Regular Article

High-Level Modeling and Simulation of a Novel Reconfigurable Network-on-Chip Router

Thanh-Vu Le-Van, Xuan-Tu Tran

SIS Laboratory, VNU University of Engineering and Technology, Cau Giay, Hanoi, Vietnam

Correspondence: Xuan-Tu Tran, tutx@vnu.edu.vn

Manuscript communication: received 28 July 2014, accepted 02 January 2015

Abstract– **In this paper, we present a novel router architecture for implementing a Reconfigurable Network-on-Chip (RNoC) at high-level design using SystemC. The RNoC is an adaptive NoC-based system-on-chip providing a dynamic reconfigurable communication mechanism. By adding a virtual port – named Routing Modification port – into the conventional router architecture, the network router is able to route communication data flexibly whenever the target routing path is blocked, by unwanted defects or intently by a software programme to meet the requirements of applications. The proposed architecture has been modeled in SystemC/C++, simulated and verified within a 2D mesh 5×5 network platform. In normal communication mode, the static XY routing algorithm is used while the West-First algorithm with a proposed prohibited router surrounding technique will be applied in reconfiguration mode. Experimental results are also reported to compare the performance of the network architecture in different operation modes as well as with the other works.**

Keywords– **Network-on-Chip, Reconfiguration, High-level modeling.**

1 Introduction

The conventional shared-bus design methodology could not meet the increasingly needs of the on-chip communication in System-on-Chips (SoCs) because of long-wire loads, resistances and shared bandwidth. The bus hierarchical architecture has become an alternative solution but still faces many constraints in on-chip interconnections due to shared-bus s limitations.

The Network-on-Chip (NoC) paradigm has been known as an emerging design methodology for billion-transistors SoCs thanks to many advantages: high communication throughput, flexibility and scalability, power management efficiency, etc. [1]. In NoC-based systems, hundreds of processing cores (i.e., Intellectual Properties (IPs)) have been integrated into a single system to meet the increasingly demand of applications. This leads to many challenges in system design. One of these challenges is how to make the system adaptive to the need of target applications at a specific time or adaptive to the faults appeared during the operation. It means that the system should be able to be reconfigured at run-time.

In this paper, we present a novel reconfigurable router architecture for 2D mesh NoCs implementation. The network router is able to route communication data flexibly thanks to a virtual port – named Routing Modification (RM) port. With this design, the network router is dynamically reconfigured whenever the target routing path is blocked, by unwanted defects or intently by a software programme to meet the requirements of applications. The proposed network router is then used to build a 2D mesh 5×5 Reconfigurable Network-on-Chip (RNoC) platform for validation purpose. All the platform has been modeled, simulated and verified at high level using SystemC, a C/C++ library for hardware modeling. The static XY routing algorithm has been used in the normal communication mode while the West-First algorithm with a proposed prohibited router surrounding technique has been applied in the reconfiguration mode.

The remaining part of the paper is organized as follows. Section 2 provides a brief review on the related works. Section 3 introduces the proposed reconfiguration solution which allows modifying routing information to adapt real situations of the network. In Section 4, we present the proposed architecture for reconfigurable router, which is used to build the RNoC. Simulation and experimental results of a 2D mesh 5×5 RNoC are given in Section 5. Finally, conclusions will be provided in Section 6.

2 Related Works

There are three main concerns in implementing reconfigurable NoCs: reconfiguration administration, reconfiguration infrastructure, and reconfiguration protocols [2]. The first issue defines the administration methods including decision, validation, and execution. The second and the third ones relate to the changes on the network structure and network protocol, respectively.

As the communication is decoupled from the computation in NoC-based systems, the design of network infrastructure and protocol should be considered to increase the systems flexibility and reconfigurability. In [3], a reconfigurable NoC infrastructure was developed to make the system interconnection more flexible. This work aims at providing a reconfigurable interconnection architecture for FPGA implementation. The work presented in [4] introduced a reconfigurable mechanism for NoC architectures to provide fault tolerance. The uLBDR (Universal Logic-Based Distributed Routing) is proposed as an efficient logic-based mechanism to adapt to irregular topologies for 2D mesh networks. The main advantage of this proposal is the flexibility in routing communication information. However, it is obvious that we need more hardware resources for implementing the network routers. The work, presented in [5], proposed a hybrid communication reconfigurable NoC architecture. By using special wrappers surrounded the network routers, the network topology can be modified to adapt the requirements of applications. In [6], Stuart *et al.* proposed ReNoC router architecture with a special circuit switching wrapper which is used to change the network topology. ReNoC architecture uses an optimization algorithm with the initial topology of the network to match the target application. Therefore, this work must use application-specific routing and needs an initial phase to configure the topology before the system operates. In [7], ViChaR architecture was proposed to flexibly store data flits into buffers at INPORT modules. However, the area overhead is a drawback of this proposal. In another work [8], Lan *et al.* proposed a router architecture allowing each communication channel to be dynamically self-reconfigured to transmit data in either direction in order to better utilize on-chip hardware resources (therefore, enhance the performance of on-chip communication). The disadvantage of this proposal is that it leads to the complexity of routing algorithms. In all of the above proposals, the reconfiguration administration has been distributed at the network routers.

