VOL. 11, NO. 19, OCTOBER 2016 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences
©2006-2016 Asian Research Publishing Network (ARPN). All rights reserved.

FPGA IMPLEMENTATION OF ARBITERS ALGORITHM FOR NETWORK-ON-CHIP

T. Kavitha, S. Shiyamala and P. Nagarajan
Department of Electronics and Communication Engineering, School of Electrical and Computing Vel Tech University, Chennai, Tamilnadu, India
E-Mail: kavithaecephd@gmail.com

ABSTRACT

On-chip communication concept is the basic requirement for modern systems that offers high throughput. Network-on-chip routers provide essential routing functionality for effective global on -chip communication with low complexity and relatively high performance. Arbiters are used in No C router when number of input ports is request for the same output port. Recently, the Round Robin Arbiter and Matrix Arbiter are the basic building block for high speed switches / routers, receives a new attention with the advent of the Network-on-chip. In this paper, we compare the performance of these two arbiters in VLSI and tested in FPGA platform. Through the experiment result, we found that the resource utilization of Matrix arbiter is less compared to Round Robin Arbiter. For example for the input request 32 of the arbiter, the number of slices used for MA is 488 slices, for RRA is 2388 slices, Look Up Tab le mappings for MA is 921, for RRA is 2652 and the number of flip flops used for MA is 401 and the number of flip flops used for RRA is 1438. However the RRA has less frequency than the Matrix Arbiter. The performance evolution is achieved by allowing routing function for each input port, distributed arbiters and output port which gives high level of parallelism. When designing a network-on-chip arbiter, the trade off between the two mechanisms should be considered.

Keywords: network-on-chip, field programmable gate array, router, large scale integration.

1. INTRODUCTION

As the era of a billion transistors on a one chip approaches, a lot of Processing Elements (PEs) could be located on a System-on Chip (SoC). Most Processing elements within the SoC communicate with each other via buses and memory. Because the number of bus masters will increase in a single chip, the significance of fast and powerful arbiters commands additional consideration. Especially, a fast arbiter is one of the greatest main factors for high performance network switches. Also fast and efficient switch arbiters are required to switch packets in a Network-on-Chip (NoC) [1] [2]. But, fairness in arbitrations could be a very tedious and error-prone task for designers, to design with high performance. Network-on-Chip may be an advanced interconnection of several functional elements. It is the essential demand to deal with complexity of recent systems which is a general purpose on-chip communication idea that gives high throughput [3-5]. It iterates communication bottleneck within the gigabit communication because of its bus based architecture. So there was a necessity of systems that express modularity and parallelism. Network-on-Chip possesses several such attractive properties and solves the problem of communication bottleneck [6]. It essentially works on the concept of interconnection of cores using on chip network. The communication on NoC is carried out by means of router, so for implementing better NoC, the router should be efficiently designed. The switching mechanism used here is packet switching that is usually used on Network-on-Chip [7-9]. The information transfers in the form of packets between cooperating routers. In packet switching routing decision is taken independently. The store and forward flow mechanism is the most excellent method because it does not reserve channels and does not lead to idle physical channels. Due to simplicity and low overhead XY routing is used in the input channel. The arbiter used in NoC router is Round robin arbiter and Matrix arbiter. When more number of input ports are request the same output port, the network on chip router uses the arbiter. The arbiter should generate the grant signal on the basis of the number of input port receiving a priority and that input port transfers a packet to output port. A Network-on-Chip is one of the designs that includes Router, Input port, output port, Crossbar switch and arbiter design. Similarly, each and every component might comprise totally different specifications like variable bandwidth, buses and different communicative protocols. NoC is used to connect the server with the host [10-12].

The switches of NoC give high speed and cost effective contention resolution scheme when multiple packets from different ports compete for the same output port. One of the foremost dominant factors for high performance NoC switches is a fast arbiter [14]. The performance of the arbiters is analyzed and compared for the above reasons. In this chapter, the behaviors of two popular arbiters: the Round-robin and Matrix arbiter are analyzed

1.1 Router architecture

The set of ports in the router is used to communicate with the logic element such as local, east, west, north, and south as shown in Figure-1. It perceives the incoming packets and then forwards the packets into appropriate ports. The packets are temporarily stored in the buffers present at various ports. The routing decisions and arbitration decisions are performed by the control logic. The motivation here is to minimize the chip area that also reduces the power consumption. Store and forward buffering technology is used because it is the simplest decoding logic used to reduce both area and power. The connection establishment is created automatically without any complex decoding logic

