www.ijecs.in
International Journal Of Engineering And Computer Science ISSN: 2319-7242
Volume 4 Issue 9 Sep 2015, Page No. 14204-14213

Design of High Performance Shared Buffer NoC Router

B.S.L.K.Anusha[1], *K.S.N.Raju*[2]
[1]*PG Student [VLSID]*, [2]*Assistant Professor*
Department of ECE, Shri Vishnu Engineering College for Women, Bhimavaram, India
[1]anusha.bslk@gmail.com
[2]ksnraju@svecw.edu.in

Abstract: **In a NoC system, modules such as processor cores, memories and specialized IP Cores exchange data using a network via multiple point to point data links interconnected by switches (Routers) by making routing decisions at Routers. These Routers generally have Buffers dedicated to their input or output ports for temporarily storing packets in case contention occurs at output physical channels. These buffers, in fact, consume significant portions of router area and power. So, sharing these buffers among input or output ports makes using the buffers more efficient. The Extended Router Architecture with Shared Queues (E-RoShaQ), shares the buffers among output ports making the operation dead-lock free. With this architecture, the performance of router has increased by 50% over conventional router architecture at all the traffic loads. This work focuses on realization of both conventional and Extended RoShaQ. The E-RoShaQ is designed with 4 input ports, 4 output ports and Shared Queue length of 8 with 2 Buffer entries each, using Field Programmable Gate Array (FPGA) technology.**

Keywords: **Network-on-chip (NoC), Router, Flits, Queues, Buffers.**

INTRODUCTION

Network-on-Chip (NoC) is an emerging paradigm for communication with in large VLSI systems implemented on single silicon chip. NoC are found to be feasible and easy to scale for supporting a large number of processing elements rather than point to point interconnect wires or shared buses [3]. Traditionally ICs have been designed with dedicated point-to-point connections, with one wire dedicated to each signal. For large designs, in particular, this has several limitations from a physical design view point. The wires occupy much of the area of chip, and in nanometer CMOS technology, interconnects dominate both performance and dynamic power dissipation, as signal propagation in wires across the chip requires multiple clock cycles. NoC links reduce the complexity of designing wires for predictable speed,power,noise,reliability,etc., due to their regular, well controlled structure. Fig.1 depicts a multicore system in which processors communicate together through a 2-D mesh network of routers. A network interface (NI) locates between a processor and its router for transforming processor messages into packets to be transferred on the network and viceversa.

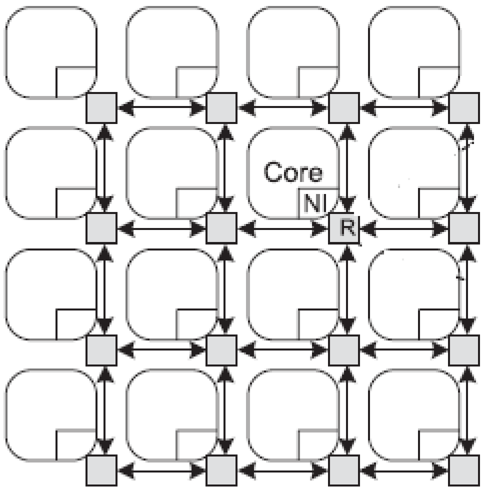


Fig.1 Multi Cores connected by a network of Routers
NI: Network Interface R: Router

In a typical router, each input port has an input buffer for temporarily storing packets in cases that output physical channels are busy. These buffers can be a single queue as in warmhole (WH) router or multiple queues in parallel as in virtual channel (VC) router[4]. These buffers, in fact, consume significant portions of area and power, that can be 60% and 30% of the whole router, respectively. Bufferless routers remove buffers from the router so save much area [5] however, their performance becomes poor that in many cases cannot meet application requirements in which packet injection rates are high. While running a traffic trace, not all the input ports have incoming packets needed to be transferred simultaneously. Therefore, a large number of buffer queues in the network are empty and the other queues are mostly busy.

This situation leads to the design of a router architecture that uses Shared Queues, to maximize the buffer utilization. Sharing queues, infact, makes using buffers more efficiect hence be able to achieve higher throughput when the network load is heavy. On the other side, at light traffic load , the router has the capability to bypass the shared queues,achieving low latency.

CONVENTIONAL RoShaQ : ROUTER ARCHITECTURE WITH SHARED QUEUES

A.Concept of sharing Queues

For maximizing queue utilization, a router is designed to share all the queues [1],[2]. With this architecture, incoming packets from an input port has to be written to shared queue before reaching the output port even if the data is facing congestion or not. Also because this architecture has no buffer at the input port, when a packet needs to be forwarded, it has to send request to the destination router and wait for grant before sending data. Therefore the shared queue arbiter for each router is highly complicated because it must handle many external requests from multiple shared queues of all neighbouring routers and also the intra router requests.

To alleviate this latency, each input port is dedicated with one buffer queue and share all remaining queues, also providing it with a facility to bypass the shared queues as shown in the Fig.2. With this design, a packet from an input queues simultaneously arbitrates for both shared queues and an output port. If it wins the output port, it would be forwarded to the downstream router in the next cycle. Otherwise, that means having congestion at the corresponding output port., it can be buffered to the shared queues in the same cycle. At low load, the network would have low latency because packets can 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.

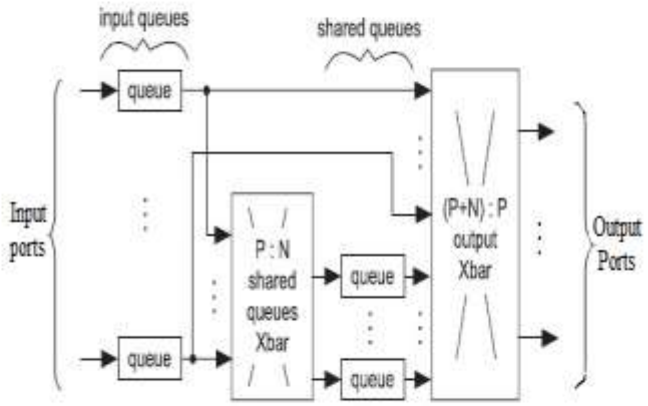


Fig.2 Concept of Sharing Queues.
P:number of router ports N: number of shared queues

B.RoShaQ Architecture

RoshaQ, a router architecture with shared queues based on the idea of Fig.2 is shown in Fig.3. When an input port receives a packet, it calcualtes its output port for the next router. At the same time it arbitrates for both its decided output port and shared queues. If it receives a grant from the Output Port Allocator (OPA), 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. Incase that it receives both grants, it will prioritize to advance to the output port.

