Research Journal of Applied Sciences, Engineering and Technology 8(6): 706-713, 2014
DOI:10.19026/rjaset.8.1026
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: December 28, 2013 Accepted: January 20, 2014 Published: August 15, 2014

**Research Article**

# Performance Analysis on Router Arbitration for On-chip Networking

[1]G. Selvaraj and [2]K.R. Kashwan
[1]Department of Electronics and Communication Engineering, Bhaktavatsalam Polytechnic College, Karaipettai, Kanchipuram-631552, TN, India
[2]Department of Electronics and Communication Engineering-PG, Sona College of Technology, Anna University, TPT Road, Salem-636005, TN, India

**Abstract:** This study is a comprehensive report on performance analyses of Round Robin and matrix arbitrations to enhance the reliability of on-chip networks. Arbiter is used in Network-on-Chip (NoC) router when number of input ports requested is the same as output ports. If many inputs are requested for same output port, the matrix arbiter deals it by forming a 5×5 matrix based on input and output ports. Next, it allots the priority to the requested input ports and simultaneously generates a control signal for selecting the input port to send the packet to output port. The Robin arbiter generates the grant signal on the basis of priority allotted to the input ports. The simulation results of arbitration analysis shows that the router design of front end model consumes less power by 8% and occupies smaller area by 3% on chip. The area on chip is around 64% of available area using Round Robin arbitration compare to that of matrix arbitration. This study also implements hamming distance in order to check the error free data transmission of the NoC router.

**Keywords:** Arbiter, hamming code, matrix arbiter, network-on-chip, round robin arbiter, router

## INTRODUCTION

ASIC's, being costly and needing longer time for manufacturing, are no more preferred, as better options such as FPGAs are available for application circuits to be implemented. Higher end FPGAs with embedded processors can be used for System-on-Chip (SoC) designs. SoC system is a complex interconnected network of many functional blocks. Interconnections of cores in SoC are challenging networks for the inter-block or inter-element communications. Existing bus based interconnect architectures are not suitable for scalable solutions. The NoC has recently been projected as a solution to the above mentioned problems in the communication taking place on chip. The NoC performs interconnections of various IP cores using on chip networks as compared to traditional shared bus approach. The interconnection is achieved by means of routers. The signals are transmitted using packet switching mechanism. NoC has many advantages over other options such as traditional bus oriented systems. The advantages such as scalability, IP reusability, better performance and modularity in architectures are of great usefulness (Dally and Towles, 2001).

Chip area utilization is desirable at the minimum while implementing any circuit on an FPGA. The programming should be good enough to reduce the area and thus it leads to a less memory space requirement. The communication network system on the chip must be compact and small in order to be faster for data packet transfers and same time it must use less memory space in order to reduce overall chip area.

The router in a NoC is a very important element. It must be designed in such a way that it needs least memory space and works fast. In this study we present a compact router design for NoC which is implemented on FPGAs. It can support five parallel connections simultaneously. The router uses store and forward type of flow control and *X-Y* routing. Reducing the size of the Finite State Machine (FSM) for *X-Y* routing and performing a simple logical OR operation for the Select/Gnt lines has resulted in reduction in the number of slices. The chip area reduction directly improves the performance and power consumption ratings.

For an efficient communication, the data flow control of virtual channel is important for smooth packet transfers. The architecture and dataflow control affects the arbiter design for an NOC. One very stringent requirement for an NoC is that it should have high speed switches. This can be very helpful scheme if a large number of packets from input ports is to be transmitted over an output port with priority scheduling in place. Fast arbiter leads to high performance of NoC switches and performs scheduling more efficiently. It also provide high throughput for on-chip network routing. We propose an arbitration analyses on Round

**Corresponding Author:** G. Selvaraj, Department of Electronics and Communication Engineering, Bhaktavatsalam Polytechnic College, Karaipettai, Kanchipuram-631552, TN, India