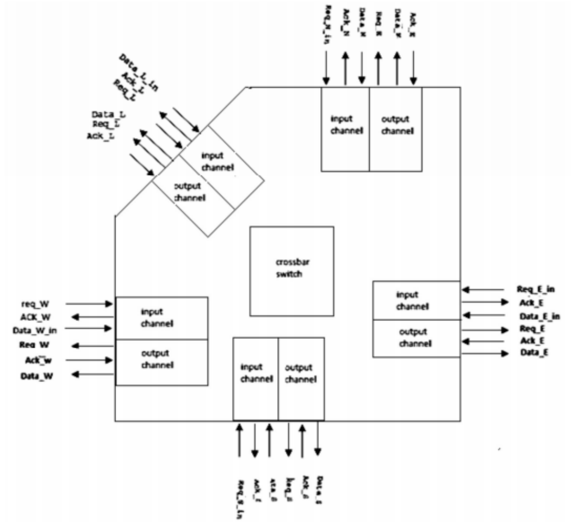

Figure-1. Router architecture.

The router switches with a set of inter-communicating ports that define the physical layer of the NoC system. There are two types of ports to establish communications, namely input and output ports. The communication is done by using two handshaking signals (Req and Ack) between the cooperating ports (input and output). The establishment of ports is done in the data link layer be handshaking signals. Dynamic establishment of connections and routing of packets constitute the network layer. Here the cross point matrix is the very important component, the control of which is maintained in the output channels. Inside the router, the Grant (Gnt) and Acknowledgement (Ack) signals are used to access the First In First out (FIFO) buffer without any need of explicit signals. The empty status signal of FIFO is used to indicate the end of communication.
Router has three main blocks, namely
Input channel
Cross point matrix, and
Output channel

2. INPUT CHANNEL

One input channel is found at each port, each input channel running its own control logic. Each input channel has a FIFO of depth 16 and data width of 8 bits and a control logic which is implemented as a Finite State Machine (FSM). The input channel accepts request from other neighboring routers. On receiving the request, it will acknowledge the request if it is free. The first flit is the header and following flits constitute the data. It will accept the data as long as the request signal and Ack signals are held high. The transfer data being completed, The Req and Ack lines go low in sequence. The packets of data received from the other routers are stored in the FIFO buffer. Next the control logic reads the header of the packet and decides which output the data is to be forwarded and then it sends the Req signal to the corresponding output channel. Each input channel has a separate FSM; therefore five parallel connections are possible at a time. Once the input channel gets a Gnt signal from the requested output channel, the control bits of cross point matrix are set appropriately by granting the output channel

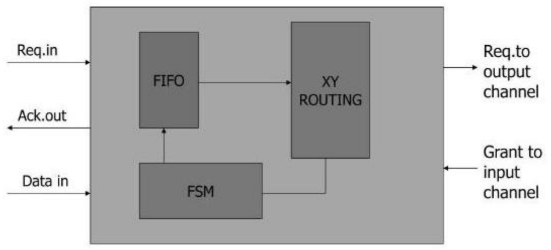

Figure-2. Input channel.

Simple logical OR function is used for optimization instead of multiplexer. Therefore area is minimized and performance is improved. At a time only one output channel is requested by the input channel. The inter data transfer is governed by the empty status of FIFO, by using this, complex decoding logic is eliminated. Empty condition automatically triggers the next data transfer. At the input channel, once the FIFO is full, the X coordinate of the destination router (say Hx) is compared with locally stored X coordinate of the router to decide the horizontal displacement.

At the Input channel, once the FIFO is filled, the X -coordinate of the destination router (say H_x) is compared with locally stored X coordinate of the router first to decide on the horizontal displacement. 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 to decide 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. When H_y equals Y it indicates that the packet is at the destination router and so the packet is forwarded to the local port.

A packet is forwarded horizontally t ill 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. This fact is exploited and the FSMs of the mentioned output channels are simplified as need not service the mentioned input ports. Translating to significant area saving and reduction in number of clock cycles in servicing requests is done by this. This helps the implementation of light weight router having area overheads at the minimum with acceptable level of performance.

