



ROUTERS IN ON CHIP NETWORK USING ROSHAQ FOR EFFECTIVE TRANSMISSION

Sivasankari.S#1, Selvaganesan.M #2

#1PGStudent,#2AssistantProfessor

Department of Electronics and Communication Engineering, Vandayar Engineering College, Thanjavur, Tamil Nadu, India. 91sivasankari@gmail.com,selvaembedded@gmail.com

Abstract

Router micro architecture plays a central role in the performance of networks-on-chip (NoCs). On-chip routers typically have buffers dedicated to their input or the outputs of a router, corresponding to an input-buffered chip routers typically have buffers dedicated to their input or output ports for temporarily storing packets in case contention occurs on output physical channels. This buffering can be at the inputs router (IBR) or an output-buffered router (OBR). A large number of buffer queues in the network are empty and other queues are mostly busy. In this paper, a new router design based on router architecture with shared queues (RoShaQ) is proposed. We introduce innovations to address the unique constraints of NoCs, includes efficient pipelining and novel flow control. Our micro architecture design can achieve significantly higher bandwidth at saturation, with an improvement of up to 20% when compared to a VC router with the same amount of buffering, and our proposed architecture can achieve up to 94% of the ideal saturation throughput.

Keywords: Adaptive routing, network on chip, router micro architecture, throughput analysis

I.INTRODUCTION

NETWORKS-ON-CHIP (NoC) is an emerging paradigm for communication within large VLSI systems implemented on a single silicon chip that is called the layered-stack approach to the design of the on-chip intercore communications. NoC architectures are becoming the de facto fabric for both general-purpose chip multi-processors and application-specific systems-on-chip designs.

In the design of NoCs, high throughput and low latency are both important design parameters and the router micro architecture plays a vital role in achieving these goals. One of the most important problems for NoC that needs to be solved is

its power consumption and throughput. In particular, power consumption and throughput of a router accounts for a large percentage. High throughput routers allow a NoC to satisfy the communication needs of multi- and many-core applications, or the higher achieve throughput can be traded off for power savings by using fewer resources to attain a target bandwidth. Ultimately, a router's role lies in the efficient multiplexing of packets onto network links. In a NoC system, modules such as processor cores, memory and specialized IP blocks exchange data using a network as a public transportation sub-system for the information traffic. A

NoC is similar to a modern telecommunications network, using digital bit-packets switching over multiplexed links.

Router buffering is used to house arriving flits that contention arises. This buffering can be at the inputs or the outputs of a router, corresponding to an input-buffered router (IBR) or an output-buffered router (OBR). Thus buffering is done in a router. Each router has five ports that connect to four neighboring routers and its local processor. A network interface (NI) locates between a processor and its router for transforming processor messages into packets to be transferred on the network and vice versa.

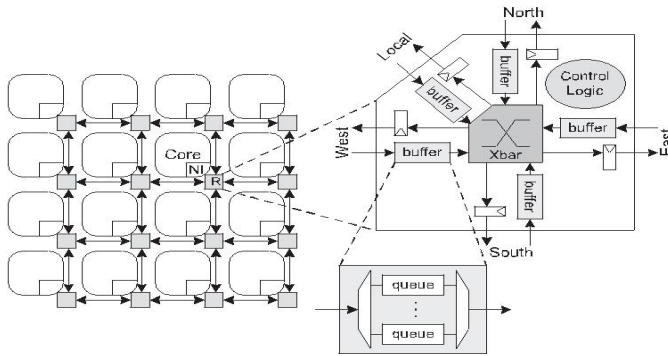


Fig. 1. Chip multiprocessors interconnected by a network of VC routers.
NI: network interface. R: router.