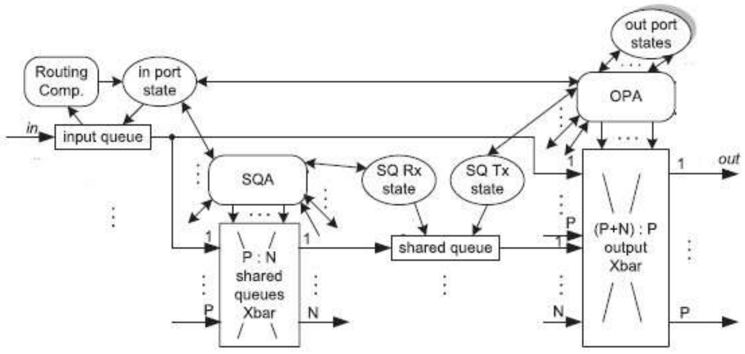


Fig.3. Generalized architecture of RoShaQ router. SQA:Shared Queue Allocator OPA:Output Port Allocator SQRx State:Shared Queue receiving/writing state SQTx State: Shared Queue Transmitting/reading State. P: Number of Router Ports N:Number of Shared Queues

Shared Queue Allocator (SQA) receives requests from all input queues and grants the permission to their packets for accessing empty shared queues. Packets from input queues are allowed to write to shared queue only if: (1) 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 the input queues and shared queues.

C.Realization of conventional RoShaQ

The realization of conventional RoShaQ is represented in this level. Fig 4 represents the top level block diagram for the conventional RoShaQ. It has 4 Input Ports, 4 Output Ports and 4 Shared Queues with 2 Buffer entries for each shared queue.

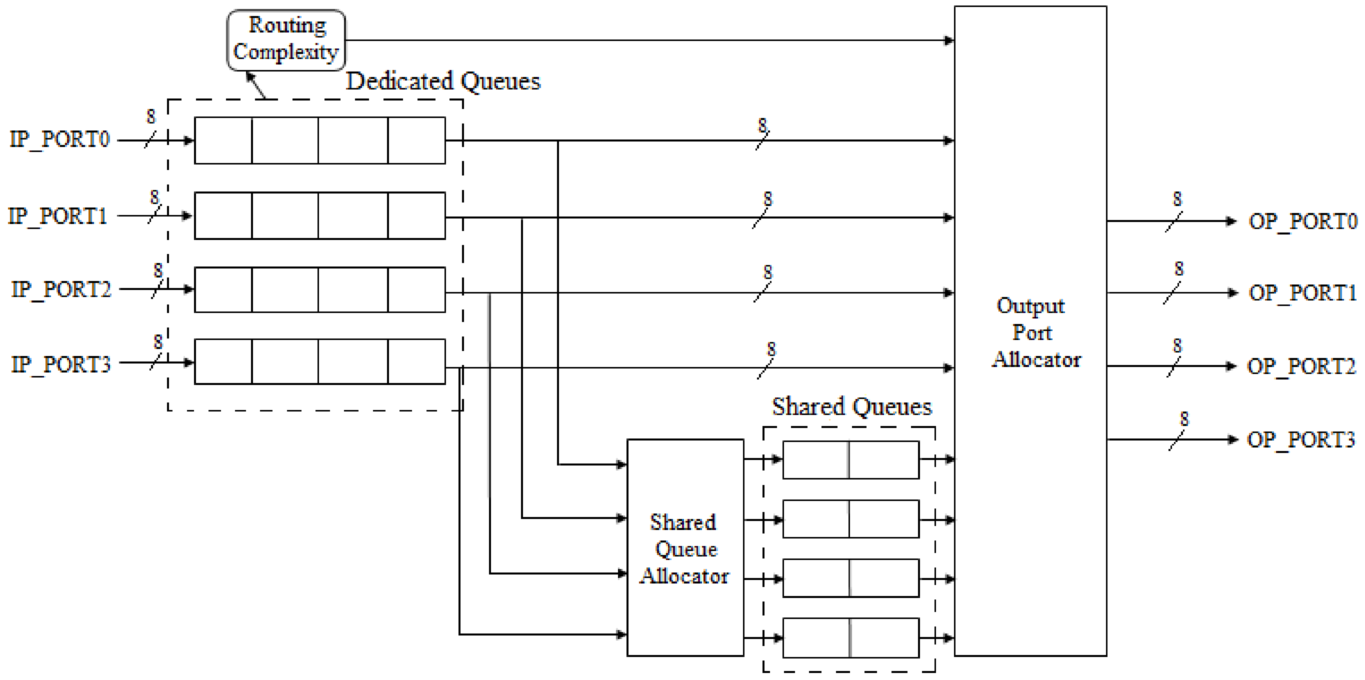


Fig.4 Realization of conventional RoShaQ

The architecture uses a dedicated queue for each input port and a set of 4 shared queues. At each time cycle, every input port sends a data packet asking for any of the output ports. Based on the output port each input packet is requesting for, the arbitration starts.

1)Structure of Data Packet: The Data Packet we have taken is of length 8 bits. Each packet is divided into 3 Flits, Head Flit, Body Flit and Tail Flit. Of the 8bits of a data packet, Head Flit and Tail Flit are of 2 bits each and Body Flit is of 4bits. Fig.5 shows the sturcture of the data packet. Head Flit contains the address of output port port which the packet is requesting for. Tail Flit contains address of input port from which it is coming and the Body Flit contains the data, the packet is carrying.

DOI: 10.18535/ijecs/v4i9.24

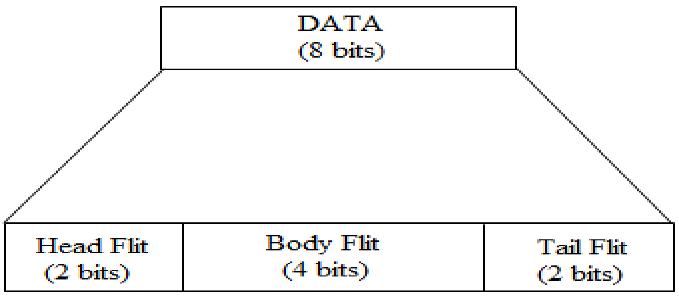


Fig. 5. Sturcture of Data Packet

2)Routing Complexity: Routing Complexity decides the traffic load status of the incoming packets. If any of the two input ports are asking for the same output port, it is said that the traffic load is High. If all the input ports are asking for different output ports, it is said that the traffic load is Low. As the router architecture has capability to bypass the shared queues, at low network load cases, and uses shared queues at heavy network load cases, routing complexity decides when to use shared queues and when not to use.