Related to the routing algorithm of reconfigurable NoCs, in [3], Bobda *et al.* proposed a Surrounding XY routing algorithm to surround obstacle components. The DyNoC router has three operation modes: N-XY (Normal XY routing), SH-XY (Surround Horizontal XY routing) and SV-XY (Surround Vertical routing). By combining deterministic and adaptive routing, Hu and Marculescu [9] presented DyAD which is smart routing to adapt network status. Manevich *et al.* [10] used a centralized routing control module to change XY routing or YX routing while the work in [4] introduced a Logic Based Distributed Routing (LBDR) algorithm, LBDR$_{fork+dr}$. This is an efficient routing to adapt to irregular topologies of 2D mesh networks.

3 Proposed Reconfiguration Solution and Routing Algorithms

Many NoC architectures have been developed by research groups, at both university and industry sectors. However, the most dominant topologies used in those NoC architectures are 2D mesh or 2D torus because of their semiconductor implementation. In our work, we focus on 2D mesh NoC architectures with source, deterministic routing. The target NoC architecture has been presented in [11].

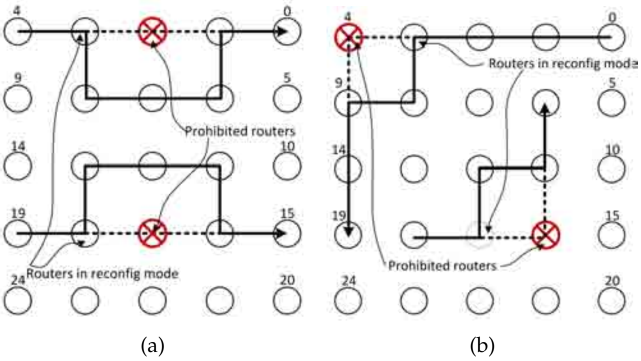

Figure 1. Update the routing path when the prohibited router appears: at the middle of a straight segment of the routing path (a) or at the corner of the routing path (b).

In 2D mesh NoC architectures, the static XY routing algorithms route communication data following straight routing segments and routing corner [12]. A routing corner appears when the routing path changes the direction from X to Y. When there is a prohibited router on the routing path, the communication data has to be routed around the prohibited router. It means that the related routers must be reconfigured to change the routing path to avoid the prohibited router. In this situation, depending on the position of the prohibited router and the destination router, the routing path should be modified in different strategies in order to ensure that the communication data will reach to the destination router with a minimum cost. By exploring 2D mesh NoC architectures, we can divide the reconfiguration strategy into 3 cases:

Case 1: The prohibited router appears at the middle of a straight segment of the routing path (see Figure 1(a)). In this case, the routing path has to be changed at the router before the prohibited router to avoid the prohibited router. The main principle is the communication data is routed to the left or the right routers instead of the prohibited router (the West-First routing algorithm is preferred in our experiment). Then the communication data will be forwarded by two hops with the same direction as the old routing path (normal routing path) before it is given back to the old routing path. In this case, we need two more hops in the new routing path (one to avoid the old routing path and one to come back to the old routing path). The router before the prohibited router must be reconfigured to add and update the routing information for 5 related hops (one hop needs to be updated, two hope need to be added, and the middle one and the last one can be kept). The old routing path is depicted as dot line and the new routing path (reconfigured routing path) is depicted as continuous line.

Case 2: The prohibited router appears at the corner of the routing path. The routing path is also changed to avoid the prohibited router as depicted in Figure 1(b). The dot line describes the old routing path and the continuous line describes the new (reconfigured) routing path. However, in this case there are only three routers affected by the reconfiguration to establish the new routing path, even if the routing corner appears at the

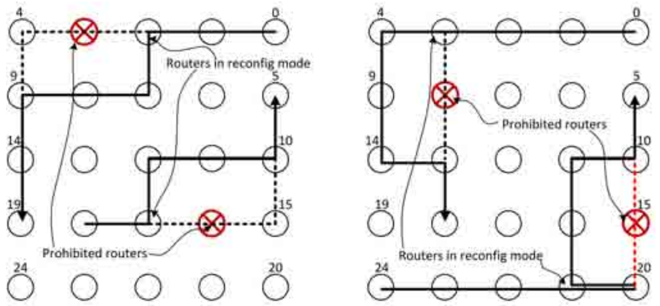

Figure 2. Update the routing path when the prohibited router appears just before or just after the corner of the routing path.

corner of network architecture. The router placed before the prohibited router must be reconfigured to change the routing information for three hops in corresponding to three related routers.