Buffers are referred to as middle memories (MMs). In a typical router, each input port has an input buffer for temporarily storing the packets in case that output channel is busy. This buffer can be a single queue as in a wormhole (WH) router or multiple queues in parallel as in virtual channel (VC) routers [5]. These buffers, in fact, consume significant portions of area and power that can be more than 60% of the whole router [6]. Buffer less routers remove buffers from the router hence save much area [7], [8]; however, their performance becomes poor in case packet injection rates are high. Because of having no buffers, previous router designs proposed to drop and retransmit packets or to deflect them once network contention occurs that can consume even higher energy per packet than a router with buffers [9].

Another approach is by sharing buffer queues that allows utilizing idle buffers [10] or emulating an output buffer router to obtain higher throughput [11]. Our work differs from those router designs by allowing input packets at input ports to bypass shared queues hence; it achieves lower zero-load latency. In addition, the proposed router architecture has simple control circuitry making it dissipate less packet energy than VC routers and achieving higher throughput by

letting queues share workloads when the network load becomes heavy.

The main contributions of this paper are:

- 1) Exploring and analyzing shared-queue router architectures that maximize buffer utilization for boosting net-work throughput.
- 2) Proposing a router architecture which allows input packets to bypass shared queues for reducing zero-load packet latency.

We propose a new router micro architecture that it is based on a distributed shared buffer (DSB) router architecture that has been successfully used in high-performance Internet packet routers [12], [13]. Rather than buffering data at the output ports, a DSB router uses two crossbar stages with buffering sandwiched in between. To emulate the First-Come-First-Serve (FCFS) order of an output-buffered router, incoming packets are time stamped with the same departure times as they would depart in an OBR. Incoming packets are assigned to one of the middle memory buffers with two constraints. First, incoming packets that are arrived at the same time must be assigned to different buffers. Second, an incoming packet cannot be assigned to a buffer that already holds a packet with the same departure time. It has been shown in [12], [13] that $M \geq (2P - 1)$ middle buffers are necessary and sufficient to ensure that memory assignments are always possible that would emulate a FCFS output-buffered router if unlimited buffering is available.

2. BACKGROUND

Router components:

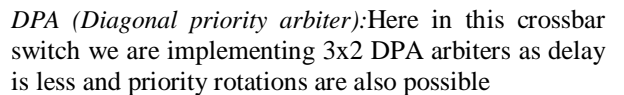
This router has the following components

1. Input buffers
 2. Route computation logic
 3. Virtual channel allocator
 4. Switch allocator
 5. Crossbar switch.
- By using the above following components the router pipeline is formed it is shown in Fig 2. and it is described as follows:

- 1) When a flit arrives at an input port, it is written to the corresponding buffer queue. This step is called buffer write or queue write (QW).
- 2) Assuming without other packets in the front of the queue, the packet starts deciding the output port for its next router (based on the destination information contained in its head flit) instead of for the current router (known as look ahead routing computation

DPA [14], [15]. In DPA request from the input ports arrive at the scheduler for passing to the destined output port.

- First request comes from the input ports to the crossbar scheduler of the switch for the destination output port. The scheduler grants a request based on a priority algorithm that ensures fair service to all the input ports. Once a grant is issued, the crossbar fabric is configured to map the granted. Input ports to their destination output ports.



B) SWITCH ALLOCATOR:

Individual flits arbitrate for access to physical channels via the crossbar on each cycle. Arbitration may be performed in two stages [16]. The first reflects the sharing of a single crossbar port by V input virtual-channels; this requires a V -input arbiter for each input port. The second stage must arbitrate between winning requests from each input port (P inputs) for each output channel. The scheme is illustrated in Fig. The request for a particular output port is routed from the VC which wins the first stage of arbitration. In order to improve fairness, the state of the V -input the second stage of arbitration. We assume this organization wherever multiple stages of arbitration are present. This switch allocator organization may reduce the number of requests for different output ports in the first stage of arbitration, resulting in some wasted switch bandwidth.