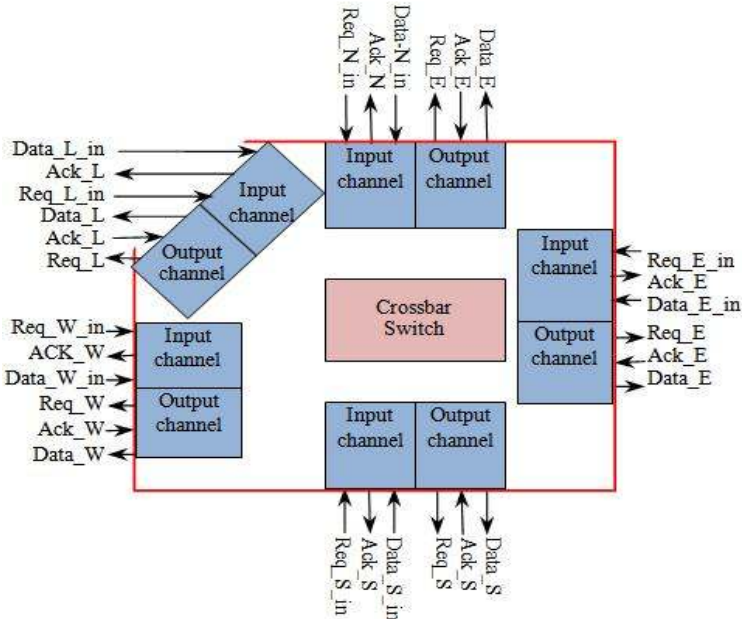



Fig. 1: General block diagram of router architecture for a network-on-chip

Robin and matrix arbitrations in a comparative manner. The implementation is done on FPGA at reduced power consumption and less number of slices used. The design is more efficient in terms of on chip area of NoC router architecture.

## METHODOLOGY

**Implementation of NoC router:**

**Router architecture:** In this study we have designed a parallel router with lower area utilization. The general architecture is illustrated in Fig. 1 as shown below. The motivation to reduce the area is based on the fact that the less is area, the lesser is the power consumption. We have chosen one of the popular methods of buffering techniques of storing and then forwarding. This has proven to be the simplest possible decoding logic. Obviously reducing both area and power has been achieved. The connections are established automatically with simpler decoding logic.

**Router design:** The router consists of five ports namely east, west, north, south and local port and a central cross point matrix (Fig. 1). Each of the ports has an input channel and an output channel to receive and transmit data packets, respectively. Data packets move through the input channel of one port of router and then these are forwarded to the output channel of other port.

Each, input and output channels, have decoding logics which help to perform the functions of the router. Buffers are incorporated at all ports to store the data temporarily. The buffering method used is of store and forward type. Control logic makes arbitration decisions. Thus communication is established between input and output ports. The connection or configuration is made between ports and central cross point matrix. According

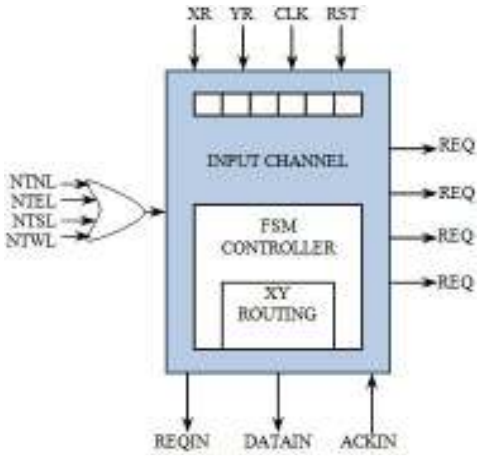



Fig. 2: Block diagram of input channel in the network-on-chip

to the destination path of data packet, control bit lines of cross point matrix are set. Directing the movement of data from source to destination is called switching mechanism. The packet switching mechanism has flit size of 8 bits. The packet size varies between 8 to 120 bits length. The process of an on-chip data routing network is called as Network-on-Chip (NoC) architecture (Kale and Gaikwad, 2011).

**Input channel:** The architecture of input channel at each port has its own control logic as shown in the Fig. 2. Each input channel has a First Input First Output (FIFO) of depth 16, data width of 8 bits and a control logic which is implemented based on the model of Finite State Machine (FSM). The input channel accepts request from other neighboring routers. On receiving the request, if it is free, it acknowledges the request.