Case 3: The prohibited router appears just before or just after the corner of the routing path. Figure 2(a) illustrates the case that the prohibited router appears before the corner of the routing path and Figure 2(b) illustrates the case that the prohibited router appears after the corner of the routing path. Similar to the two cases above, the routing path is also changed to avoid the prohibited router. However, in this case there are four routers affected by the reconfiguration to establish the new routing path. The West-First routing algorithm is also preferred in changing the routing direction. The router before the prohibited router has to be reconfigured to change the routing information for four hops (if the prohibited router appears before the corner of the routing path) or five hops (if the prohibited router appears after the corner of the routing path). In the case 3a, one hop needs to be updated, one hop needs to be added, and two last ones can be kept). In the case 3b, one hop needs to be updated, two hops need to be added, the middle one and the last one can be kept.

There is a special case, when the destination router is prohibited (the destination router becomes the prohibited router). The communication data cannot reach to its destination and therefore should be deleted to increase network load. This is also considered in our proposal.

With deterministic, source routing NoC architectures, the routing information is stored in “Path-to-Target” field of the header flit [11]. Therefore, to change the routing path the network router should be able to update/modify the routing information in “Path-to-Target” field. That s why we have proposed a novel router architecture for reconfigurable NoCs as presented in the next section.

4 Reconfigurable Router Architecture

In this work, we propose a novel architecture for the network router used in reconfigurable Network-on-Chips, depicted in Figure 3. The proposed reconfigurable network router is composed of five INPORT modules, five OUTPUT modules, and a virtual Routing Modification (RM) port. Four INPORT/OUTPORT pairs are connected to four neighbour routers and the remaining pair (local INPORT/OUTPORT) is connected to the nearest IP core. The virtual RM port is used for reconfiguration purpose. These modules are connected together through two signals matrix for two virtual channels. The local INPORT has a signal vector to indicate the status of OUTPORT modules; and the local OUTPORT has four flag signals to inform the status of INPORT modules.

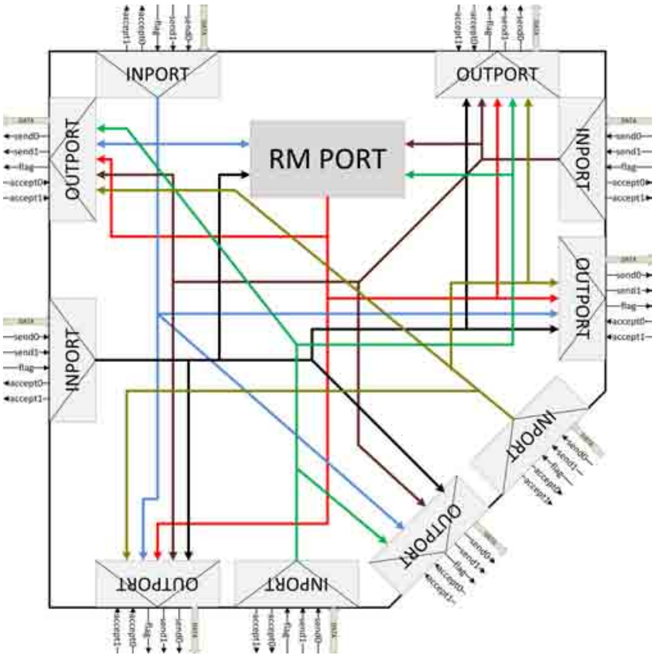

Figure 3. The proposed reconfigurable network router.

There are two operation modes of the router: normal mode and reconfig mode. In normal mode, a data packet is received at INPORT and will be sent to next router through the target OUTPORT for all flits in the packet. In reconfig mode, INPORT received a header flit, but it cannot be sent to selected OUTPORT because the selected OUTPORT is blocked. Therefore, the INPORT changes its operation mode, then sends header flit to the RM port and waits for the response from the RM port. At the RM port, the routing information (“Part-to-Target” field) of the header flit will be changed so that the header flit is sent to a new OUTPORT; the RM port sends a command back to the INPORT to control the sending of the body flits of the packet (these flits will be routed to the new OUTPORT instead of the previously selected OUTPORT).

The detail of INPORT structure is shown in Figure 4. This work supports communication with two virtual channels (0 and 1), so that INPORT has sub-blocks corresponding virtual channels: RouteVC0 & RouteVC1, Sendacc0 & Sendacc1. The VC_Demux sub-block is used to receive flit and give routing information at “Path-to-Target” field in normal mode. When the selected OUTPORT is blocked, the Prohibited Control sub-block detects and sets the mode of INPORT into reconfig mode. When a cell is prohibited, this sub-block receives status flag from LOCAL port and then it prohibits the OUTPORT module which is connected to it.