Fig. 4. Switch allocation in NoC router

Page 183

C) ARBITER:

Arbiter controls the arbitration of the ports and resolves contention problem. It keeps the updated status of all the ports and knows which ports are free and which ports are communicating with each other.[16]. Packets with the same priority and destined for the same output port are scheduled with a round-robin arbiter. Supposing in a given period of time, there was many input ports request the same output or resource, the arbiter is in charge of processing the priorities among many different request inputs. The arbiter will release the output port which is connected to the crossbar once the last packet has finished transmission. So that other waiting packets could use the output by the arbitration of arbiter.

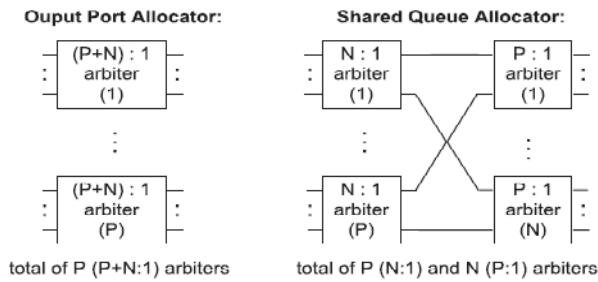


Fig 5. OPA and SQA structures in RoShaQ router

3. RoShaQ: ROUTER ARCHITECTURE WITH SHARED QUEUES:

In this design, a packet from an input queue simultaneously arbitrates for both shared queues and an output port; if it wins the output port, it would be forwarded to the downstream router at the next cycle. Otherwise, that means having congestion at the corresponding output port; it can be buffered to the shared queues. Intuitively, at low load, the network would have low latency because packets seem to frequently bypass shared queues. While at heavy load, shared queues are used to temporarily store packets hence reducing their stall times at input ports that would improve the network throughput.

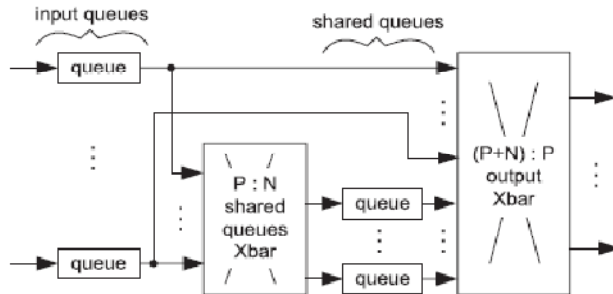


Fig 6. Allows input packets to bypass shared queues. P: the number of router ports. V: the number of VC queues per input port in a VC router N: the number of shared queues.

RoShaQ Architecture:

RoShaQ, router architecture with shared queues based on the idea of Fig. 6 is shown in Fig. 7. When an input port receives a packet, it calculates its output port for the next router (look ahead routing), at the same time it arbitrates for both its decided output port and shared queues. If it receives a grant from the output port allocators (OPAs), it will advance to its output port in the next cycle. Otherwise, if it receives a grant to a shared queue, it will be written to that shared queue at the next cycle. In case that it receives both grants, it will prioritize to advance to the output port.

Shared-queues allocator (SQA) receives requests from all input queues and grants the permission to their packets for accessing non full shared queues. Packets from input queues are allowed to write to a share queue only if: 1) the shared queue is empty or 2) the shared queue is containing packets having the same output port as the requesting packet. This shared queue writing policy guarantees deadlock-free for the network.

The OPA receives requests from both input queues and shared queues. Both SQA and OPA grant these requests in round-robin manner to guarantee fairness and also to avoid starvation and live lock. Input queue, output port, and shared-queue states maintain the status (idle, wait, or busy) of all queues and output ports, and incorporate with SQA and OPA to control the overall operation of the router. Only input queues of RoShaQ have routing computation logic because packets in the shared queues were written from input queues hence they already have their output port information. RoShaQ has the same I/O interface as a typical router that means they have the same number of I/O channels with flit-level flow control and credit-based backpressure management [21],[22].