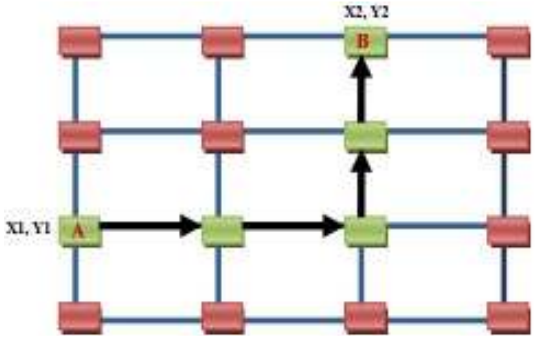



Fig. 3: Block diagram showing XY routing algorithm

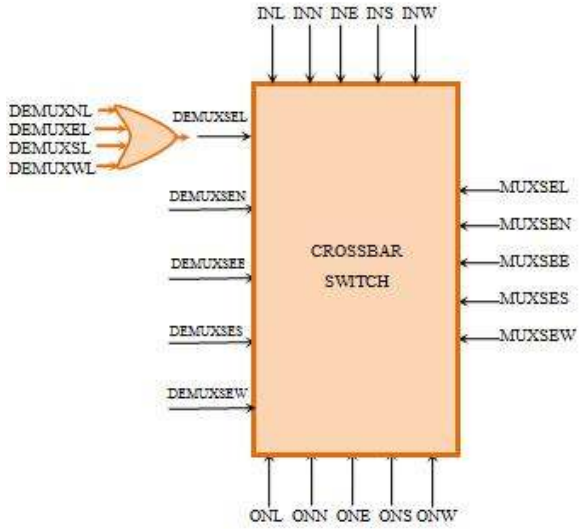



Fig. 4: Architecture of crossbar switch for network-on-chip router

The first flit is the header and the following flits constitute the data bits. It accepts the data as long as the request signal is held high. The requesting router's output channel ensures that the request line is held high until it empties the packet of data, being accepted by the input channel. The input channel keeps the acknowledgement line at high, as long as the transfer of data is continued which is indicated by the request line.

Once the transfer data is accomplished, the request and the acknowledgement lines go low in a sequence. The data packets received from the requesting router are stored locally in the FIFO for store and forward dataflow style. Next the control logic reads the header of the packet and decides which output channel is to be requested for sending out data packets from the router. On decision, it sends the request to identified output channel. It is to be noted that each of the input channel is running an independent FSM and hence can initiate five possible parallel connections simultaneously. Once the input channel gets a grant from the requested output channel, the control bits of cross point matrix are set accordingly to establish data flow process.

**Functioning of input channel:** The input channel starts functioning as soon as the input is given to it at *GNTL* (Fig. 2). Each and every input from the user is given to the MUX, the operation starts if there is a data strings, such as "00001001" at input and the acknowledgement line is held high corresponding to the request (*req*) made.

**XY routing:** At the input channel, once the FIFO is filled, the *X*-coordinate of the destination router (say for example $H_x$) is compared with locally stored *X*-coordinate of the router which first decides on the horizontal displacement as shown in the Fig. 3. If $H_x>X$ then the packet is forwarded to the east port of the router and if $H_x<X$ then the packets goes out through the west port of the router. If $H_x$ is equal to $X$ then the *Y*-coordinate of the router is to be decided on the vertical displacement. If $H_y>Y$ the packet is forwarded to the north port and if $H_y<Y$ the packet is forwarded to the south port. If $H_y$ equals $Y$, it indicates that the packet is at the destination router and thus the packet is forwarded to the local port. A packet is forwarded horizontally till the target column is reached and is then forwarded vertically to the destination router in a *XY* routing. This means that there is no request for the east or west output ports by the north or south ports. For this reason the FSMs of the mentioned output channels are simplified, as they need not to wait for the said input ports. This results in a significant reduction in both the area and number of clock cycles in serving requests. This helps the implementation of light weight router, having area overheads at the minimum with acceptable level of reasonable performance. The functioning of *XY*-routing may be summarized as follows:

- XY routing algorithm routes packets first in *x*-direction (horizontal) to the correct column and then in *y*-direction (vertical) to the receiver.
- Then the routing operation is done on the basis of the conditions of $X>Y$, $X = Y$ and $X<Y$.
- One of the advantages of *XY* routing is that it hardly enters into deadlock.
- The *XY* routing has addresses of the routers in terms of *XY* coordinates and is more suitable for networks based on mesh topology.