3) Dedicated Queues: Each input port is accommodated with a Dedicated Queue with 4 buffers each. Dedicated buffers accommodate data if the packet does not get grant for both the output port and also to store in the shared queues. As the data packet arrives at the input port, it is written to the Dedicated Queue. This is called Queue Write (QW). Then the packets arbitrates at the same time for shared queues and also the output ports. If it gets grant from any one of those., it is written into the corresponding one or else it will be stored in the dedicated queue itself waiting for the grant to come. Fig. 6 gives the view of 4 dedicated queues, one for each input port and the buffers of them. A data packet when it is about to write into the queues, it checks if the buffers are empty. The packet is written into the empty buffer of the Queue.

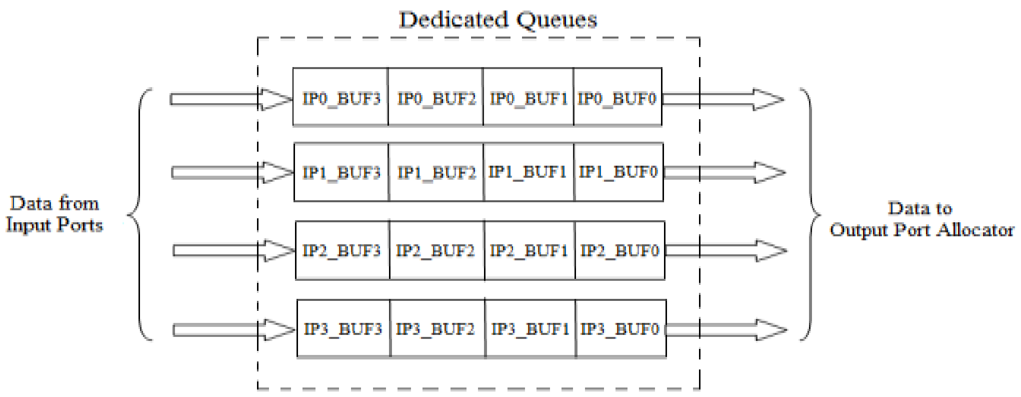


Fig. 6. View of Dedicated Queues

4) Shared Queues: This architecture uses 4 shared queues with 2 buffer entries each. A data packet that doesnot get grant to the output ports will arbitrate for shared queues. For a packet to be written into the shared queues it has to satisfy the two conditions mentioned in section B, which ensures deadlock free operation of the router. Fig. 7. gives the view of shared queues and buffers in shared queues.

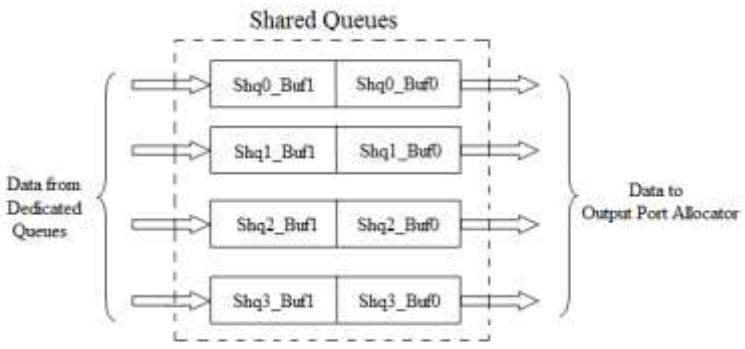


Fig. 7. View of Shared Queues

5) Allocators: This architecture uses two types of allocators, Output Port Allocator(OPA) and Shared Queue Allocator (SQA). Output Port Allocator decides whether the data can be sent to the requested Output Port or not. This arbiter sets priority for a specific input port and if morethan one input port is asking for same output port, it gives grant to the data from input port for which the priority has set. And at the same time the remaning data packets will already be arbitrating for shared queues and by the time the packets are rejected by the OPA, either they should have get grant from Shared Queue Allocator (SQA) or even rejected by the SQA too. If the latter condition was satisfied., the packet will be stored in dedicated queues itself.

EXTENDED RoShaQ

After the realization of conventional router, to increase the performance of the router to a further extent, we have increased the count of shared queues. We have added an another set of shared queues which can accommodate 8 data packets. We have added 4 shared queues with 2 buffers each. Fig. 8 shows the architecture of Extended Router Architecture with Shared Queues (E-RoShaQ).

In this architecture, it uses another Shared Queue Allocator(SQA) which works on same conditions as of conventional shared queue allocator. In this architecture, if the data packet does not get grant for first set of Shared queues, it arbitrates for second set of shared queues. Even if it doesnot get grant for second set of shared queues also, it wll be stored in dedicated queues. So, on a whole, the packet is stored in the dedicated queues only if both the sets of shared queues are full.