3. CROSSPOINT MATRIX

Cross point matrix is the essential component in the design of router and it is placed within the central point of the router. It has group of multiplexers and demultiplexers. All five input channels and output channels are interconnected by using this cross point matrix. At the same time it allows all five connections by configuring the input and output channels. The cross point matrix is controlled by the output channel whereas granting the request of any input channel; output channel configures the multiplexers and demultiplexers of it. Therefore the connection is established between the channels for transfer of packets. Multiple inputs and multiple outputs are connected in a matrix manner by using a crossbar switch and also it is known as cross point switch or matrix switch. It has five inputs and five outputs. Figure-3 shows multiplexer based cross bar switch. Each input has 8 bits, so totally crossbar switch receives 40 bits from all inputs. Hence five number of 5:1 multiplexers are used inside the cross bar. All the inputs are fed to the multiplexers. Depending on the select bit one of the input is selected. That input sends the packets to the output.

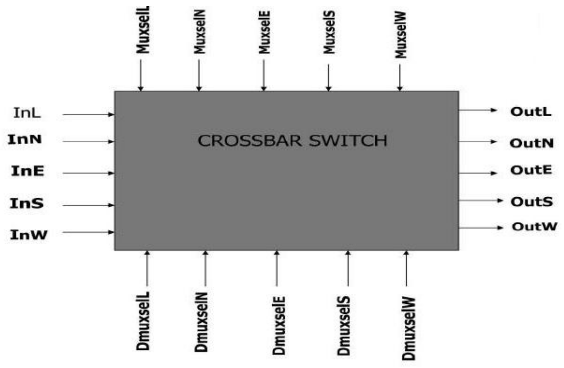

Figure-3. Crossbar switch.

4. OUTPUT CHANNEL

One output channel at each port has control and decoding logic. Output channel consists of FSM, FIFO and Arbiter as shown in Figure-4. The operation of FIFO and FSM are same as in input channel. Arbiter is used in output channel instead of using XY logic. The output channel gets request from the different input channels and grants one input channel and sets the control bit lines of cross point matrix. It accepts the packets and stores the packets in FIFO as long as the sending input FIFO is not empty thereby providing a simple decoding logic. The control lines in cross point matrix are reset when transfer is complete. Using handshake mechanism FSM initiates the process to send the packets into the neighboring router. When the FIFO buffer is empty it triggers the next data transfer.

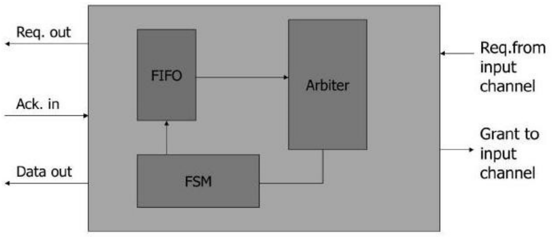

Figure-4. Output channel.

5. ARBITER

The Arbiter is a building block for many applications. In particular, it is a crucial building block for high-speed network switches / routers. Here we use the following two arbitration algorithms in the output channel and compare the performance of this two arbitration schemes

Round robin arbitration
Matrix arbitration

6. ROUND ROBIN ARBITRATION

Round-robin (RR) is one of the simplest scheduling algorithms for processes in networks operation .It is generally used to operate the time slices are assigned to each process in equal portions and in circular order, handling all processes without any priority (also known as cyclic executive). Round -robin scheduling is simple, easy to implement, and starving-free. Round-robin scheduling can also be applied to other scheduling problems, such as data packet scheduling in computer networks. Figure-5 shows the gate architecture of Round robin arbiter which realizes 3 bit roll arbitration mechanism.

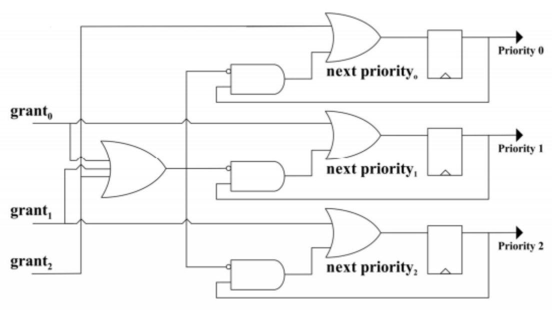

Figure-5. Architecture of round robin arbitration.