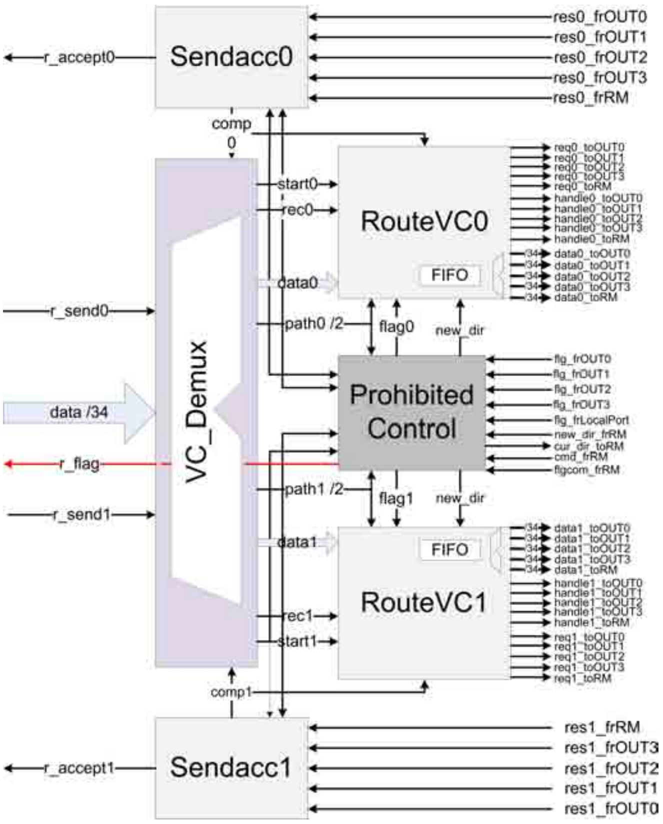

Figure 4. Micro-architecture of INPORT modules.

OUTPORT module has eight sub-blocks and supports two virtual channels as described in Figure 5. There are three pair sub-blocks supporting two virtual channels, such as: Arbiter0 & Arbiter1, Forward0 & Forward1, and MuxVC0 & MuxVC1.

The VC-Mux sub-block is used to combine two virtual channels and to send data to next router. At the Arbiter sub-blocks, we use First In-First Served (FIFS) mechanism when there are more than one request from INPORT modules. The OUTPORT goes into reconfig mode when it receives a request from RM port, and then the Prohibited Control sub-block actives flag to control other sub-blocks in reconfig mode. At reconfig mode, the OUTPORT receives header flit from RM port and command to forward other flits of data packet. If RM port only sends header flit, the response of OUTPORT is changed to INPORT which is indicated from RM port.

The architecture of the virtual RM port is shown in Figure 6. In this virtual port, two sub-blocks, Receiver0 and Receiver1, are used to receive flits from INPORT modules and two sub-blocks, Sender0 & Sender1, send flits to OUTPORT modules for virtual channels 0 & 1, respectively. The Controller sub-block is used to control the sub-blocks in RM port and receives request from INPORT modules. The process which is used to change routing information in "Path-to-Target" field and selected new OUTPORT is implemented in the Update sub-block. In the Update sub-block, we use small sub-block to check the status of OUTPORT modules in router. The checking results are used to select a possible OUTPORT to forward data into previous router in the new routing path. In this work, the RM port is equivalent to a virtual port, it only use one buffer for two virtual channels.

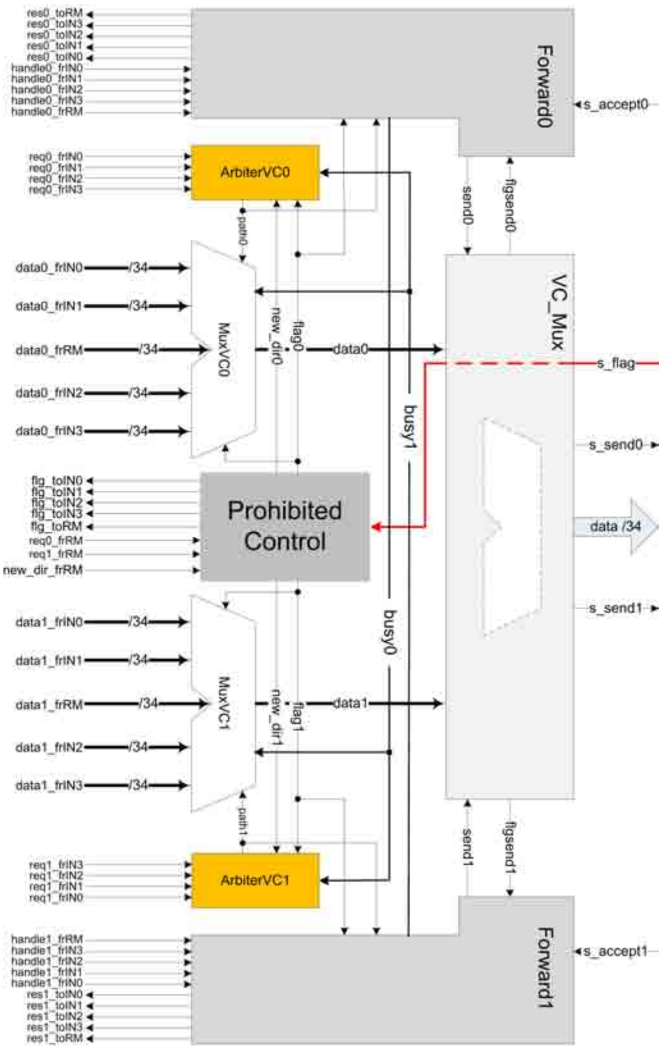

Figure 5. Micro-architecture of OUTPORT modules.

