. In this paper, we propose and implement a priority-driven arbitration mechanism for the hybrid switching router used in reconfigurable NoCs. The router supports both guaranteed-throughput (GT) service and best-effort (BE) service without reserving resources for GT service. In addition, the proposed solution not only guarantees GT-type QoS, but also enhance the average performance for BE service by using communication resources efficiently. The router has been modelled and then synthesized on Xilinx technology (using Virtex-7 FPGA). The obtained results show that this router can guarantee reliability and enhance significantly the average performance of BE load compared with the generic router while the overheads in terms of area and power consumption are acceptable.**

Keywords—Priority-driven arbitration; Quality-of-Service (QoS); Reconfigurable Network-on-Chip; System-on-Chip

I. INTRODUCTION

Emerging System-on-Chips (SoCs) are typically battery-powered systems and must support a wide range of applications such as communication baseband processing or multimedia processing. The high performance and low power consumption are becoming two key requirements when designing such devices. New SoC architectures should be able to support multiple applications with a high performance, while maintaining low-power consumption, low area cost, low non-recurring engineering cost, and shorter time-to-market. These architectures should also offer the capability of hardware updating after systems have been deployed and should be able to resolve the appearance of defects or faults in the system to guarantee the correct operation of the system.

Heterogeneous SoCs were proposed to meet the above requirements [1]. They are usually composed of several types of processing elements, such as software-programmable processors, application-specific IP cores, reconfigurable hardware, and so on. To establish the communication between these processing elements, the Network-on-Chip (NoC) paradigm [2],[3] has been proposed to achieve not only better performance and lower energy consumption but also higher flexibility in comparison with the conventional on-chip bus architectures. When on-chip integration is more and more increasing with billion transistors in the future, the on-chip communication will determine the performance of a system [4]. Therefore, the on-chip communication should aim at providing high throughput, high scalability, low latency, low power consumption, reduced contention, ensured Quality-of-Service (QoS), less occupied area, etc. In the literals, some design methodologies have been proposed to deal with NoCs and can be classified into two main categories: (i) design-time methodologies (e.g. [5] and [6]); and (ii) run-time methodologies (e.g. [7],[8],[9],[10], and [11]). The design-time methodologies generally aim at designing the NoC architectures for a specific application. Unfortunately, the application-specific NoCs are not flexible enough to support dynamic environments where communication characteristics need to adapt smoothly to various contexts at runtime. In the last decade, researchers have been focusing on the development of run-time methodologies for reconfigurable NoCs. These methodologies provide the techniques which allow NoCs to autonomously adapt their structure and behavior during the operating time. The elements of a NoC that can be modified at run-time include reconfigurable topology [7], reconfigurable links [8], and reconfigurable router architectures [9],[10],[11].

The main objective of NoC design is to find suitable NoC instances to get the best trade-off between the cost (area, power) and performance (latency, throughput, and reliability). Pratomo *et al.* in [12] built scenarios for evaluating of the impact of NoC parameters (packet rate, packet size, buffer size, routing algorithm) on the performance of NoC. Le-Van *et al.* in [13] also developed an evaluation and simulation platform at high-level design to quickly evaluate the performance of NoC architectures with different parameters. Liu *et al.* in [14] drew a comparison between circuit-switched and packet-switched NoCs. In real-life applications, almost most of parameters (routing algorithm, network topology, communication load, buffer size, switching diagram, packet rate, etc.) are determined in the NoC router. Thus, one of the most critical issue in designing NoC is the design of an efficient and high-performance router. In reconfigurable NoCs, a reconfigurable router architecture is needed to adapt to the immediate status of network so that the performance is not degraded while the flexibility will be increased. For example, in situation of faulty network, the reconfigurable router can adapt the NoC to the other operation mode by adjusting the number of virtual channels [9], or changing the type of its routing algorithm and switching scheme [15],[16].