**Crossbar:** Crossbar switch consist of a set of multiplexers and de-multiplexers. These are interconnected in such a way that all possible connections between the five input and output channels are established if required. The crossbar switch architecture is shown in Fig. 4.

The output channel while granting the request to an input channel configures the multiplexers and de-multiplexers of available input and output channels.

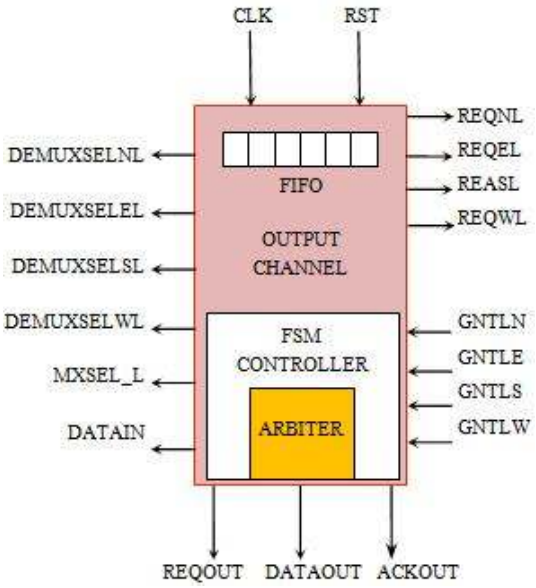



Fig. 5: General architecture of output channel of a router for network-on-chip

Subsequently the connections are established between channels for the transfer of the data packets. A crossbar switch, also known as cross-point switch or matrix switch, is a switch connecting multiple inputs to multiple outputs in a matrix format. The design of crossbar switch has 5 inputs and 5 outputs. Figure 4 shows multiplexer based crossbar switch. As there are five input packets of 40 bits each from five ports of router, the five numbers of 5:1 multiplexers are used by the crossbar shown in Fig. 4. All five inputs are connected to all the multiplexers. Select line is of 3 bits long. Out of five select lines available, which one is to be selected, depends upon the logic of arbiter. Outputs of multiplexers are the output ports of the 5×5 router (Zhizhou and Xiang, 2010).

**Output channel:** Depending upon the selection bits, the multiplexer works according to the request from the input channel. Further based on the input, the corresponding output will receive data from the crossbar. Each output channel at each port has an 8 bit FIFO of depth 16 and a control logic which makes arbitration decisions. The output channel gets request from different input channels and grants permission to anyone. It further sets the control bit lines of cross point matrix, as shown in the Fig. 5.

The output channel accepts the packet with the help of a simple decoding logic into its FIFO as long as the sending input FIFO is not empty. If the data transfer is completed, then the cross point matrix controls are reset. FSM then initiates the process to send the data into the neighboring router using handshake mechanism. Empty status of its FIFO initiates the next inter-channel transfer.

**Functioning of output channel:** The output from the arbiter is sent to the de-multiplexer block of output channel and correspondingly it enables the particular output, i.e., "00001001" data packet is received at output channel from the crossbar switch.

**Arbiter:** Arbiter, a main component of router, is used to control the packet switching mechanism. It has the ability to use an algorithm and to make switching based upon that algorithm (Yen-Lung *et al*., 2009; Zhizhou and Xiang, 2010). There are mainly two types of the arbitrations, Round Robin arbitration and matrix arbitration. System level arbiter architecture with IOs is shown in Fig. 6.

**Round robin arbitration:** Round-Robin (RR) is one of the simplest scheduling algorithms for processes in networks operation as shown in the Table 1. Round Robin method is self explanatory from the Table 1 and 2 as how the output is selected based on the data address and selection line value with or without content table as shown in Table 3.