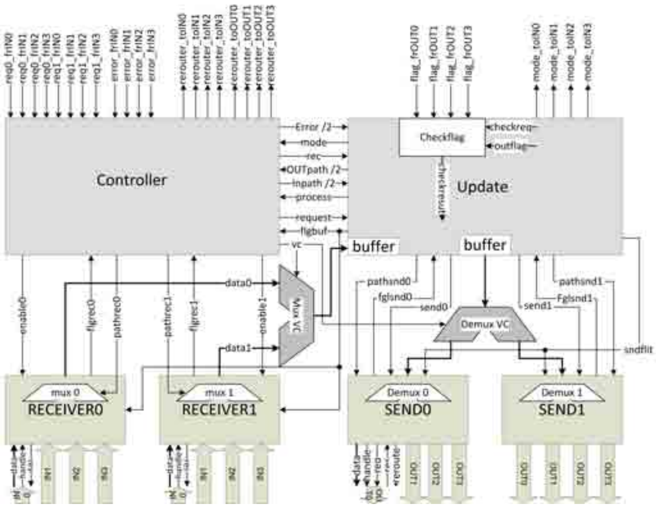

Figure 6. Architecture of the virtual Routing Modification (RM) port.

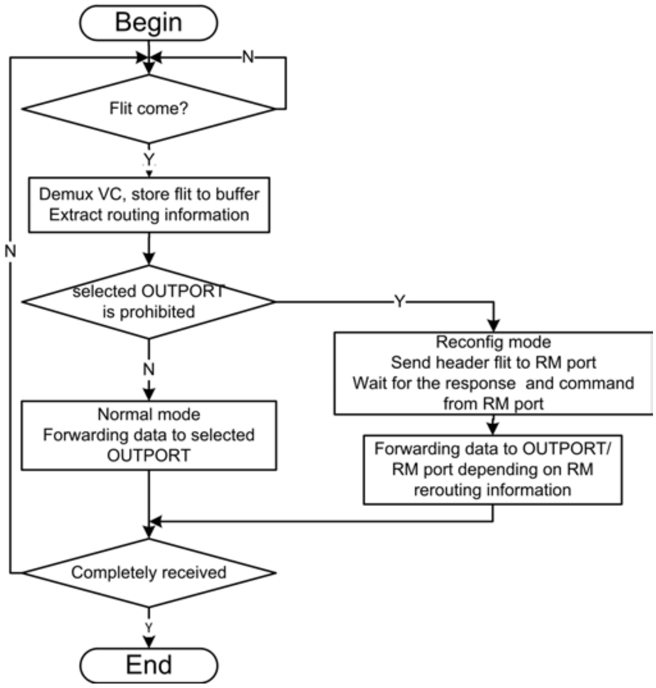
Begin
N
Flit come?
Y
Demux VC, store flit to buffer
Extract routing information
N
selected OUTPORT is prohibited
Y
N
Reconfig mode
Send header flit to RM port
Wait for the response and command from RM port
Normal mode
Forwarding data to selected OUTPORT
Forwarding data to OUTPORT/ RM port depending on RM rerouting information
Completely received
Y
End

Figure 7. Operation flowchart of INPORT.

5 Simulation and Experimental Results

The simulated platform was configured in 2D mesh topology with 5×5 network size. We use complement communication pattern and source static XY routing algorithm to generate data packet and inject into network from stimulating IP cores [13]. When RNoC is simulated, we control it to change prohibited cell (obtain IP core and router) and network load is injected into network. After simulating RNoC, the evaluation is executed to compute network latency and average throughput.

The operation of INPORT modules is shown in Figure 7 and the flowchart describes operation of OUTPORT in Figure 8.

The reconfig mode is started at INPORT and is processed at RM port in the router. When the corresponding OUTPORT is blocked, the Prohibited Controller sub-block release an active mode flag and controls other sub-block in the reconfig mode. The header flit is forwarded to RM port, after INPORT waits until the RM port completely processes the header flit. If mode_frRM signal is not active, INPORT will forward other flits of packet to a new OUTPORT that is selected from RM port.

At OUTPORT module, the reconfig mode is started when it is requested from RM port, the Prohibited Controller controls other sub-block of OUTPORT module. If mode_frRM is not active, the OUTPORT interrupts response for RM port and then the RM port informs the INPORT to send other body flit of packet.

