International Journal of Computer Applications Technology and Research
Volume 4– Issue 8, 623 - 628, 2015, ISSN: 2319–8656

Internal Architecture of Junction Based Router

Tulikapriya Sinha
Symbiosis Institute of Technology,
Symbiosis International University,
Pune, India

Shraddha Patil
Symbiosis Institute of Technology,
Symbiosis International University,
Pune, India

Smita Khole
Symbiosis Institute of Technology,
Symbiosis International University,
Pune, India

Abstract: The router is an important component in NoC as it provides routes for the communication between different cores. A router consists of registers, switches, arbitration and control logic that collectively implement the routing and flow control function required to buffer and forward flits to their destination. This router will be implemented on FPGA using Spartan-3 kit. This paper describes the internal blocks of a junction based router and there operation.

Keywords: Router, NoC, FPGA, Verilog, arbiter.

1. INTRODUCTION

1.1Network on Chip (NoC)

Network on chip or network on a chip is a communication subsystem on an integrated circuit (commonly called a "chip"), typically between IP cores in a System on a Chip (SoC). Network-on-Chip (NoC) architectures provide a good way of realizing efficient interconnections and largely alleviate the limitations of bus-based solutions. NoCs can span synchronous and asynchronous clock domains or use un-clocked asynchronous logic. NoC technology applies networking theory and methods to on-conventional bus and crossbar interconnections. NoC improves the scalability of SoCs (System on Chips), and the power efficiency of complex SoCs compared to other designs.[1]

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 viewpoint. The wires occupy most of the area on the chip, interconnects dominate both performance and dynamic power dissipation, as signal propagation in wires across the chip requires multiple clock cycles. NoC links can reduce the complexity of designing wires for predictable speed, power, noise, reliability, etc., thanks to their regular, well controlled structure. From a system design viewpoint, with the advent of multi-core processor systems, a network is a natural architectural choice. A NoC can provide separation between computation and communication, support modularity and IP reuse via standard interfaces, handle synchronization issues, serve as a platform for system test, and, hence, increase engineering productivity. The whole router design can be implemented on a FPGA (Field programmable gate arrays).

There are three main components of NoC.

- Resource
- Resource Network Interface (RNI)
- Router

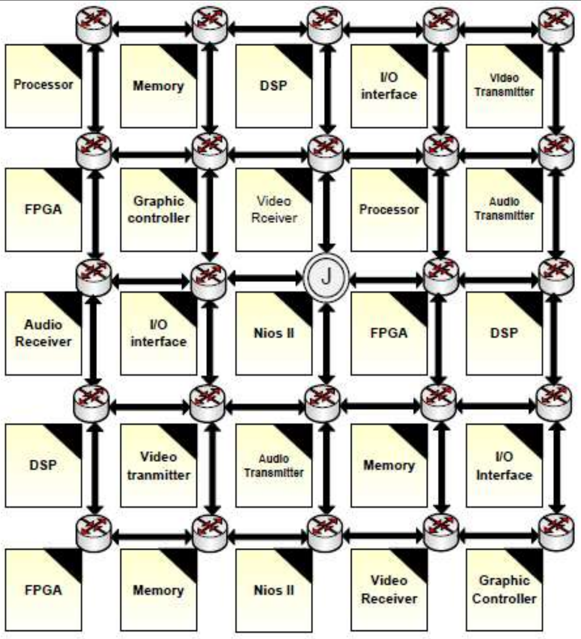


Figure 1 Example of Network on Chip platform[1]

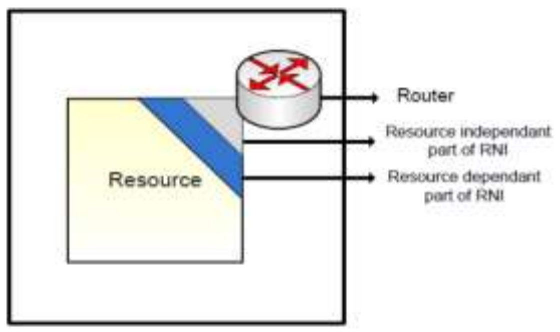

Router
Resource independant part of RNI
Resource dependant part of RNI
Resource

Figure 2: NoC Components [1]

A Core is connected to the router through RNI. RNI is the interface between router and the core. Core sends packet to the router through RNI, and from router it is send to the other routers or core depending on the destination.