It is generally used to operate the time slices, assigned to each process in circular order without any priority. It is easy to implement and can be applied to other scheduling applications such as data packet scheduling in computer networks etc., (Si and Mei, 2007; Zheng *et al*., 2002) (Fig. 7).

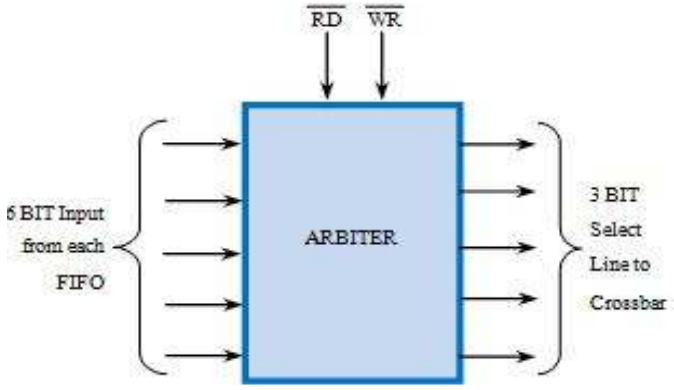



Fig. 6: System level architecture of arbiter

Table 1: Round robin arbitration without contention and with contention table

| I/P port | SA | DA | Select line | O/P port without contention | O/P port with contention |
|---|---|---|---|---|---|
| Port a | 000 | 001 | Selb (000) | Port b | Port b |
| Port b | 001 | 010 | Selc (001) | Port c | Port b |
| Port c | 010 | 011 | Seld (010) | Port d | Port b |
| Port d | 011 | 100 | Sele (011) | Port e | Port b |
| Port e | 100 | 000 | Sela (100) | Port a | Port b |

I/P: Input; SA: Selection address; DA: Destination address; O/P: Output; Sel: Select; a, b, c, d and e: Ports

Table 2: Matrix arbitration without contention table

| I/P port | SA | DA | Grant (4:0) | Select line | O/P port |
|---|---|---|---|---|---|
| Port a | 000 | 001 | Ma (01000) | Selb (000) | Port b |
| Port b | 001 | 010 | Mb (00100) | Selc (001) | Port c |
| Port c | 010 | 011 | Mc (00010) | Seld (010) | Port d |
| Port d | 011 | 100 | Md (00001) | Sele (011) | Port e |
| Port e | 100 | 000 | Me (10000) | Sela (100) | Port a |

I/P: Input; SA: Selection address; DA: Destination address; O/P: Output; M: Matrix; Sel: Select; a, b, c, d and e: Ports

Table 3: Matrix arbitration with contention table

| I/P port | SA | DA | Grant (4:0) | Select line | O/P port |
|---|---|---|---|---|---|
| Port a | 000 | 001 | Ma (01000) | Selb (000) | Port b |
| Port b | 001 | 001 | Mb (01000) | Selb (001) | Port b |
| Port c | 010 | 001 | Mc (01000) | Selb (010) | Port b |
| Port d | 011 | 001 | Md (01000) | Selb (011) | Port b |
| Port e | 100 | 001 | Me (01000) | Selb (100) | Port b |

I/P: Input; SA: Selection address; DA: Destination address; O/P: Output; M: Matrix; Sel: Select; a, b, c, d, and e: Ports