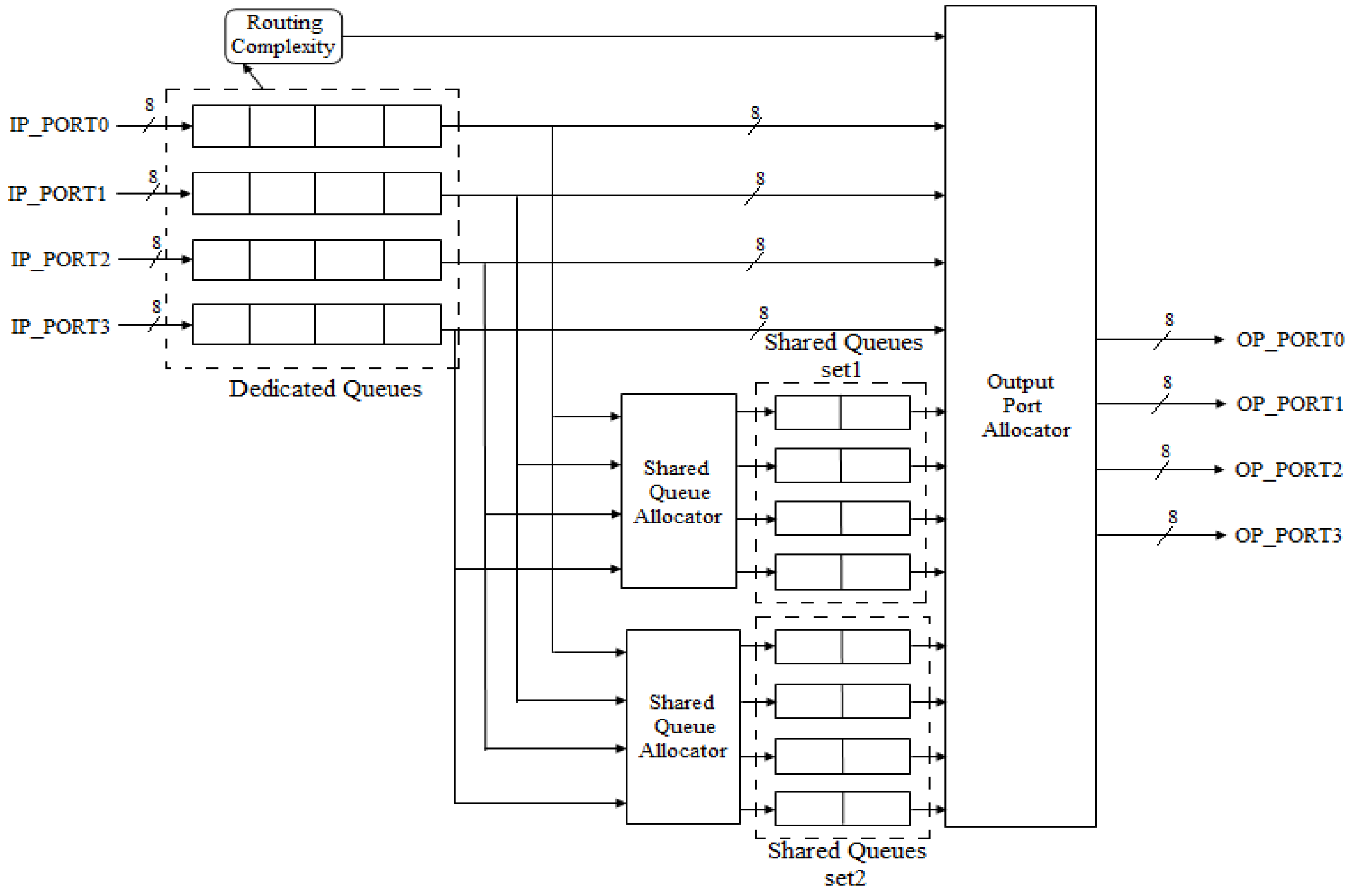
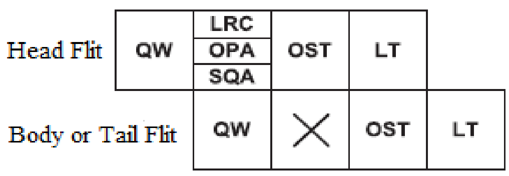


Fig. 8. Extended RoShaQ

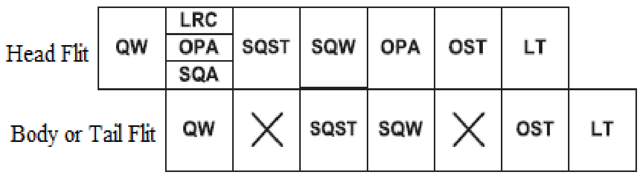
IV. DATAPATH PIPELINE OF RoShaQ

After a packet is written into a dedicated queue in the first cycle, in the second cycle it simultaneously performs three operations: LRC,OPA and SQA. At low network load, there is a high chance for the packet to win the OPA due to low congestion at its desired output port, hence it is granted to traverse through the output crossbar and output link towards next router. Therefore, it incurs four stages including link traversal as shown in Fig.9(a)

When network load becomes heavy, the packet at an input queue may fail to get grant from OPA, but it can get a grant from SQA and is allowed to traverse the shared queue crossbar and write to the granted shared queue in the next cycles. After that, it arbitrates for the output port again and would traverse across the output crossbar and output channel towards the next router at next cycles if it is granted by the OPA at this time. Thus, in this situation, it incurs seven inter router stages as shown in Fig. 9(b). This larger number of traversing stages, infact, allows the router to use shared queues for reducing stall times of packets at input queues, hence improves throughput at heavy network load. In both cases, body and tail flits of a packet traverse through the router in the same way as its head flit, except they do not need to arbitrate for the resources, as they are already reserved by the head flit.



(a)



(b)

Fig. 9. RoShaQ pipeline (a) at light load case (b) at heavy load case QW: Queue Write LRC: Look Ahead Routing OPA:Output Port Allocation SQA:Shared Queue Allocation OST: Output Switch Traversal LT:Output Link Traversal SQST: Shared Queue Switch Traversal SQW: Shared Queue Write (X): Pipeline stall

EXPERIMENTAL RESULTS

The router is coded using Verilog HDL and the results are observed in ISim Simulator. The architecture is verified under several conditions under all the possible combinations of data i.e., when each input port is asking for different output ports, two input ports asking for same output port, three input ports are asking for same output port and finally the results are examined under the worst case condition where all the four input ports are asking for same output port continuously, for both Conventional RoShaQ and Extended RoShaQ .

1) Routing Complexity: If all the input ports are asking for different output ports, Routing Complexity should be Low. In any other case, it has to be High. Only the Head Flits of all the datapackets are considered as inputs for this module.

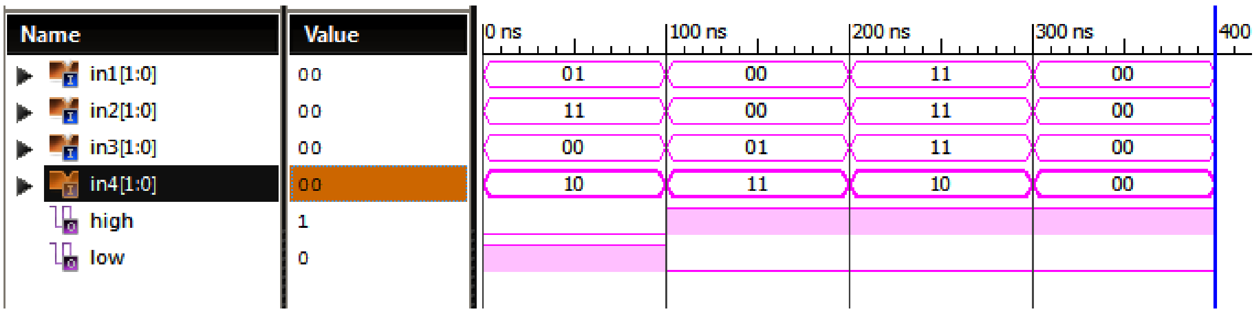


Fig. 10. Simulation Results of Routing Complexity

2)When there is no congestion: If all the 4 input ports are asking for different output ports, all the data packets will win their respective output ports and all the queues are empty.