It is implemented as FSM at each output channel. RRA arbitrates and decides which input channel is to be given access to that output channel when many channels are requesting the same output. Generally the output channel must follow priority based arbitration. If a fixed priority scheme is followed the same input channel gets access repeatedly. Hence in our arbiter the priorities of the input ports are changed dynamically taking the last input port serviced into account. The priorities are implemented in clockwise fashion i.e., if the last input port serviced was north, then during Next service then the priority will be in the order of East, South, West, Local and North. It should be no clock cycles are wasted in our scheme as the grant is issued only if there is a request from corresponding input channel.

As each input channel has its own XY routing FSM and each output channel has its own RRA FSM, there is no latency in establishing the connections. This allows five different output channels. This provides a significant improvement in the performance o f our router. It is to be noted that the router coordinates are stored in two registers inside each of the router, which can be accessed from the primary inputs.

7. MATRIX ARBITRATION

In matrix arbitration when all input packet 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. If suppose the initial states in the upper triangle of the matrix arbiter are set to 1. According to the principle of the complementary, the elements in the lower triangle should be 0. In the original matrix arbiter, the request 1 has the highest priority, then request 2 and request 3 has lowest priority. Matrix Arbiter generates a control signal so particular select line is selected and source packet is transmitted to destination.

$$\begin{bmatrix} X & W_{1,2} & W_{1,3} & W_{1,4} \\ W_{2,1} & X & W_{2,3} & W_{2,4} \\ W_{3,1} & W_{3,2} & X & W_{3,4} \\ W_{4,1} & W_{4,2} & W_{4,3} & X \end{bmatrix}$$

Priority Matrix of Arbiter

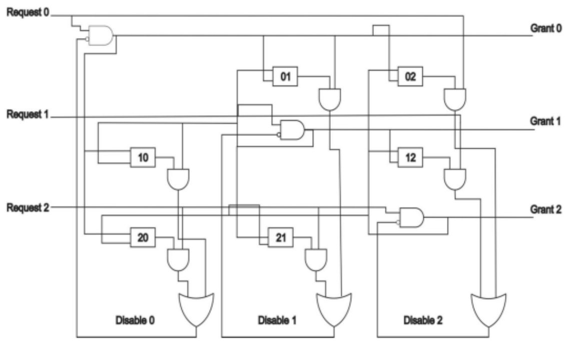

Figure-6. Architecture of matrix arbitration.

Figure-6 shows the three input gate architecture of a matrix arbiter. In the Figure each block numbered 01, 02 and 12 describes the S-R Latch, and the state is maintained in these S-R latches in the upper triangular portion of the matrix. Each of the blocks numbered 10, 20 and 21 in the lower triangular portion of the matrix represents the complementary output of the diagonally symmetric solid box.

8. RESULTS AND DISCUSSIONS

We use the Xilinx spartan3 board, which has a XCS400 FPGA to functionally verify the standalone router and the NoC system. We use the Xilinx 10.1[13] to synthesize the system and Modelsim 6.3[16] to simulate the model and generate the activity data of the place and router (PAR) model. The router is implemented in verilog HDL in a modular fashion. The data width and the FIFO depth are parameterizable. In this work the data size is fixed at 8 bit. We set different numbers of request inputs, which means the different lengths of request vectors. We get the statistics about the resource utilization, maximum clock frequency and delay of the two different arbitration mechanisms. Once the packets from the input channel simultaneously request the crossbar switch, the number of the request inputs of arbiter increased.

The number of slices used by the matrix arbiter and round robin arbiter is depicted in Figure-7. When there are a few requests, nearly 300 slices are consumed. When the number of input requests increases, Round robin arbiter will uses more resource. Matrix arbiter doesn't cost so much resource. When the request inputs approach 48, the round robin arbiter will utilize 2590 slices, while the matrix arbiter just uses 609 slices. We should make a decision between two arbiters while designing NoC router. The silicon areas of NoC chip will follow the same trend.

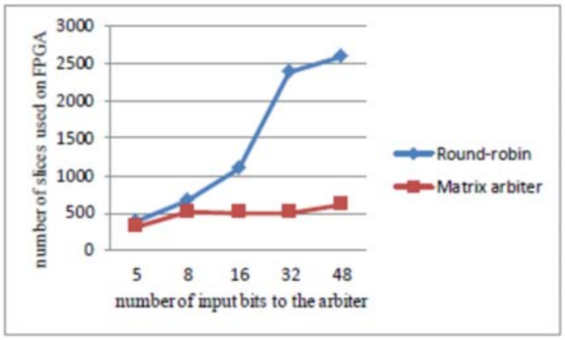

Figure-7. Number of slices used with number of bits to the arbiter.