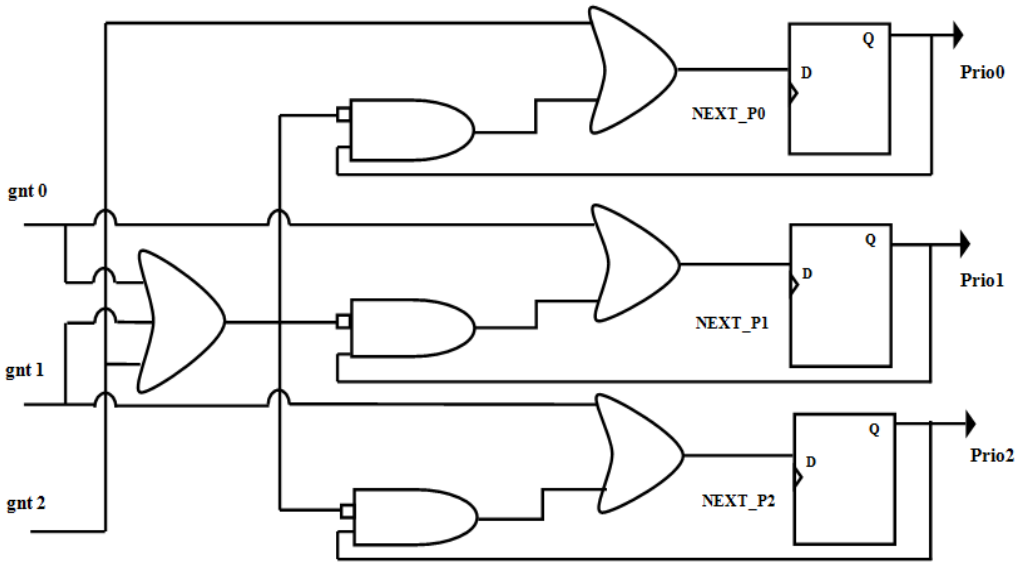



Fig. 7: Architecture of proposed round robin arbitration

**Matrix arbitration:** In matrix arbitration method, if all input packets have the same priority request for same output port, then matrix arbiter generate the matrix depending upon input and output port. In that matrix arbiter set the corresponding bit which is requested for same output port. Next matrix arbiter checks the priority. For example if input *a* has the highest priority and input e has the lowest priority then matrix arbiter gives priority to input *a* and input *e* gets lowest priority.

Matrix arbiter generates a control signal so that a particular line is selected and source packet is transmitted to destination port (Suyog and Gaikwad, 2012).

**Hamming code:** Hamming codes are linear error correcting codes. Hamming code can detect up to two bit errors and correct one bit error. These codes are a good choice for satellite communications as the most frequently occurring faults in on-board electronics are bit flips induced by radiation. By contrast, the simple parity code cannot correct errors. Hamming codes achieve the higher rate for codes. This study implements hamming distance in order to check the error free arbiter used in router.

**Calculation of the hamming code:** The parity check bits of each byte of the S-Box LUTs are pre-calculated.

Table 4: Simplified simulation flow for hamming codes algorithm

| Begin |
|---|
| Inputs: |
| X = 1, 2, 3, 4, 5, etc |
| Write the bit numbers in binary: 1, 10, 11, 100, 101, etc. All bit positions that are powers of two are parity bits determine: |
| Values of parity bit as 1, 2, 4, 8... |
| Change the position for parity bit 1 bit 1, 3, 5, 7, 9, etc. Change the position for parity bit 2 bit 3, 6, 7, 10, 11, etc. Repeat: |
| Until change of position for the last parity End |

Table 5: Device utilization: on-chip area used by the architecture of NoC router using matrix arbitration method

| Logic utilization | Used | Available | Utilization (%) |
|---|---|---|---|
| Total number slice registers | 1585 | 3840 | 41 |
| Number used as flip flops | 1580 | | |
| Number used as latches | 5 | | |
| Number of 4 input LUTs | 1225 | 3840 | 31 |
| Logic distribution | | | |
| Number of occupied slices | 1297 | 1920 | 67 |
| Number of slices containing only related logic | 1297 | 1297 | 100 |
| Number of slices containing unrelated logic | 0 | 1297 | 0 |
| Total number of 4 input LUTs | 1225 | 3840 | 31 |
| Number of bonded IQBs | 87 | 141 | 61 |
| Number of BUFGMUXs | 1 | 8 | 12 |

Table 6: Details of power utilization NOC router architecture using matrix arbitration

| Name | Power (W) | Used | Total available | Utilization (%) |
|---|---|---|---|---|
| Clocks | 0.762 | 1 | - | - |
| Logic | 0.354 | 1108 | 3840 | 28.9 |
| Signals | 0.234 | 2418 | - | - |
| IOs | 0.045 | 87 | 141 | 61.7 |
| Total quiescent power | 0.053 | | | |
| Total dynamic power | 1.516 | | | |
| Total power | 1.569 | | | |