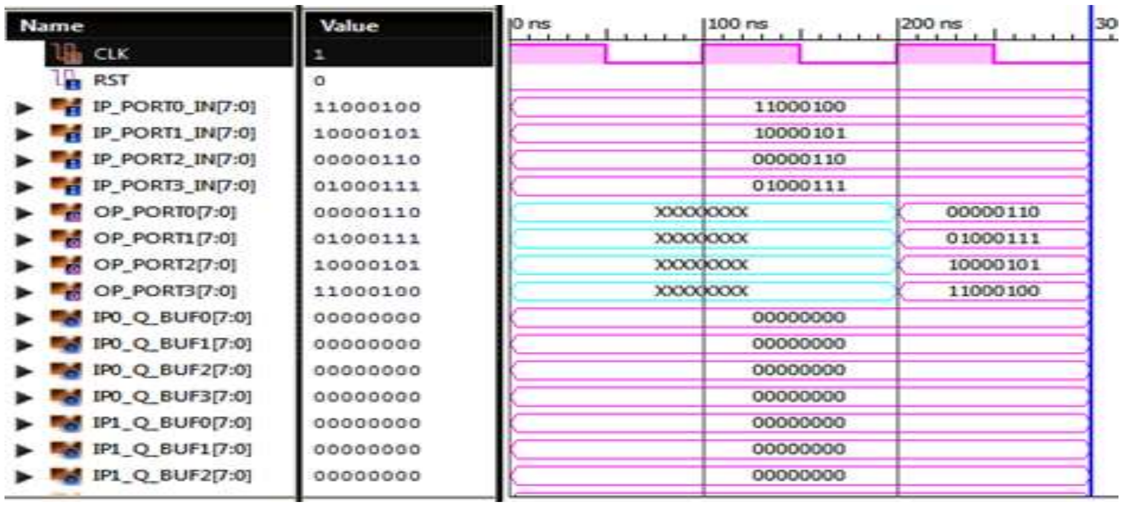
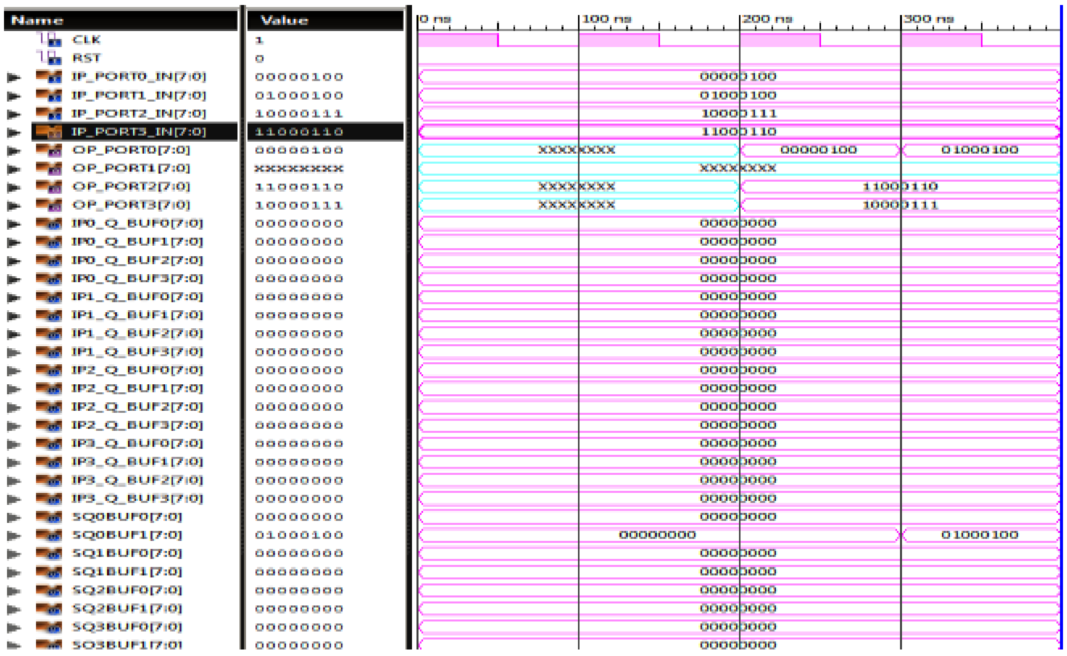