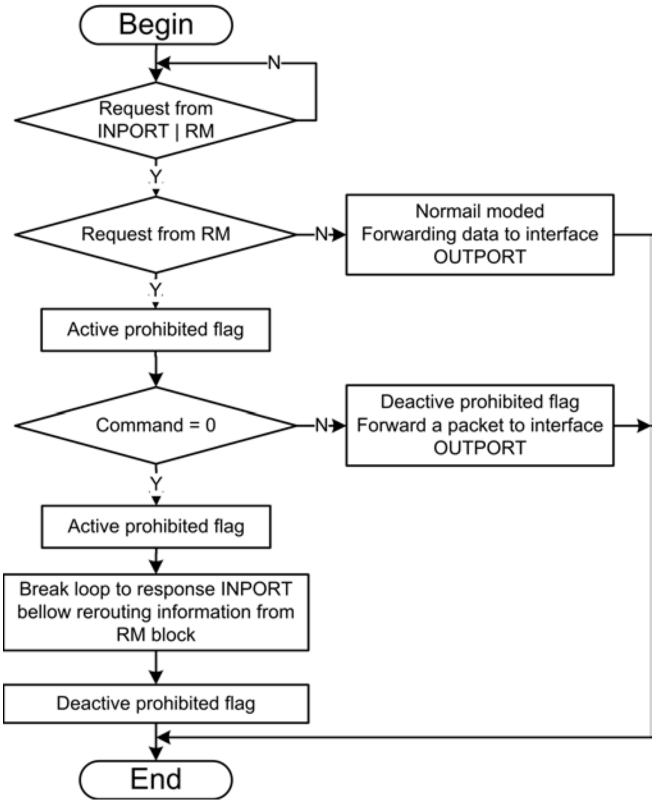

Figure 8. Operation flowchart of OUTPORT.

The RM port is active when there are any requests from INPORT modules. The RM port receives data flit similar OUTPORT mechanism and stores it into receiver buffer. Base on routing the current information routing in “Path-to-Target” field, the ID of INPORT, and the ID of the error OUTPORT, the Update sub-block decides a new corresponding OUTPORT and modifies routing information. After completing the process for header flit, RM port sends the header flit to OUTPORT and a command to INPORT to ask the INPORT to go into reconfig mode. In the normal mode, the RM port is not used; the data packets will be transferred from INPORT through OUTPORT modules.

To show the difference between normal mode and reconfig mode of the proposed RNoC, we compare the communication performance in terms of latency and throughput of these modes with different positions of prohibited router. Figure 9 shows the relationship between the network latency and the network load with normal mode (*Normal*) and different reconfig modes (*Corner*, *BorderX*, *BorderY*, and *Inner*). In the figure, the latency of normal mode is lowest and the latency of reconfig mode in which the *Inner* router is prohibited (see Figure 1(a) for router location) is highest. In fact, when the *Inner* router is prohibited, four directions will be blocked. As a result, many packets are blocked in routing paths; thus there are many routing paths which are reconfigured. When the prohibited router is placed at the *Corner* or the *Border*, the network latency is less than in the *Inner* cases.

The communication throughput in corresponding with the network load is presented in Figure 10. We can see that when the network load is small enough, the throughput in reconfig mode is similar to the throughput of normal mode. When a prohibited router is located at the network boundary, the throughput is slightly reduced. In the worst case, the throughput sat-

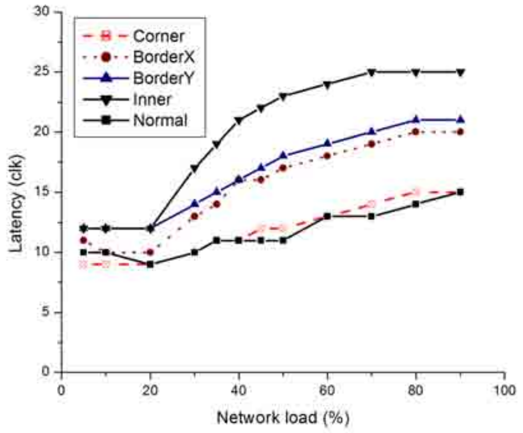

Figure 9. The network latency in corresponding with the network load.

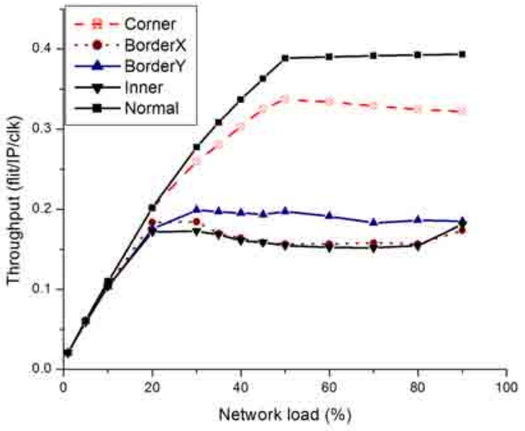

Figure 10. The network throughput in corresponding with the network load.

urates at 0.16 *flit/IPcore/clk* when network load is equal to 30%.