Table 7: Device utilization summary: on-chip area used by architecture of NoC for round robin arbitration method

| Logic utilization | Used | Available | Utilization (%) |
|---|---|---|---|
| Number of slice flip flops | 1579 | 3840 | 41 |
| Number of 4 input LUTs | 1119 | 3840 | 29 |
| Logic distribution | | | |
| Number of occupied slices | 1242 | 1920 | 64 |
| Number of slices containing only related logic | 1242 | 1242 | 100 |
| Number of slices containing unrelated logic | 0 | 1242 | 0 |
| Total number of 4 input LUTs | 1119 | 3840 | 29 |
| Number of bonded IOBs | 87 | 141 | 61 |
| Number of BUFGMUXs | 1 | 8 | 12 |

Table 8: Details of power utilization of NOC router architecture using round robin arbitration

| Name | Power (W) | Used | Available | Total utilization (%) |
|---|---|---|---|---|
| Clocks | 0.765 | 6 | - | - |
| Logic | 0.242 | 1218 | 3840 | 31.7 |
| Signals | 0.224 | 2509 | - | - |
| IOs | 0.044 | 87 | 141 | 61.7 |
| Total quiescent power | 0.052 | | | |
| Total dynamic power | 1.395 | | | |
| Total power | 1.446 | | | |

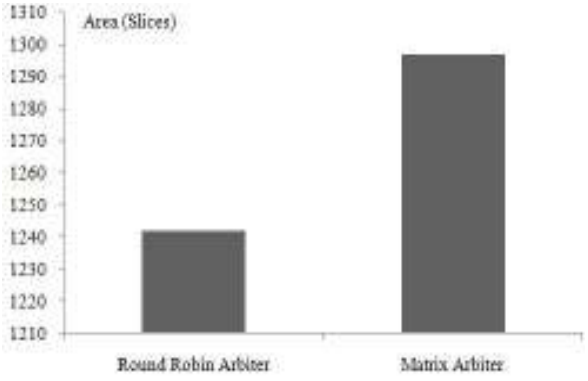
The Hamming code bits can be mathematically represented as follows:

$$h\,(S_{RD}[a]) \rightarrow h_{RD}[a]$$
$$h\,((S_{RD}[a] \otimes 2) \rightarrow h_{2RD}[a]$$
$$h\,((S_{RD}[a] \otimes 3) \rightarrow h_{3RD}[a]$$

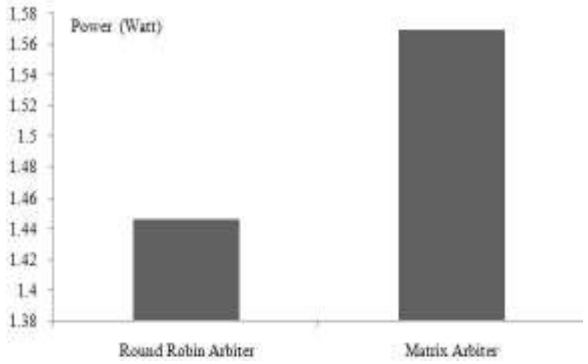
where, *a* is the state byte and *h* is the calculation of the Hamming code. Hamming codes algorithm steps are listed in Table 4 as shown above.

The synthesis report and simulated results of NoC router using matrix arbitration are listed in Table 5 and 6, respectively. The device utilization and power consumption statistics are listed. Similarly Table 7 and 8 show the corresponding statistics for round robin arbitration.