In this paper, we propose a hybrid switching router architecture with the priority-driven arbitration mechanism. This

router can dynamically reconfigure its switching scheme to the wormhole switching, the virtual cut-through switching or combination of both schemes depending on the traffic load and network status. In addition, the hybrid switch arbiter which can exchange flexibly between arbitration modes to support both Best-Effort (BE) and Guaranteed-Throughput (GT) services depending on the traffic scenario. The experimental results have proven that the proposed router improves the total performance at an acceptable expense of the implementation cost.

The rest of this paper is organized as follows. The problem definition and the proposed solution are addressed in Section II. Section III introduces the proposal of reconfigurable router. Section IV presents the implementation and evaluation of the proposed router in comparison with the related works. Finally, some conclusions are drawn in Section 5.

II. Problem and Solution

The Network-on-Chip is the communication among on-chip components and must satisfy QoS requirements and implementation cost [3]. Network QoS parameters consist of latency, throughput, and reliability, whereas the implementation cost is evaluated by power consumption, area size and efficiency of using resources. The QoS is classified into two categories: GT service and BE service.

The BE service implies that the network tries to achieve minimum delay for transporting a packet from a source node to a destination node. The actual delay is determined not only by the distance between the resources but also on the other traffic in the network. In the BE service, packets are forwarded as soon as possible without the reserved resources. Most of the NoC architectures offer BE services based on packet-switching because it efficiently utilizes the available bandwidth. Unfortunately, BE service may not be acceptable to real-time applications. By contrast, the GT service assures both throughput and latency over a finite interval and supports uncorrupted, lossless, and ordered data transfer by reserving the resources between source and destination. GT services are usually provided by circuit-switching [17] or by using connection-oriented packet-switching [18] (i.e. virtual circuit-switching at where some virtual channels are dedicated to GT-type packets). For example, Æthereal proposed by Goossens et al. [17] is another mesh-based NoC that supports GS router and a BE router in parallel to guarantee throughput traffic. It is obvious that the area overhead of this architecture is very high. Moreover, many real-time applications (such as capturing video from a camera on the smartphone) produce bursty traffic that sends large quantity of data during some intervals and less or no data during the others. A reserving-based NoC would take a significant part of the systems silicon area and only a fraction of its capacity is utilized by a given application.

To trade-off between cost and high performance, recent NoCs are usually built on the packet-switching mechanism. Two popular packet-based switching schemes are the virtual cut-through (VCT) and the wormhole. In wormhole switching, the packet is divided to flits including one header flit, followed by one or more body flits, and one tail flit. To reduce the implementation cost, the wormhole routing scheme normally uses just 1-flit depth for buffering. Consequently, the packet spreads over multiple different routers along its path just like a worm. In VCT switching, the same method as wormhole is adopted. However, the significant difference is that the VCT provides a buffer enough to store the whole packet at several nodes in the network. On the other hand, each packet is also assigned a certain priority so that the arbiter can grant the switch to only one specific packet when many packets require the connection to the same output port in order to ensure the QoS. The highest priority level is assigned to the packet that requires high throughput and low latency to response the real-time constraints. For example, Vellanki *et al.* [18] proposed a VC (virtual channel)-based router architecture for supporting QoS in the mesh-based NoC. This proposal always reserves two of four VCs for supporting GT services. In addition, when GT load increases high, they can transfer through the other VCs, but not vice versa. By this way, the QoS of GT load is ensured but at the expense of degradation of average performance for BE load. Moreover, the efficiency of using hardware is not high because some reserved VCs cannot be used for BE load even if they are not used for GT load.

The problem with the above NoC is starvation of BE load. It happens when a packet cannot reach its destination because some resources do not grant access to it due to its low priority. In consequence, the starved packets can block many other packets leading to the degradation in performance and efficiency of using resources. This problem is extremely serious in the NoC using the wormhole switching scheme that allows a packet to spread over several different routers. Hybrid packet-switching technique [19] can deal with the above issue. However, this technique is still insufficient to improve the average performance for BE load.