Fig. 11. Simulation results showing when there is no congestion

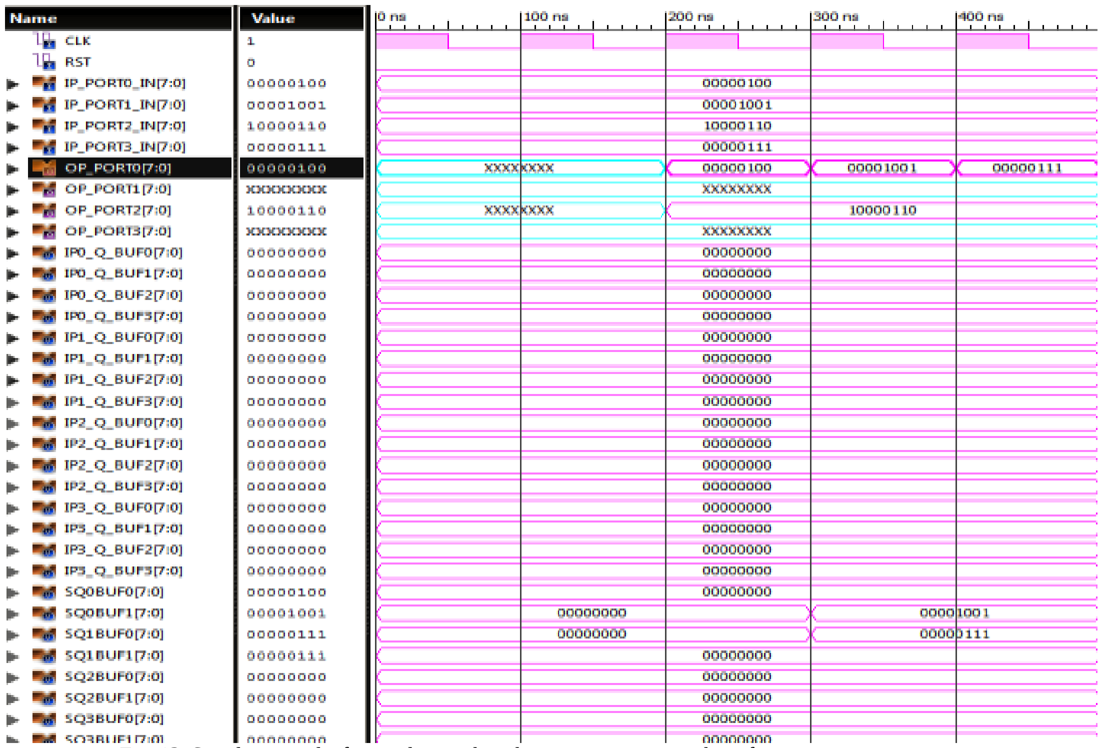
3)When two input ports are asking for same output port: In this condition, as two input ports are asking for same output port, only one will get grant and the other will not. The packet from higher priority input port will get grant and the other packet will be stored in Shared Queue. The packets from other two ports, which are asking for free output channels will be deliverd to their respective output ports.



Name	Value
CLK	1
RST	0
IP_PORT0_IN[7:0]	00000100
IP_PORT1_IN[7:0]	01000100
IP_PORT2_IN[7:0]	10000111
IP_PORT3_IN[7:0]	11000110
OP_PORT0[7:0]	00000100
OP_PORT1[7:0]	XXXXXXXX
OP_PORT2[7:0]	11000110
OP_PORT3[7:0]	10000111
IP0_Q_BUF0[7:0]	00000000
IP0_Q_BUF1[7:0]	00000000
IP0_Q_BUF2[7:0]	00000000
IP0_Q_BUF3[7:0]	00000000
IP1_Q_BUF0[7:0]	00000000
IP1_Q_BUF1[7:0]	00000000
IP1_Q_BUF2[7:0]	00000000
IP1_Q_BUF3[7:0]	00000000
IP2_Q_BUF0[7:0]	00000000
IP2_Q_BUF1[7:0]	00000000
IP2_Q_BUF2[7:0]	00000000
IP2_Q_BUF3[7:0]	00000000
IP3_Q_BUF0[7:0]	00000000
IP3_Q_BUF1[7:0]	00000000
IP3_Q_BUF2[7:0]	00000000
IP3_Q_BUF3[7:0]	00000000
SQ0BUF0[7:0]	00000000
SQ0BUF1[7:0]	01000100
SQ1BUF0[7:0]	00000000
SQ1BUF1[7:0]	00000000
SQ2BUF0[7:0]	00000000
SQ2BUF1[7:0]	00000000
SQ3BUF0[7:0]	00000000
SQ3BUF1[7:0]	00000000

Fig. 12. Simultion results for condition when two input ports are asking for same output port

4)When three input ports are asking for same output port: In this condition, as three input ports are asking for same output port, only one will get grant and the other two will not. The packet from higher priority input port will get grant and the other packets will be stored in Shared Queue.



Name	Value
CLK	1
RST	0
IP_PORT0_IN[7:0]	00000100
IP_PORT1_IN[7:0]	00001001
IP_PORT2_IN[7:0]	10000110
IP_PORT3_IN[7:0]	00000111
OP_PORT0[7:0]	00000100
OP_PORT1[7:0]	XXXXXXXX
OP_PORT2[7:0]	10000110
OP_PORT3[7:0]	XXXXXXXX
IP0_Q_BUF0[7:0]	00000000
IP0_Q_BUF1[7:0]	00000000
IP0_Q_BUF2[7:0]	00000000
IP0_Q_BUF3[7:0]	00000000
IP1_Q_BUF0[7:0]	00000000
IP1_Q_BUF1[7:0]	00000000
IP1_Q_BUF2[7:0]	00000000
IP1_Q_BUF3[7:0]	00000000
IP2_Q_BUF0[7:0]	00000000
IP2_Q_BUF1[7:0]	00000000
IP2_Q_BUF2[7:0]	00000000
IP2_Q_BUF3[7:0]	00000000
IP3_Q_BUF0[7:0]	00000000
IP3_Q_BUF1[7:0]	00000000
IP3_Q_BUF2[7:0]	00000000
IP3_Q_BUF3[7:0]	00000000
SQ0BUF0[7:0]	00000100
SQ0BUF1[7:0]	00001001
SQ1BUF0[7:0]	00000111
SQ1BUF1[7:0]	00000111
SQ2BUF0[7:0]	00000000
SQ2BUF1[7:0]	00000000
SQ3BUF0[7:0]	00000000
SQ3BUF1[7:0]	00000000

Fig. 13. Simultion results for condition when three input ports are asking for same output port

5)When all input ports are asking for same output port: In this condition, as four input ports are asking for same output port, only one will get grant and the other three will not. The packet from higher priority input port will get grant and the other packets will be stored in Shared Queue. Considering the worst case condition, if that is run for a number of cycles, we can abserve that data given in 8th cycle and 10th cycle are lost after all the shared buffers and corresponding dedicated queues are filled. Fig.14 and Fig. 15 are results of Conventional RoShaQ and Fig.16 shows results of Extended RoShaQ.