The obtained simulation results have been also compared with the previous works as depicted in Figure 11. From this comparison, we can see that the ViChar proposal [7] has a better communication throughput. However, the implementation of this solution is more expensive than our proposal in terms of area overhead, each router needs 40 or 80 flits buffering for ViChar-8 or ViChar-16, respectively. In addition, they use 4 virtual channels for each physical communication channel. The Reduced BiNoC [8] has a lowest communication throughput even the total buffer size is the biggest. Our proposal has a high throughput when the prohibited router is placed at the corner of the network. In the worst case, the prohibited router is an inner router, the achieved throughput is nearly the same as ViChar-8/ViChar-16 when the network load is less than 20% and acceptable when the network load is greater than 20%. The details of each architecture has been reported in Table I.

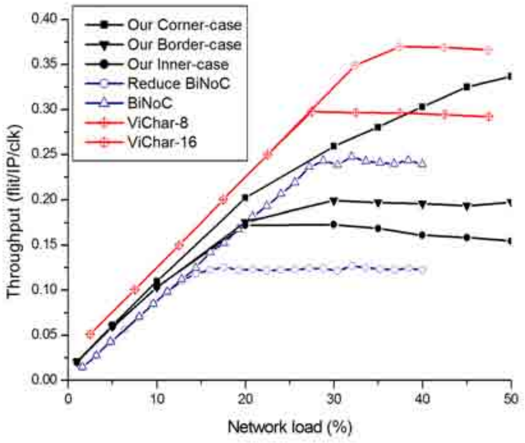

Figure 11. Throughput comparison with the previous works.

Table I
Communication Resource for Each Router (RNoC, ViChar, and BiNoC)

Architectures	**our RNoC**	BiNoC [8]	Re. BiNoC [8]	ViChar-8 [7]	ViChar-16 [7]
Total # buffers	6	10	5	5	5
Buffer/Direction	1	2	1	1	1
Each buffer size	2 *flits*	16 *flits*	32 *flits*	8 *flits*	16 *flits*
Total buffer size	12 *flits*	160 *flits*	160 *flits*	40 *flits*	80 *flits*
Crossbar	2(6×6)	10×10	5×5	5×5	5×5
Ports	5+1	10	5	5	5
Virtual channels	2	2	1	4	4

6 Conclusions

We have presented the routing method and the design of a reconfigurable router which can be used for implementing a reconfigurable Network-on-Chip (RNoC). The design has been modeled at high-level using SystemC. By adding a virtual Routing Modification port (RM port) into the router architecture, the network router is able to route communication data flexibly whenever the target routing path is blocked to meet the requirements of reconfiguration or to adapt the working situation caused by unwanted defects. The proposed architecture has been simulated and verified within 2D mesh 5×5 network platform. In this verification platform, the static XY routing algorithm has been used in the normal communication mode while the West-First algorithm and a proposed prohibited router surrounding technique have been applied in the reconfig mode. The experimental results show that our proposed solution provide a better communication performance in terms of throughput and latency with a smaller hardware resource.

Acknowledgment

This work is supported by the research project "Reconfiguration Solution in Designing Network-on-Chip Architectures (ReSoNoC)" (No. 102.01-2013.17) granted by Vietnam National Foundation for Science and Tech-

nology Development (Nafosted).

References

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

[2] R. Dafali, J.-P. Diguet, and M. Sevaux, “Key research issues for reconfigurable network-on-chip,” in *Proceedings of the 2008 International Conference on Reconfigurable Computing and FPGAs (ReConFig)*, Cancun, December 2008, pp. 181–186.

[3] C. Bobda, A. Ahmadinia, M. Majer, J. Teich, S. Fekete, and J. van der Veen, “DyNoC: A dynamic infrastructure for communication in dynamically reconfigurable devices,” in *Proceedings of the International Conference on Field Programmable Logic and Applications*, August 2005, pp. 153–158.

[4] S. Rodrigo, J. Flich, , A. Roca, S. Medardoni, D. Bertozzi, J. Camacho, F. Silla, and J. Duato, “Addressing manufacturing challenges with cost-efficient fault tolerant routing,” in *Proceedings of the 4th ACM/IEEE International Symposium on Networks-on-Chip*, Grenoble, France, May 2010, pp. 25–32.

[5] L. Zheng, C. Jueping, D. Ming, Y. Lei, and L. Zan, “Hybrid communication reconfigurable network on chip for MPSoC,” in *Proceedings of the 2010 IEEE International Conference on Advanced Information Networking and Applications (AINA)*, Perth, WA, April 2010, pp. 356–361.

[6] M. B. Stuart, M. B. Stensgaard, and J. Sparso, “The ReNoC reconfigurable network-on-chip: Architecture, configuration algorithms, and evaluation,” in *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 10, no. 4, New York, NY, USA, November 2011, pp. 45:1–45:26.