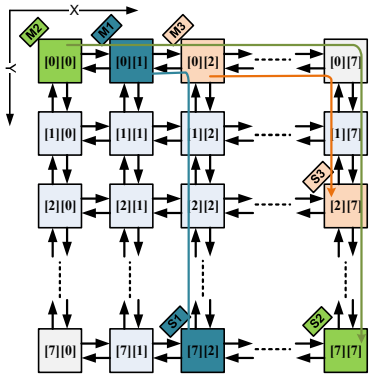

Fig. 1. An application scenario of the NoC-based system.

To understand this situation, let's examine a simple NoC-based system as shown in Fig. 1. The NoC performs three communication connections: M1 sends packets P1 (denoted by blue color) to S1; M2 sends packets P2 (i.e. green color) to S2; M3 sends packets P3 (i.e. yellow color) to S3. The highest priority is assigned to the packet P3 and the lowest priority is assigned to the packet P1. If the routing scheme is based on the XY-routing algorithm, so the routing path of P1, P2, and P3 is shown by the blue, green, and yellow line, respectively.

Assuming that at time t_0 there is only M1 ready, so it is granted physical links to transfer packets P1. At time t_1, M2 becomes ready, because it has a higher priority than M1, it preempts M1 and begins transferring. At time t_2, M3 are ready. Because P3 is the highest-priority packet, it is guaranteed to transfer until it finishes and therefore P2 must wait in the queue. Timing diagram of transferring packets on the NoC is shown in

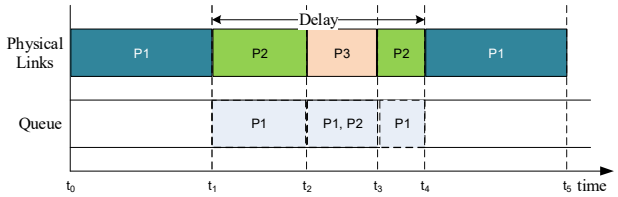

Fig. 2. Now, we zoom in the connection between Router[0][1] and Router[0][2] as shown in Fig. 3. At interval [t_2, t_3], although links from the west port to the south port of the Router[0][2] is available for M1, but P1 cannot preempt P2 to transfer via link Router[0][1] – Router[0][2] because of its lower priority. Only after both M3 and M2 finish, P1 just can be transferred. Consequently, the average performance and efficiency of using resources are degraded. The efficiency of using resources, and therefore average performance of P1 may be improved if the router supports the arbitration mechanism that allows P1 to be transferred in parallel with P3.

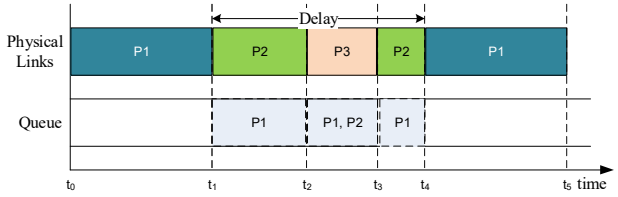

Fig. 2 Schedule of tranferring packets on NoC.

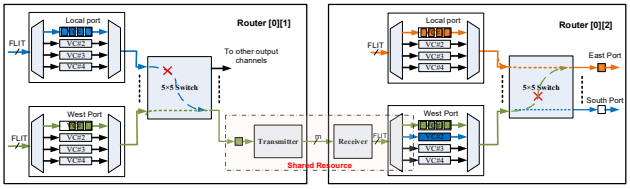

Fig. 3. The resource occupied by a blocked packet in wormhole routing.

To deal with the above problems, in this paper, we enhanced the hybrid switching router [19] by equipping it an arbiter with a priority-driven arbitration mechanism. The router therefore supports both GT and BE services without reserving resources for GT load. When a node need to inject high speed load, it encapsulates data into GT-type packets so that these packets are given higher priority over other packets to be transferred on the NoC. In addition, the arbiter has two operation modes: *Pre-emptive* and *Priority inheritance*. Pre-emptive mode helps to guarantee the QoS for GT load, meanwhile Priority inheritance mode helps to enhance the average performance for BE load by using resources more efficiently.