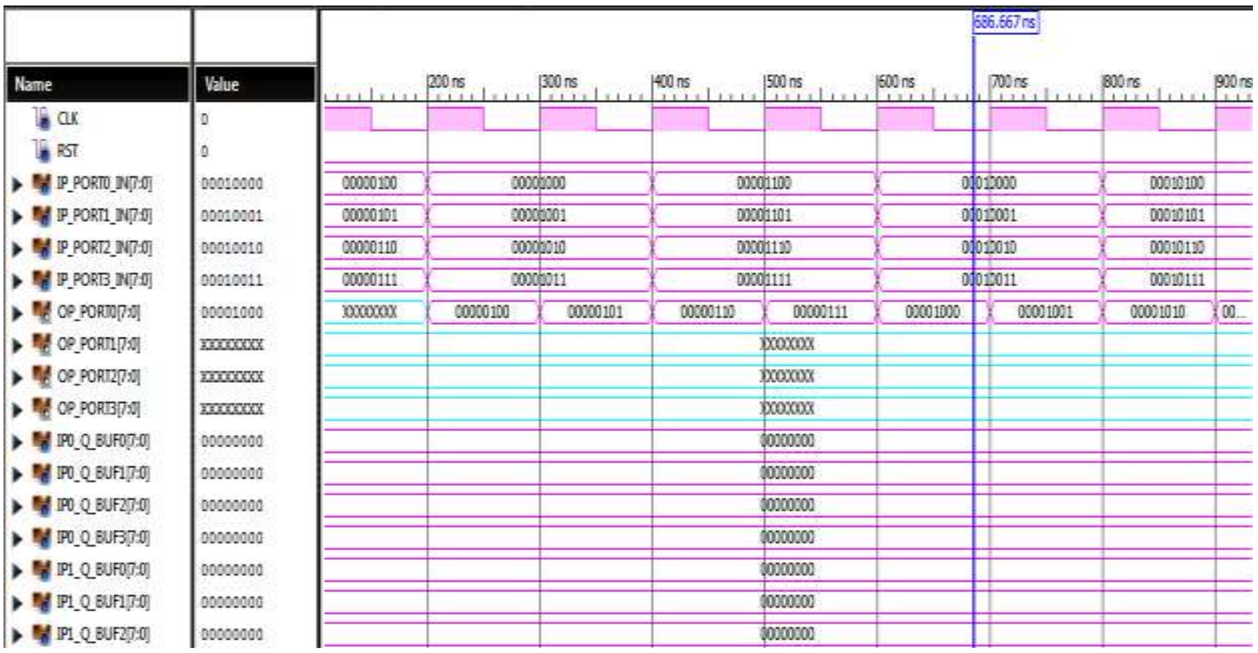


Fig. 14. Simultion results for condition when four input ports are asking for same output port

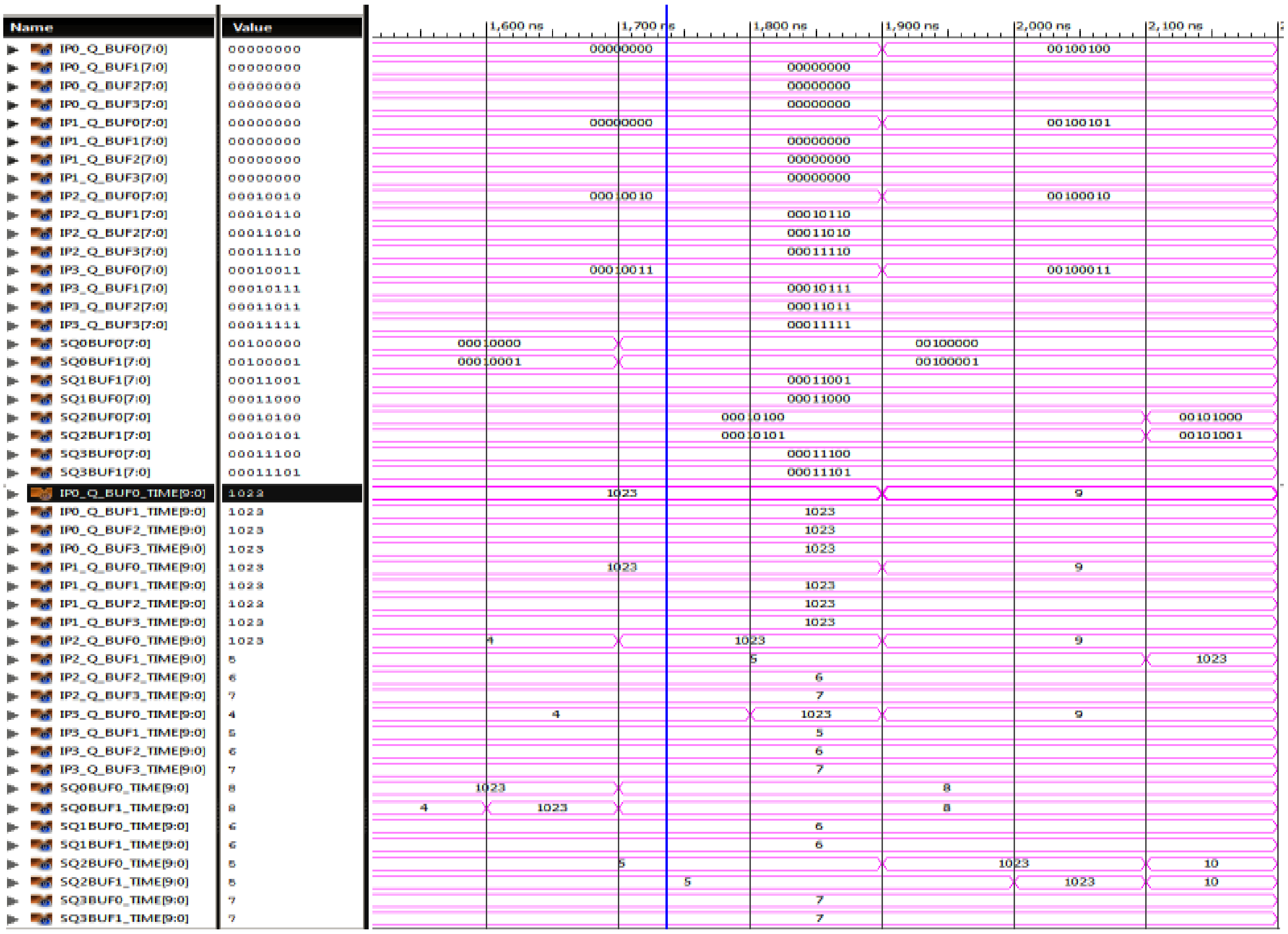


Fig. 15. Simultion results for condition when four input ports are asking for same output port, status of Queues and timing of datas

In this condition, as four input ports are asking for same output port, only one will get grant and the other three will not. The packet from higher priority input port will get grant and the other packets will be stored in Shared Queue. Considering the same data is given to E-RoShaQ, and run for a number of cycles, we can observe that data given in 8th cycle and 10th cycle datas that are lost in conventioanal architecture were made not to loss.