It can be summarized for conclusion that for NoC router implemented has input channel with a FIFO depth of 16, data width of 8 bits and a control logic implemented on FSG model as shown in Fig. 2. At the




(a)




(b)

Fig. 8: Performance analyses of area and power in NoC router using matrix and round robin arbitrations

input channel, once the FIFO is full, the XY algorithm begins for optimal routing of addresses of the routers. Here XY represents coordinates (Fig. 3). This routing technique is better choice for NoC networks compared to one using mesh or torus topology. The cross bar is a set of multiplexers and de-multiplexers having an interconnection allowing all possible combination of connections among the five input and output channels. The multiplexer here works based upon request from the input channel and based on the input itself. The corresponding output comes from the crossbar. Arbiter is one of the internal components used to control the packet switching which has ability to use the algorithms highlighted in the contention and without-contention tables of the arbitrations for switching. We used the Xilinx Spartan-3 board front end model, with a chip namely *xc3s200 FPGA* to implement functionality and subsequently verify the standalone router for the NoC system. We used the Xilinx 10.1ISE to synthesize the system and Model sim 6.3c to simulate the model and generate the activity data of the router model of both arbitrations, Round Robin and Matrix based. The hamming code is used for error free data transmission, the synthesized and simulated results of both arbitration designs are listed in Table 5 through Table 8.

Table 9: Comparison results of various parameters for NoC router using matrix and round robin arbitration methods

| Arbiter | Area (slices) | Power (w) |
|---|---|---|
| Matrix arbiter | 1297 (67%) | 1.569 |
| Round robin arbiter | 1242 (64%) | 1.446 |
| Improvement % | 3% | 8% |

The simulation results and analyses are discussed in the next sections (Fig. 8).

## RESULTS AND DISCUSSION

Table 7 is of indicative that implementation of NoC router using Round Robin Arbitration has utilized 1242 slices which is about 64% of total available slices. The power consumption statistics are listed in Table 8. Here, it can be observed that the power dissipation is about 1.446 W in case of Round Robin Arbitration. The Matrix Arbitration method has utilized 1297 slices and corresponding power dissipation is 1.569 W. Here the area is about 67%. Based on these observations, as listed in Table 9, it is obvious that a 3% improvement in area utilization and an 8% improvement in power dissipation are achieved by using Round Robin Arbitration method.

## CONCLUSION

This study presents an arbitration analyses between Matrix Arbiter and Round Robin Arbiters for router architecture of NoCs. The simulation result shows that the Round Robin Arbiter consumes less power and occupies smaller area in router architecture compare to Matrix Arbiter. The implantation of Round Robin Arbiter for the router architecture on front-end design on FPGA chip has the advantage of easy implementation and reconfigurable nature. It has provided reduction in utilization of area by a factor of 3% and power dissipation reduction by a factor of 8%. The power consumption is around 1.446 W in front end model design. This study also implements hamming distance in order to check the error free data transmission of the NoC router. The results and analyses on Hamming codes is not included in this study as it will form the base for further optimization of NoC routers. This design as proven to be better by a small but significant on area utilization and power dissipation parameters may further be explored with newer algorithms and different approaches for implementation.

## REFERENCES

Dally, W.J. and B. Towles, 2001. Route packets, not wires: On-chip- interconnection networks. Proceeding of Design Automation Conference, pp: 684-689.

Kale, A.S. and M.A. Gaikwad, 2011. Design and analysis of on-chip router for network on chip. Int. J. Comput. Trends Technol., 2(1).

Si, Q.Z. and Y. Mei, 2007. Algorithm-hardware code sign of fast parallel round-robin arbiters. IEEE T. Parall. Distr., 18(1).

Suyog, K.D. and M.A. Gaikwad, 2012. Design and analysis of matrix arbiter for NOC architecture. Int. J. Adv. Res. Comput. Sci. Electr. Eng., 1(5).

Yen-Lung, L., M.J. Jer and C. Yen-Yu, 2009. A high-speed and decentralized arbiter design for NOC [J]. Proceeding of the IEEE/ACS International Conference on Computer Systems and Applications (AICCSA, 2009), pp: 350-353.

Zheng, S.Q., Y. Mei, J. Blanton, P. Golla and D. Verchere, 2002. A simple and fast parallel round-robin arbiter for high-speed switch control and scheduling. Proceeding of the 45th Midwest Symposium on Circuits and Systems (MWSCAS-2002), 2: II-671-II-674.

Zhizhou, F. and L. Xiang, 2010. The design and implementation of arbiters for network-on-chips. Proceeding of the 2nd International Conference on Industrial and Information Systems.