III. The Proposed Architeture

A. Router micro-architecture

In this work, we propose reconfigurable router based on the state-of-the-art virtual channel router micro-architecture [3] (which is referred as the generic router). Fig. 4(a) shows the Input/Output (I/O) interface of the proposed reconfigurable router for the 2D-mesh NoCs. Each router consists of four I/O ports (North, East, South, and West) for connecting to four neighbour routers and a Local port for connecting a processing element. Compared with the generic router, each port of the proposed router is assigned an integer-valued priority level. By default, the Local port has the highest priority and the West port has lowest one. However, this priority can be configured at run-time by writing a proper configuration word to the SW arbiter (switching arbiter).

The micro-architecture of the proposed reconfigurable router is shown in Fig. 4(b) with only one couple of input channel and output channel. In general, the generic architecture is partitioned into three main modules: *crossbar; input channels;* and *output channels*. It is popular that each router port consists of a couple of I/O channels for exchanging data with neighbour routers or local processing elements. The I/O channels are connected to physical links via the *receiver* and *transmitter* that adapt the bandwidth of the links to the flit size. The data from *receiver* is written to the *input buffer*. The input buffer is composed of four Virtual Channels (VCs), each with its own FIFO (First-In First-Out) buffer. One of VCs is chosen to buffer an incoming packet by the *Write control* unit. *Data Flow Control* (DFC) unit informs the available status of VCs and determines which VC channel is selected to proceed to the next stage. After a VC is ready, DFC detects and decodes the packet's header flit and then performs the given protocol to establish the physical channel and control transferring flits from the Input channel to the Output channel through the Crossbar. *Routing Computation* (RC) unit takes charge of look-ahead calculating the next router which the packet will be forwarded to. The routing is based on the target address in each header flit, and the XY-routing algorithm. After traversing across the crossbar, the packet is placed in the output buffer of the Output channel module before being sent to the next router by the transmitter. The *VC allocation logic* unit finds and assigns an available VC of the received router that the packet will be written to.

Comparing with the generic router, our proposed router offers a *configurable input buffer*, adds *conflict sensing (CFS) unit*, and *configuration controller* to its micro-architecture [19]. These improvements allow the router to be configured between VCT switching and wormhole switching at run-time to deal with the network congestion. The configurable buffer is composed of an array of storage elements that can be flexibly organized into four VCs with variable size depending on target applications. The CFS unit helps DFC to monitor possible conflicts when many VCs try to access to the same shared resources. When DFC detects a conflict, it will send a request for reconfiguring the input buffer to the configuration controller. The configuration controller takes in charge of controlling operation mode of the other modules. Depending on the status information collected during the period of the router's operation, the controller will make decision on switching scheme to adapt the router to the dynamic status of the NoC. In addition, the router is enhanced with a priority-driven *SW arbiter* to improve the average performance of the NoC. The details of the arbitration

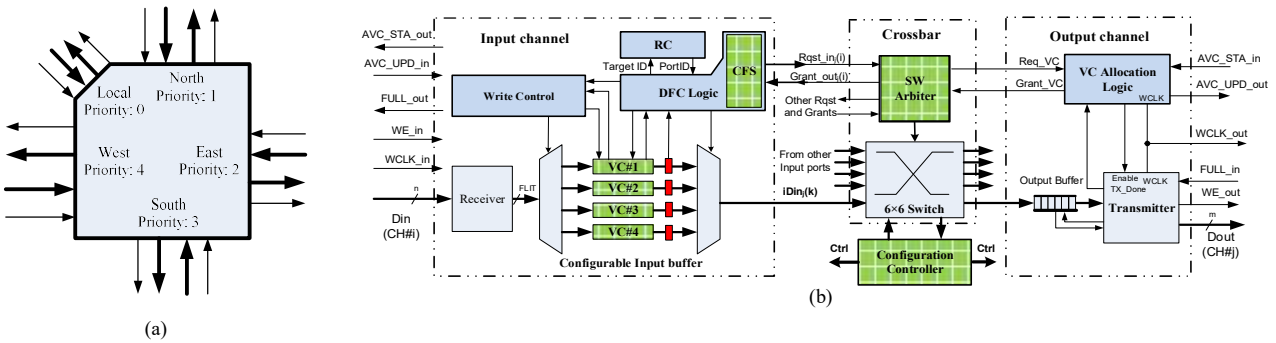

Fig. 4. (a) The I/O interface; and (b) micro-architecture of the proposed reconfigurable router.

mechanism will be described in the next sub-sections.