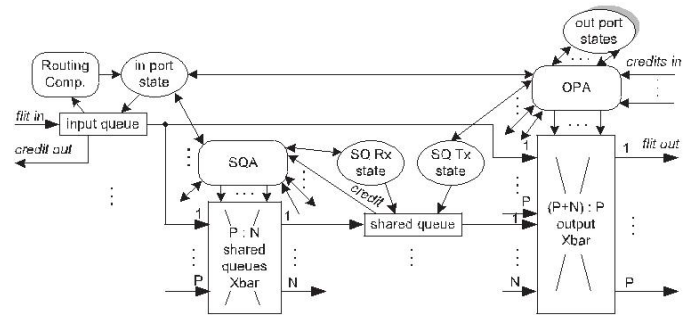


Fig. 7. RoShaQ router micro architecture. SQA: shared-queue allocator. OPA: output port allocator. SQ Rx state: shared queue receiving/writing state. SQ TX state: shared queue transmitting/reading state.

RoShaQ's Properties:

- 1) *A network of RoShaQ routers is deadlock-free.*
At light load, packets normally bypass shared queues, so RoShaQ acts as a WH router hence the network is deadlock-free [17]. At heavy load, if a packet cannot win the output port, it is allowed to write only to a shared queue which is empty or contains packets having the same output port. Clearly, in this case RoShaQ acts as an output-buffered router which is also shown to be deadlock-free [17].
- 2) *A network of RoShaQ routers is live lock-free.*
Because both OPA and SQA use round-robin arbiters, each packet always has a chance to advance to the next router closer to its destination; hence the network is also free from live lock.
- 3) *RoShaQ supports any adaptive routing algorithm.* The output port for each packet is only computed at its input queue, not at shared queues. Therefore, any adaptive routing algorithm which works for WH routers also works for RoShaQ.
- 4) *RoShaQ can be used for any network topology.* If we hide all design details inside RoShaQ, we would see RoShaQ only has one buffer queue at each input port similar to a WH router. Therefore, we can change the number of RoShaQ's I/O ports to make it compatible with any network topology known in the literature along with an appropriate routing algorithm.

4. ADAPTIVE ROUTING ALGORITHM:

NoC is flexible to dynamically support the communication among modules in a system with heavily varying workloads. To augment resource utilization and flexibility, the NoC needs to be integrated with novel adaptive methodologies employing resource multiplexing mechanisms at varying workloads. In the scope of this thesis, several novel adaptive schemes for the on-chip network architecture are presented to exploit all the benefits that can be obtained. Each component of on-chip network platform can be implemented adaptively to increase the utilization and performance. In order to route data packets to the non-congested area and/or give the priority to a packet passing through a congested area of the network to ease the congestion faster, respectively. Network interfaces can handle in-order delivery which is a practical approach when exploiting an adaptive routing algorithm for distributing packets through the network or when exploiting dynamic memory access scheduling in memory controller to reorder memory requests. Adaptive on-chip network interface architectures are utilized to increase the resource utilization and performance.

Adaptive routing proposes that each router be aware of the network's traffic situation and adapt its routing (wormhole

packet switching) accordingly. The issue is to avoid traffic congestions and be fault-tolerant towards both disabled nodes and connection.[18],[19].Therefore, certain algorithms permit misrouting which leads packets away from their intended destination to avoid high-traffic areas. In the case where traffic is low, an adaptive routing algorithm should seek to provide a minimal (ideally the shortest) path between source and destination. Implementations of adaptive routing can cause adverse effects if care is not taken in analyzing the behavior of the algorithm under different scenarios (concentrated traffic, non-uniform and uniform traffic).

*Experimental set up:**NS2 (network simulator):*

Network Simulator (Version 2), widely known as NS2, is simply an event driven simulation tool [20] that has proved useful in studying the dynamic nature of communication networks. Simulation of wired as well as wireless network functions and protocols (e.g., routing algorithms, TCP, UDP) can be done using NS2.