[7] C. A. Nicopoulos, D. Park, J. Kim, M. S. Y. N. Vijaykrishnan, and C. R. Das, “ViChaR: A dynamic virtual channel regulator for network-on-chip routers,” in *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Orlando, FL, December 2006, pp. 333–346.

[8] Y.-C. Lan, H.-A. Lin, S.-H. Lo, Y. H. Hu, and S.-J. Chen, “A bidirectional NoC (BiNoC) architecture with dynamic self-reconfigurable channel,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 3, pp. 427–440, March 2011.

[9] J. Hu and R. Marculescu, “DyAD - smart routing for networks-on-chip,” in *Proceedings of the 41st IEEE Design Automation Conference (DAC)*, San Diego, CA, USA, July 2004, pp. 260–263.

[10] R. Manevich, I. Cidon, A. Kolodny, and I. Walter, “Centralized adaptive routing for NoCs,” in *IEEE Computer Architecture Letters*, vol. 9, no. 2, 2010, pp. 57–60.

[11] N.-K. Dang, T.-V. Le-Van, and X.-T. Tran, “FPGA implementation of a low latency and high throughput network-on-chip router architecture,” in *Proceedings of the 2011 International Conference on Integrated Circuits and Devices in Vietnam (ICDV)*, Hanoi, Vietnam, August 2011, pp. 112–116.

[12] C. J. Glass and L. M. Ni, “The turn model for adaptive routing,” *Journal of the Association for Computing Machinery*, vol. 41, no. 5, pp. 874–902, September 1994. [Online]. Available: http://doi.acm.org/10.1145/185675.185682

[13] T.-V. Le-Van, X.-T. Tran, and D.-T. Ngo, “Simulation and performance evaluation of a network-on-chip architecture based on SystemC,” in *Proceedings of the 2012 International Conference on Advanced Technologies for Communications (ATC)*, Hanoi, Vietnam, October 2012, pp. 170–175.

Thanh-Vu Le-Van was born in Hue, Vietnam. He received a Bachelor of Science degree in Physics from Hue University of Sciences – a member university of Hue University and a Master of Science degree from Vietnam National University Hanoi (VNU) in Electronics Engineering and Communications.

Currently, he is pursuing his Ph.D degree at VNU University of Engineering and Technology (VNU-UET), a member university of Vietnam National University, Hanoi (VNU). He joined with the Key Laboratory for Smart Integrated Systems (SIS Lab) of VNU-UET in 2010. His research interests include Network-on-Chip (NoC), High-level modeling methods, Reconfigurable architecture.

He is a member of the IEEE (SSCS) and the Radio Electronics Association of Vietnam (REV).

Xuan-Tu Tran was born in Nghe An, Vietnam. He received a B.Sc. degree in 1999 from Hanoi University of Science and a M.Sc. degree in 2003 from Vietnam National University, Hanoi, all in Electronics Engineering and Communications; and a Ph.D. degree in 2008 from Grenoble INP (in collaboration with the CEA-LETI), France, in Micro Nano Electronics.

Xuan-Tu Tran is currently an associate professor at the Faculty of Electronics and Telecommunications, VNU University of Engineering and Technology (VNU-UET), a member university of Vietnam National University, Hanoi (VNU). He has worked as a lecturer at Vietnam National University, Hanoi (1999-2003), as a research engineer at the CEA-LETI, MINATEC, France (2003-2008). He was an invited professor at the University Paris-Sud 11, France (2009, 2010), visiting professor at Grenoble INP in 2011. He is currently Deputy Director of UET-key Laboratory for Smart Integrated Systems (SIS) and Head of VLSI Systems Design laboratory. He is in charge for CoMoSy, VENGME, ReSoNoC projects for embedded systems and multimedia applications. His research interests include design and test of systems-on-chips, networks-on-chips, design-for-testability, asynchronous/synchronous VLSI design, low power techniques, and hardware architectures for multimedia applications.

He is a Senior Member of the IEEE, IEEE Circuits and Systems (CAS), IEEE Solid-State Circuits and Systems (SSCS), member of IEICE, and the Executive Board of the Radio Electronics Association of Vietnam (REV). He serves as Vice-Chairman of IEICE Vienam Section, Chairman of IEEE SSCS Vietnam Chapter. He also served as chair/co-chair and technical/organizing committee member for numerous international conferences (ATC, ICDV, IEEE DELTA, IEEE GIIS, ICCE, IEEE RIVF, IEEE APCCAS, IEEE ICCD, IEEE ICCAS, HP3C, NICS, APCC, ACOMP, CommandTel, ICGHIT...), Associate Editor-in-Chief of REV Journal on Electronics and Communications (JEC), Editor of VNU Journal of Computer Science and Communication Engineering (JCSCE), International Journal of Computing and Digital Systems (IJCDS).