B. Packet format

As mentioned, the packet is divided into a header flit, followed by body flits and one tail flit. The size of a flit is 35 bits. Where, the most significant bit (i.e. bit 34th) always shows the network layer the packet belongs to. If this is '1', the packet belongs to the application layer, so it must be forwarded to the IP core at the destination router; otherwise the packet belongs to communication layer, so it must be forwarded to the configuration controller at destination router. Two next bits of every flit (i.e. bit 33th and bit 32th) are used for the field *ToF* (*Type of Flit)* that indicates the type of this FLIT.

Header flit contains information for controlling how the packet is travelling on the NoC. The structure of a header flit for the application packet is depicted in Fig. 5. Control information in the header flit includes: (1) co-ordinate (Y, X) of target router; (2) Port_ID that is used by next router to specify where the packet is forwarding to; (3) Type of Packet (ToP); (4) Length of Body (LoB); and (5) Source node ID. Especially, header flit contains two fields, called *GT/BE* and *Priority,* to aim at providing both GT service and BE service simultaneously. Here,

- Bit 22 (GT/BE) specifies the QoS type of a packet. The switch arbiter uses this bit to make decision on granting the switch for the packet. More detail about the role of this bit is discussed in the sub-section C.
- Bit 21 and bit 20 defines priority of a GT-type packet. These bits are valid only when bit GT/BE=1.

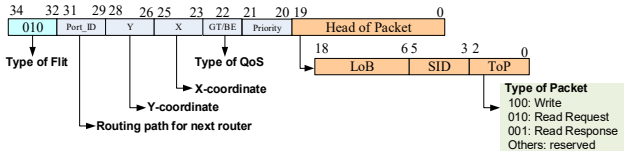

Fig. 5. Header flit of a packet in the application layer.

C. SW Arbiter

SW arbiter determines which VC channel of Input Port is selected to traverse over the switch. The judgment of the arbiter is based on the bits, *GT/BE* and *Priority*, in the Header flit of a packet. The arbiter support two operation modes:

- **Round-robin mode**: providing BE-type QoS. Ready VCs are kept in a queue and scheduled one after the other. Round-robin mode provides a form of fairness in that all VCs get a chance to send data to the next stage. However, it does not guarantee the throughput of any packet. Once being granted channel, the VC occupies the channel till entire packet is transmitted.
- **Priority mode**: providing GT-type QoS. Each GT-type packet is assigned an integer-valued priority that determines the priority of the VC in which the packet is being buffered. Because a VC can buffer various packets, therefore, the priority of each VC changes in time. The VC granted the switch is the highest priority one in the list of ready VCs. This mode, in turn, is divided into two sub-modes to not only guarantees QoS of GT load but also enhance the average performance for BE load by using resources more efficiently as follows:
  - *Pre-emptive mode* allows a packet to preempt a lower-priority packet. This mode deals with the case at which a higher-priority packet is blocked because the resource it requires has been granted to a lower-priority packet, and therefore guarantees QoS in real-time applications. This mode is only provided for GT packets which are distinguished from PE packets by setting the bit GT/BE in the header flit.
  - *Priority inheritance mode* allows a BE packet can be promoted temporarily to the priority of the GT packet that is blocked by a higher-priority GT packet. As soon as the priority has been upgraded, the BE packet can establish a channel for transmission until it is completed, or the blocked GT packet becomes unblocked.

The functional block diagram of the arbiter is shown in Fig. 6. At the priority mode, the scheduler takes charge of assigning proper priorities to VCs depending on the QoS type of each packet. Based on the priority assigned to VCs, the *Priority Decoding Logic* arbitrates which VC is served. In addition, the scheduler also makes decisions about whether upgrading the priority of a package or not, depending on the conflict signal from the CFS block.