NS2 provides users with executable command `ns` which take on input argument, the name of a Tcl simulation scripting file. Users are feeding the name of a Tcl simulation script (which sets up a simulation) as an input argument of an NS2 executable command `ns`. In most cases, a simulation trace file is created, and is used to plot graph and/or to create animation.

KEY LANGUAGE:

NS2 consists of two key languages: C++ and Object-oriented Tool Command Language (OTcl). While the C++ defines the internal mechanism (i.e., a backend) of the simulation objects, the OTcl sets up simulation by assembling and configuring the objects as well as scheduling discrete events (i.e., a frontend). The C++ and the OTcl are linked together using TcLCL. Mapped to a C++ object, variables in the OTcl domains are sometimes referred to as *handles*. Conceptually, a handle (e.g., `n` as a Node handle) is just a string (e.g., `_o10`) in the OTcl domain, and does not contain any functionality. Instead, the functionality (e.g., receiving a packet) is defined in the mapped C++ object (e.g., of class Connector). In the OTcl domain, a handle acts as a front end which interacts with users and other OTcl objects. NS2 provides a large number of built-in C++ objects. It is advisable to use these C++ objects to set up a simulation using a Tcl simulation script.

After simulation, NS2 outputs either text-based or animation based simulation result.

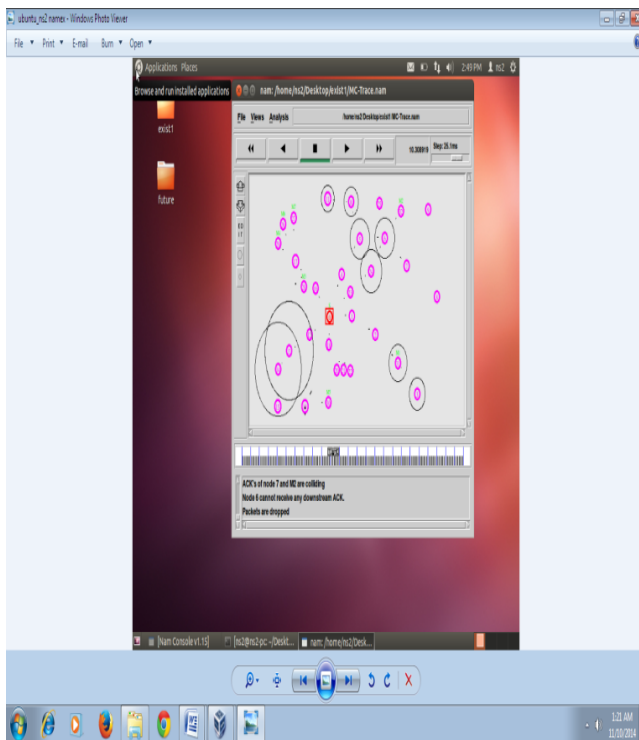


Fig 8. Packets are dropped, retransmission takes place



Fig 10. Retransmission does not take place

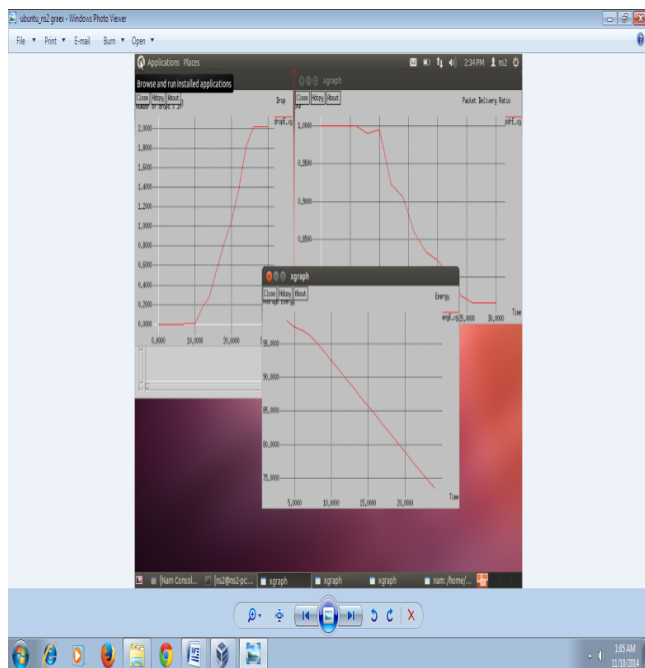


Fig 9. Drop, Energy, Packet delivery ratio are calculated

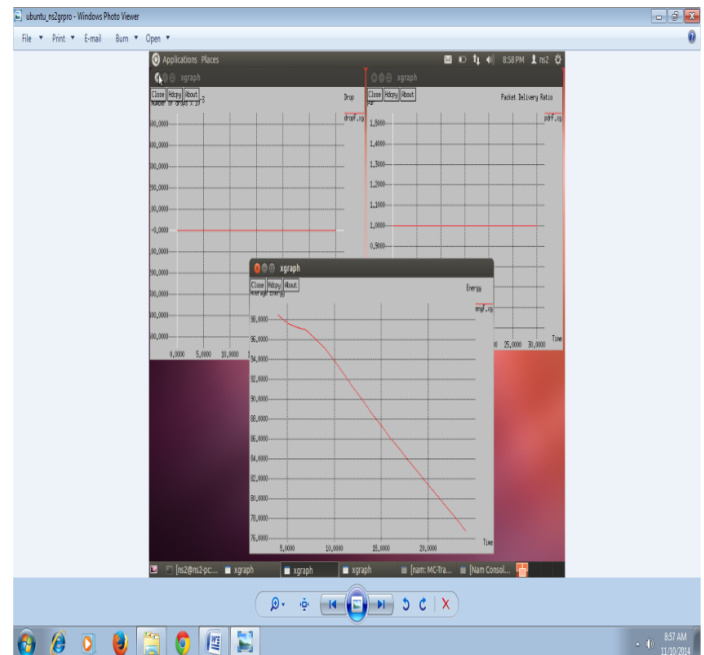


Fig 11. Drop, Energy, Packet delivery ratio are calculated

5. CONCLUSION AND FUTURE ENHANCEMENT:

We presented RoShaQ, a novel router architecture that allowed multiple buffer queues for improving network Throughput. Input packets also can bypass the shared queues to achieve low latency in the case that the network load was low. The performance interconnection network can be improved by organizing the buffers associated with each network channel. The proposed router achieves up to 19% higher saturation throughput than VC router and upto 94% of the ideal saturation throughput for synthetic traffic patterns. Compared with a typical VC router, while having the same buffer space, over synthetic traffic patterns it had 17% lower zero-load latency. It had also 5% higher throughput than a full-crossbar VC router with 3% lower power and 3% less area. In this paper energy level, dropping level and packet delivery ratio are calculated and achieved about 90%. We have suggested feature level approach is using the adaptive routing with optimized link state routing it has an advantage of having the routes immediately available when needed.