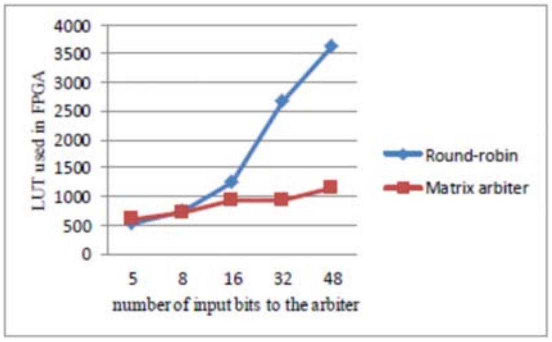

Figure-8. Number of LUT used with various input bits of arbiter.

Variation of look up table used in FPGA with the various input requests is drawn in Figure-8 for Matrix arbiter and Round robin arbiter. The look up table (LUT) mechanisms for matrix arbiter and round robin arbiter is almost in the same level for the input requests 5 and 8. When the input request increases the look up table mechanisms for matrix arbiter is less compared to the Round robin arbiter

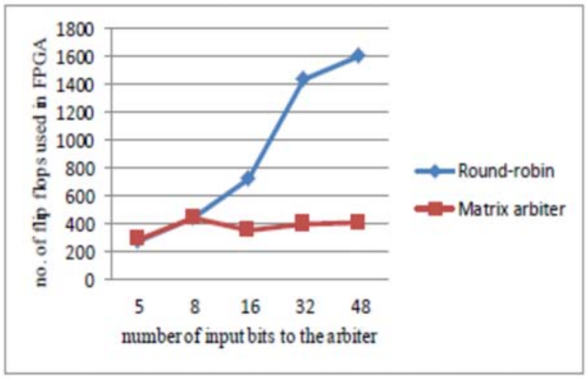

Figure-9. Number of flip flops used with various input bits of arbiter.

Variation of Flip flops used with respect to number of input requests is presented in Figure-9. From the figure it is seen that the Matrix arbiter uses less number of flip flops than the Round robin arbiter. For example, for the input request 32, the number of flip flop used for matrix arbiter is 401, and Round robin arbiter uses 1438 flip flops.

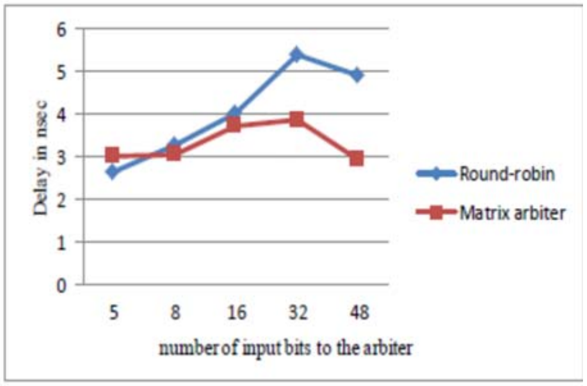

Figure-10. Variation of delay in milliseconds with various input bits of arbiter.

We analyze and compare the delay of the two mechanisms. In Figure-10, we can see that the delay will increase as the number of input request increases. The matrix arbiter consumes less delay than the Round robin arbiter. When the input request is 32, the matrix arbiter has a delay of 3.861 nanosec and the round robin arbiter has the delay of 5.397 nanosec.

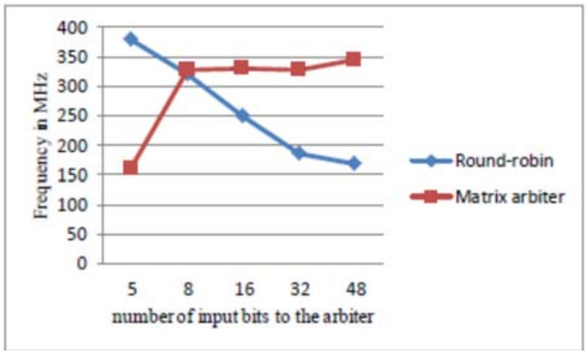

Figure-11. The frequency changed as different input bits of arbiter.