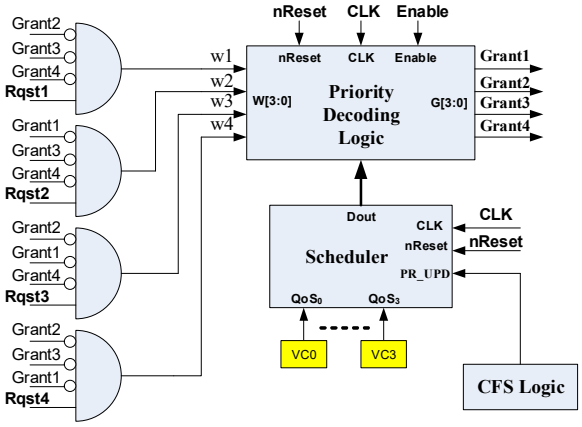

Fig. 6. SW arbiter.

D. Conflict sensing unit

The function of *Conflict Sensing (CFS)* unit is to detect possible conflicts on the output channel when there are many requests sent to the same shared resource. *CFS* unit basically includes two main units that are *slot timer* and *counter*. Each request is assigned a short interval, called time slot and defined by *slot timer*, to wait for the grant signal. If the request is not granted after the assigned time slot, it will be changed to the pending state and others request will continue to be processed. Each the failed request is tried again to get grant signals in *N* times that is determined by the *counter*. Once a request is failed, the *counter* associated with it will be started. When *counter* count to *N* and the request still has not get the grant signal, this means the request is not granted after N tries, *CFS* will send the configuration controller a request to reconfigure the operation mode of the router. The count value of slot timer and counter can be re-configured by the configuration controller.

IV. Implementation and Evaluation

A. Simulation Enviroment

The proposed router has been evaluated in terms of performance and implementation cost using the HDL-based simulator. To do that, a 2D-mesh 8×8 NoC evaluation platform (as shown in Fig. 1) have been built from the RTL model. Besides, we also developed a Network Interface (NI) [19] with built-in dummy IP cores.

Based on the platform in Fig. 1, we defined a script to evaluate the performance and implementation cost of the proposed router. The packet size is set to ten 35-bit flits. The buffer size of each input port is equal to sixteen 35-bit flits; therefore, the depth of each virtual channel is 4 flits in normal operations. There are three scenarios have been defined in our scripts as follows:

- *Scenario 1*: Only M1 transmits data to S1 at the maximum rate of 0.04 packets per cycle.
- *Scenario 2*: In addition to the settings of Scenario 1, M2 is also enabled to transmit data to S2 at the maximum rate of 0.04 packets per cycle. The number of packets P2 is set to a half of the number of packets P1.
- *Scenario 3*: In addition to the settings of Scenario 1 and Scenario 2, M3 is also enabled to transmit data to S3 at the maximum rate of 0.04 packets per cycle. The number of packets P3 is set to a quarter of the number of packets P1.

B. Experimental Results

The proposed router have been synthesized by the Vivado Design Suite (Xilinx). The synthesis result of the proposed router using the Virtex-7 XC7VX485 FPGA chip is shown in TABLE I. It takes about 0.4%, 1.13%, and 0.26% of the XC7VX485 chip resource in terms of Flip-Flop, LUTs, and Memory LUT, respectively. Compared with the generic router, the resource utilization overhead of our proposed router increases about 12.35%, 15.15%, and 13.7% in terms of Flip-Flop, LUTs, and Memory LUT, respectively. The maximum frequency of the proposed router is approximate to 108*MHz*. The average latency for transferring the header flit and body flit through one router is 12.5 and 5.5 cycles, respectively. The power consumption is estimated about 0.246*W*, increasing 6% compared with the generic router.