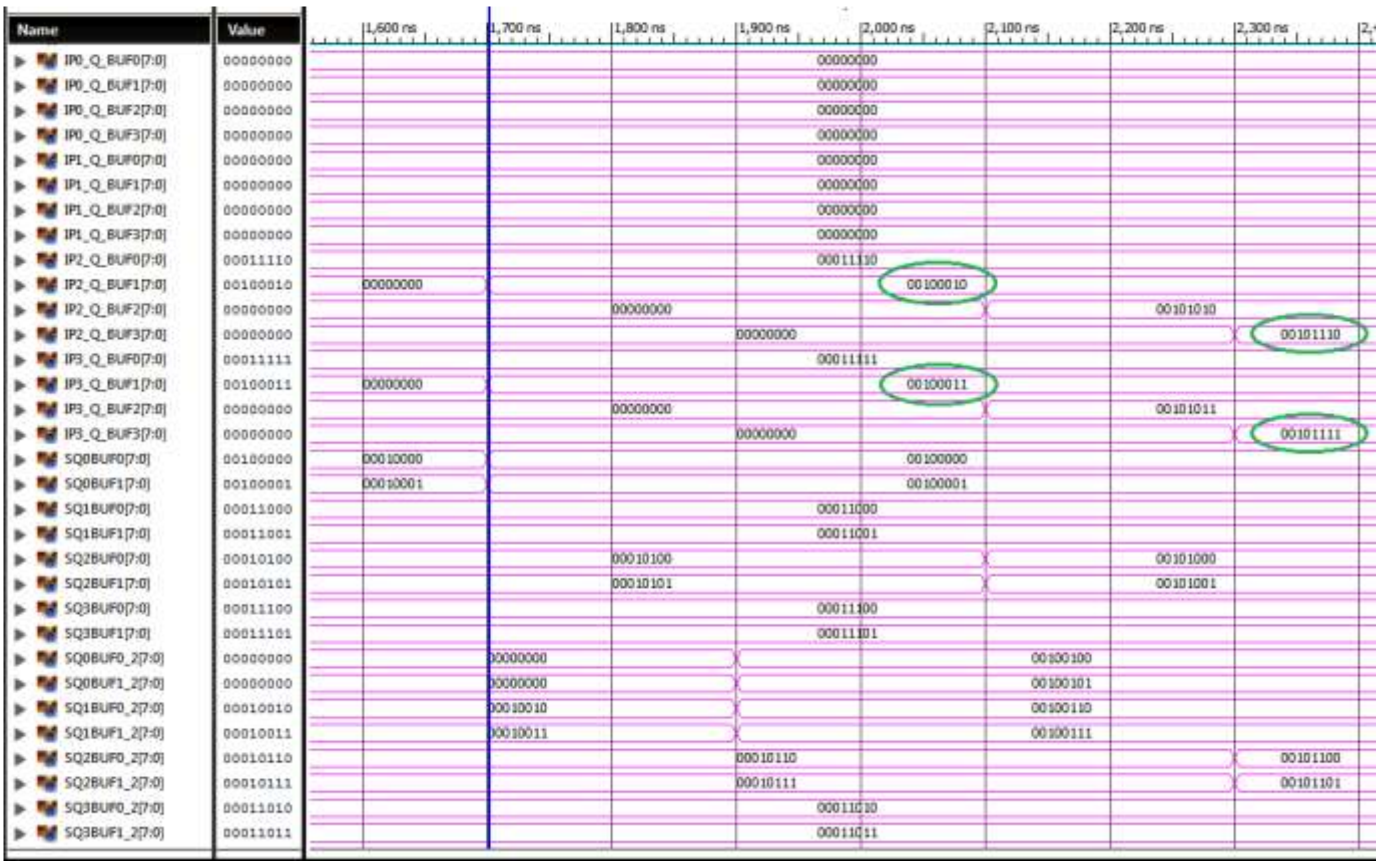


Fig. 15. Simultion results for condition when four input ports are asking for same output port, showing data that hasnot lost due to Extended RoShaQ

COMPARISION OF TWO ROUTERS

Experimental results show that Extended RoShaQ is giving 50% higher saturation throughput than the conventional RoShaQ with 2% larger area due to its higher performance. Table 1 shows the logic utilization for both the routers on a Spartan 6 device.

TABLE1
COMPARISION OF TWO ROUTERS

LOGIC UTILIZATION	Conventional RoShaQ	Extended RoShaQ
Number of LUT's	575(0%)	3567(2%)
Number of Fully used LUT-FF Pairs	460(11%)	1892(13%)

CONCLUSION

In this paper, implementation of Conventional RoShaQ and Extended RoShaQ is done and the results are observed for both the routers for different input sequences in all the possible conditions and the synthesis is done using XILINX 14.2 observing just a little increase in use of LUT's, achieving higher performance than conventional routers.

REFERENCES

[1] Anh T. Tran and Bevan M. Bass, "Achieving High Performance On-Chip Networks With Shared Buffer Routers", IEEE Transactions on VLSI Systems, Vol 22, No.6,June 2014

DOI: 10.18535/ijecs/v4i9.24

[2] Anh T. Tran and Bevan M. Bass, “RoShaQ: High Performance On-Chip Router with Share Queues”, ICCD, Oct 2011
[3] William J Dally and B.Towels, “Route Packets, not Wires: On-Chip interconnection Networks,” DAC2001
[4] William J Dally, “Virtual channel Flow Control”, IEEE Transactions on Parallel Distribution. Systems., Vol 3, No 2, Mar 1992
[5] T.Moscibroda and O.Mutlu, “A case for bufferless routing in Onchip Networks” ISCA, June 2009
[6] A.T. Tran, D.N. Truong and B.M.Baas, “A GALS many-core heterogenous DSP platform with source-synchronous on-chip interconnection network,” in Proc. ACM/IEEE Int. NOCS, May 2009
[7] E.Begine, “An asynchronous power aware and adaptive NoC based circuit”, IEEE J. Solid State Circuits, vol 44, Apr. 2009
[8] R.S.Ramanujam, V.Soteriou, B.Lin and L.S.Peh, “Extending the effective throughput of NoC’s with Distributed Shared Buffer Routers”, IEEE Transactions on Computer-Aided Design Integrated Circuits Systems., Vol 30, no. 4, Apr 2011