Figure-11 describes how the maximum clock frequency changes as the increase of the number of the inputs in these two mechanism arbiters. Matrix arbiter has higher clock frequency than round robin arbiter because round robin arbiter consumes mo re resources than the matrix arbiter. When the number of input request is 8, the two mechanisms can reach the same clock frequency, approximately 320 MHz. Almost, the maximum clock frequency of matrix arbiter is far higher than that of the Round robin arbiter, means the matrix-arbiter has higher throughput and more fast computation speed.

9. CONCLUSION AND FUTURE RESEARCH

In this paper we emulated the performance of Round robin arbiter and Matrix arbiter on FPGA platform. We compared the performance of these two arbiter in terms o f number of resources utilized, clock frequency and delay. Round robin arbiter consumes more resources, which mean it utilizes more silicon area. Matrix arbiter has high clock frequency than the round robin arbiter, which means matrix arbiter could process data more quickly. In the design of arbiter, we should make a trade-off among the resource or silicon area, maximum clock frequency and delay and choose suitable arbitration mechanism according to that. This work can be further carried out to build an advanced prototype supporting High Level Protocol (HLP) having less area overhead [15]-[17].

REFERENCES

[1] Mr. SuyogK.Dahule and Dr. M.A.Gaikwad. 2012. Design and Analysis of Matrix Arbiter for NoC Architecture. International Journal of Advanced Research in Computer Science and Electronics Engineering. 1: 100-103.

[2] Yun-Lung Lee, Jer Min Jou; Yen-Yu Chen. 2009. A high-speed and decentralized arbiter design for NoC. IEEE/ACS International Conference on Computer Systems and Applications. pp. 350-353.

[3] Zheng S. Q., Yang M. 2007. Algorithm- Hardware Codesign of Fast Parallel Round-Robin Arbiters.

IEEE Transactions on Parallel and Distributed Systems. 18: 84-95.

[4] G. De Micheli, C. Seiculescu_, S. MuraliL. Benin, F. Angiolini, A. Pullini. 2010. Networks on Chips:from Research to Products. IEEE Design Automation conference. pp. 13-18.

[5] A. Agarwal, Cyril Iskander and Ravi Shankar. 2009. Survey of Network on Chip (NoC) Architectures and Contributions. Journal of Engineering. pp. 1-15.

[6] Wen-Chung Tsai, Ying-CherngLan, Yu-Hen Hu and Sao-JieChe. 2012. Networks on Chips: Structure and Design Methodologies. Journal of Electrical and Computer Engineering. pp. 1-15.

[7] Paul Gratz, Karthikeyan Sankaralingam. 2007. On-chip Interconnection Networks of the Trips Chip. IEEE Computer Society. pp. 41-50.

[8] Jer-Min Jou and Yun- Lung Lee. 2010. An Optimal Round-Robin Arbiter Design for NoC. Journal of Information Science and Engineering. pp. 2047-2058.

[9] Naveen Choudhary. 2012. Bursty Communication Performance Analys is of Network-on-Chip withDiverse Traffic Permutations. International Journal of Soft Computing and Engineering. Vol. 1.

[10] Roman Gindin, Israel Cidon, IditKeidar. 2007. NoC-Based FPGA: Architecture and Routing. IEEE International conference on networks-on-chip. pp. 253-264.

[11] Ms. A.S. Kale, Prof. M.A.Gaikwad. 2011. Design and Analysis of On-Chip Router for Network on Chip. International Journal of Computer Trends and Technology. pp. 182-186.

[12] Villeantala, Teijo Lehtonen Juha Plosila. 2006. Network on Chip Routing Algorithms. TUCS Technical Report No 779, pp. 1-38.

[13] Xilinx Inc. http://www.xilinx.com.

[14] Sh. Moazzeni, S. Pour Mozafari and A. Emami,

[15] 2011. A Novel Design of a Network on Chip Input Buffer Using GALS Technique. Canadian Journal on Electrical and Electronics Engineering. 2(2).

[16] R. Kiruthika, Dr.T. Kavitha and V. Aravinda Rajan. 2015. Clock Gating Optimization Technique Using Buffer Gates. International Journal of Applied Engineering Research (IJAER). 10(20): 17940-17944.

[17] MentorGraphicsInchttp://www.Mentorgraphics.com.

[18] Nagarajan P, Saravanan R and Thirumurugan P. 2014. Design of register element for low power clocking system. Information-An International Interdisciplinary Journal.