TABLE I. Synthesis Result on Virtex-7 FPGA Technology

Resource Type	Used Resource		Area overhead (%)	Used/Available Ratio (%)
	Generic	Configurable		
Flip-Flop	2181	2450	+12.35	0.40
LUTs	2976	3427	+15.15	1.13
Memory LUT	295	335	+13.7	0.26
Power (W)	0.232	0.246	+6.03	-

The comparison in respect of throughput and latency between the generic wormhole router and the reconfigurable router is provided in TABLE II. In this table, the latency is defined as the time taken for one packet to travel from a source to a destination while the throughput is the network's transfer rate and is evaluated as flits per cycle. These values of each communication pairs (M1-S1, M2-S2, and M3-S3) are measured at the target nodes S1, S2, and S3, respectively. From the definition of scenarios, it is obvious that *Scenario 1* has no conflict, therefore, there is no difference between the generic router and the reconfigurable router. In *Scenario 2*, M1-S1 communication must compete the link between Router[0][1] and Router[0][2] with the M2-S2 communication. Because the priority of M2-S2 communication is higher than M1-S1 communication, the performance of M1-S1 communication is decreased. There is not a significant difference in terms of throughput and latency between the generic router and reconfigurable router. In *Scenario 3*, M2-S2 communication also must compete with M3-S3 communication. Theoretically, there is no conflict between the M3-S3 communication and M1-S1 communication. However, the appearance of the M3-S3 communication not only degrades throughput and latency of the M2-S2 communication but also affect these of M1-S1 communication in the case of the generic router as shown in TABLE II. The reason for this situation was explained in Section II. This problem is overcome in the reconfigurable router by two operations as follows. Firstly, the buffer size of the VC assigned

TABLE II. EVALUATION IN TERMS OF THROUGHPUT AND LATENCY.

Scenario	Average throughput (flits/cycle)						Average latency (cycles)					
	Generic router			**Reconfigrable router**			Generic router			**Reconfigrable router**		
	S1	S2	S3	S1	S2	S3	S1	S2	S3	S1	S2	S3
1	0,139	-	-	0,139	-	-	558	-	-	558	-	-
2	0,081	0,096	-	0,081	0,096	-	610,0	930,1	-	610,0	930,1	-
3	**0,070**	0,071	0,139	**0,080**	0,071	0,139	**645,9**	966,0	496,2	**610,5**	966,0	496,2

to the M2-S2 communication is reconfigured to release the resources occupied by the M2-S2 communication. Secondly, the priority of the M1-S1 communication is promoted to equal to that of the M2-S2 communication. After these, the M1-S1 communication is performed in parallel with the M3-S3 communication. In consequence, the throughput and latency of the M1-S1 communication has been improved about 13.8% and 5.5%, respectively, for the case where the number of packets P3 is set to a quarter of the number of packets P1. The improvement in performance depends on the number of packets in a message of M3-S3 and M2-M2. The bigger the number of packets P3 is, the higher the improvement in performance is.

V. CONCLUSION

The paper presented our proposal, implementation and evaluation of a hybrid switching router with the priority-driven arbitration for the reconfigurable NoCs. In the proposed solution, router's resource is effectively exploited by dynamically switching scheme in order to improve the network total performance. Experimental results show that our proposal is reliable and can enhance significantly the average performance of BE load compared with generic router when the congestion happening. In terms of implementation cost, our proposed router consumes an insignificant ratio of the XC7VX485 FPGA (Xilinx Virtex-7) chip's resources. The router is feasible to apply for high-flexibility and high-performance on-chip embedded systems.

ACKNOWLEDGMENT

This work has been supported by Vietnam National University, Hanoi under Project No. QG.16.33.

REFERENCES

[1] Jimson Mathew, Rishad A. Shafik, Dhiraj K. Pradhan: "*Energy-Efficient Fault-Tolerant Systems*", Springer, 2014, pp211-240.

[2] Wen-Chung Tsai, Ying-Cherng Lan, Yu-Hen Hu, and Sao-Jie Chen: "*Networks on Chips: Structure and Design Methodologies*", Journal of Electrical and Computer Engineering, 2012, doi:10.1155/2012/509465.

[3] N. E. Jerger, L. S. Peh, "On-Chip Networks", Morgan and Claypool, 2009.