REFERENCES

- [1] .W. J. Dally, "Virtual-channel flow control," *IEEE Trans. Parallel Distrib.Syst.*, vol. 3, no. 2, pp. 194–205, Mar. 1992.
- [2] .Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar, "A 5-GHz mesh interconnect for a teraflops processor," *IEEE Micro*, vol. 27, no. 5, pp.51–61, Sep. 2007.
- [3] .T. Moscibroda and O. Mutlu, "A case for bufferless routing in on-chip networks," in *Proc. ISCA*, Jun. 2009, pp. 196–207.
- [4] .M. Hayenga, N. E. Jerger, and M. Lipasti, "SCARAB: A single cycle adaptive routing and bufferless network," in *Proc. 42nd Ann. IEEE/ACM Int. Symp. Micro architect.*, Dec. 2009, pp. 244–254.
- [5] .G. Michelogiannakis, D. Sanchez, W. J. Dally, and C. Kozyrakis, "Evaluating buffer less flow control for on-chip networks," in *Proc. 4th NOCS*, 2010, pp. 9–16.
- [6] .K. Latif, T. Seceleanu, and H. Tenhunen, "Power and area efficient design of network-on-chip router through utilization of idle buffers," in *Proc. 17th IEEE Int. Conf. Workshops ECBS*, Mar. 2010, pp.131-138.
- [7] R. S. Ramanujam, V. Soteriou, B. Lin, and L.-S. Peh, "Extending the effective throughput of NoCs with distributed shared-buffer routers," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 30, no. 4. Apr. 2011.
- [8].T. Nakamura, H. Matsutani, M. Koibuchi, K.Usami, and H. Amano, "Fine-grained power control using a Multi-voltage variable pipeline router," in *Proc. Int. Symp.Embedded Multicore Soccs*, Sep. 2012,59–66.
- [9]. H. Matsutani, Y. Hirata, M. Koibuchi, K. Usami, H. Nakamura, and H. Amano, "A multi-Vdd dynamic variable-pipeline on-chip router CMPs," in *Proc. 17th ASP DAC*, Feb. 2012, pp. 407–412.
- [10] .K. Latif, T. Seceleanu, and H. Tenhunen, "Power and Area Efficient design of network-on-chip router Through utilization of idle buffers," in *Proc. 17th IEEE Int. Conf. Workshops ECBS*, Mar. 2010, pp. 131–138.
- [11] .R. S. Ramanujam, V. Soteriou, B. Lin, and L.-S. Peh, "Extending the effective throughput of NoCs with Distributed shared-buffer routers," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol.30,no. 4,pp. 548–561, Apr. 2011.
- [12]. S. Iyer, R. Zhang, and N. McKeown, "Routers with a Single stage of buffering," in *Proc. ACM Conf. Applicat., Technol., Architect., Protocols Comput. Commun. (SIGCOMM)*, Sep. 2002, pp. 251–264.
- [13] .A. Prakash, A. Aziz, and V. Ramachandran, "Randomized parallel schedulers for switch-memory-switch routers: Analysis and numerical studies," in *Proc. 23rd Conf. IEEE Comput. Commun. Soc. (INFOCOM)*, Mar. 2004, pp. 2026–2037.
- [14] .M. Galles, "Spider: A high-speed network interconnect," *IEEE Micro*, vol. 17, no. 1, pp. 34–39, Jan. 1997.
- [15].Y. Tamir, H.C. Chi, "Symmetric crossbar arbiters for VLSI communication switches", *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no. 1, pp.13- 27, 1993.
- [16].K. Mai, Smart Memories: A Modular Reconfigurable Architecture, Proc ISCA, June 2000, pp. 161-71.
- [17] .W. J. Dally and C. L. Seitz, "Deadlock-free message Routing in multiprocessor interconnection networks," *IEEE Trans. Comput.*, vol. 36, no. 5, pp. 547–553, May 1987.

- [18] .M. G. Hluchyj and M. J. Karol, "Queuing in high-Performance packet switching," *IEEE J. Sel. Areas Commun.*, vol. 6, no. 9, pp. 1587–1597, Dec. 1988.
- [19].R. E. Shannon, "Introduction to the art and science Of simulation," in *Proc. Of the 30th conference on Winter simulation (WSC'98)*, 1989.
- [20]. R. Lu, A. Cao, and C. Koh, "SAMBA-Bus: A High Performance Bus Architecture of System- on-Chips", *IEEE Transactions on VLSI Systems*, Vol. 15, Issue 1, pp. 69-79, Jan 2007.
- [21] .W. J. Dally and B. Towels, *Principles and Practices of Interconnection Networks*. San Francisco, CA, USA: Morgan Kaufmann, 2004.
- [22] .A. T. Tran and B. M. Baas, "RoShaQ: High-Performance on-chip router with shared queues," in *Proc. ICCD*, Oct. 2011, pp. 232–238.