[4] M. Duranton et al., "*The HiPEAC Vision," HiPEAC Roadmap"*, 2015. [Online]. Available: www.hipeac.net/system/files/hipeacvision.pdf.

[5] M. Janidarmian, A. R. Fekr, V. S. Bokharaei, "*Application-Specific Networks-on-Chips Design*", IAENG International Journal of Computer Science, 38:1, pp16-25, 2011.

[6] F. K. Koupaei, A. Khademzadeh, and M. Janidarmian, "*Fault-Tolerant Application-Specific Network-on-Chip*", Proceedings of the World Congress on Engineering and Computer Science 2011, Vol II, WCECS 2011, October 19-21, 2011, San Francisco, USA.

[7] M. B. Stensgaard, and J. Spars, "*ReNoC: A Network-on-Chip Architecture with Reconfigurable Topology*", The second ACM/IEEE International Symposium on Networks-on-Chip, 2008.

[8] M.A. Al Faruque, T. Ebi, J. Henkel, "*Configurable Links for Runtime Adaptive On-chip Communication*", Design, Automation & Test in Europe Conference & Exhibition, 2009. DATE '09.

[9] C.A. Nicopoulos, D. K. Park, J.M Kim, N. Vijaykrishnan, M.S. Yousif, C.R. Das, "*ViChaR: A Dynamic Virtual Channel Regulator for Network-on-Chip Routers*", The 39th Annual IEEE/ACM International Symposium on Microarchitecture, 2008. MICRO-39.

[10] A. F. Beldachi, M. Hosseinabady, J. L. Nunez-Yanez, "*Configurable Router Design for Dynamically Reconfigurable Systems based on the SoCWire NoC*", International Journal of Reconfigurable and Embedded Systems (IJRES), Vol. 2, No. 1, March 2013, pp. 27~48, ISSN: 2089-4864.

[11] G. Kumaran, S. Gokila, "*Dynamic Router Design for Reliable Communication*", In Proceedings of 2014 International Conference On Global Innovations In Computing Technology (ICGICT'14).

[12] I. Pratomo and S. Pillement, "*Impact of design parameters on performance of adaptive network-on-chips*", International Conference on High Performance Computing and Simulation (HPCS), pp 724-725, July 2012.

[13] Thanh-Vu Le-Van, Xuan-Tu Tran, *"Simulation and Performance Evaluation of a Network-on-Chip Architecture based on SystemC"*, The 5th International Conference on Advanced Technologies for Communications (ATC), pp. 170-175, Hanoi, October 2012.

[14] S. T. Liu, A. Jantsch, Z. H Lu,: "*Analysis and Evaluation of Circuit Switched NoC and Packet Switched NoC*", The 2013 Euromicro Conference on Digital System Design (DSD), pp21-28.

[15] Modarressi, M, Sarbazi-Azad, H. ; Arjomand, M.: *A hybrid packet-circuit switched on-chip network based on SDM*, Conference & Exhibition on Design, Automation & Test in Europe, 2009 (DATE '09).

[16] G . Chen, M.A. Anders, et al.: "*A 340 mV-to-0.9 V 20.2 Tbs Source-Synchronous Hybrid Packet Circuit-Switched 16×16 Network-on-Chip in 22 nm Tri-Gate CMOS*", IEEE Journal of Solid-State Circuits (Issue: 1), pp. 59-67, 2015.

[17] Goossens Kees, John Dielissen, and Andrei Radulescu: "*Æthereal network on chip: concepts, architectures, and implementations*", IEEE Design & Test of Computers 22.5 (2005): 414-421.

[18] Vellanki, Praveen, Nilanjan Banerjee, and Karam S. Chatha. "*Quality-of-service and error control techniques for mesh-based network-on-chip architectures*." INTEGRATION, the VLSI journal 38.3 (2005): 353-382.

[19] Kiem Hung Nguyen, Xuan Tu Tran (2016) *Design and Implementation of a Hybrid Switching Router for the Reconfigurable Network-on-Chip.* In: the 2016 International Conference Advanced Technologies for Communications (ATC), 12-14 October 2016, Hanoi, Vietnam.