Benefits of NoC Architecture:

- Independent implementation and optimization of layers.
- Simplified customization per application.
- Supports multiple topologies and options for different parts of the network.
- Simplified feature development, interface interoperability, and scalability.

1.2 Design for Junction based routing

Routing in NoC can be classified in many ways. Router design depends on the routing protocol and routing algorithm used. [1]
Two kinds of routing algorithms are source routing and distributed routing.
In source routing, path to be followed by the flit is locked from source to destination. The complete route information is available in head flit. Here, router takes decision by looking at the head flit. Path tables are stored inside the resource network interface (RNI). These path tables contain the complete path information for a specific destination in the network. Path information is calculated by applying routing algorithms.
In distributed routing, routing decisions are taken inside every router on the path and hence is a complex design due to extra hardware. There is no information about the path inside the header packet.
Compared to distributed routing, source routing has speed advantage because the routing information is stored in the packet itself. But source routing leads to overhead to store complete path information in the header of each packet.
An algorithm called Junction based source routing was developed to overcome these flaws. Junction Based Source Routing limits the required path information to be stored in every packet to a small number of bits which correspond to only a few hops as shown in figure 3. There are temporary destinations called junctions to cover the large distance such that sub-paths are always smaller than or equal to a maximum hop count. If a packet needs to go through a junction, the source just appends path information from source to the junction. On reaching the junction, the packet picks up path information to reach the destination from this junction.
The design of a NoC router depends on various aspects of NoC architecture and the performance requirement. The Junction Router contains path information in tables to reach any destination. The table can implement either in the router itself or in the resource network interface (RNI) or in the resource (core). There are three distinct cases for a packet to reach a junction:
i. If the destination of the packet is the resource connected to the Junction itself, then it should be routed to the resource through RNI.
ii. If the destination is not very far and the packet header has enough information to reach the destination, then the router forwards the packet just by looking at the relevant field in the header.
iii. If the destination is far, then the junction will be the intermediate destination. This will be clear from the relevant field in the header. In this case, Junction modifies the path information in the packet header for onward journey to the destination, if possible otherwise to another junction as intermediate destination. [1]

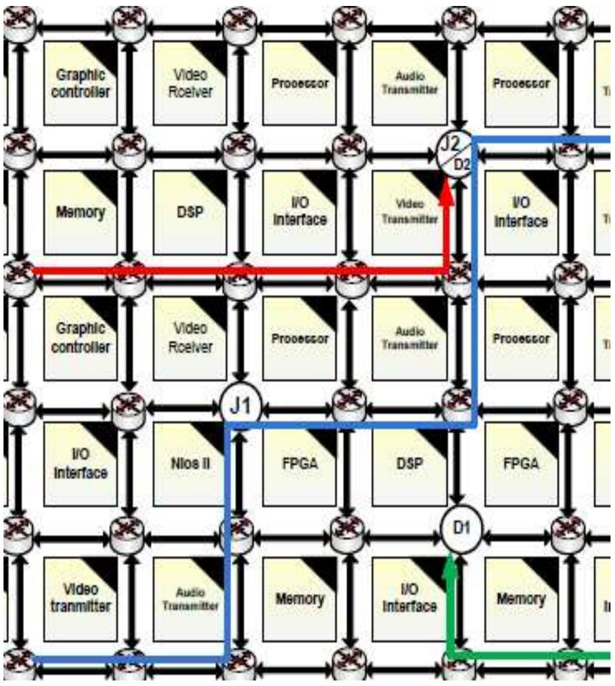


Fig.3 Junction Based Network on Chip based system [1]

The router architecture is a typical Network on Chip router. Design of a router depends on the routing algorithm used. Here junction based resource router has been described. This router architecture supports both kinds of routers which mean that the router can be used as a normal router as well as a junction router. Main components in the router architecture are as follow:

i. Arbiter and Control
ii. Crossbar
iii. Input Handler [1].

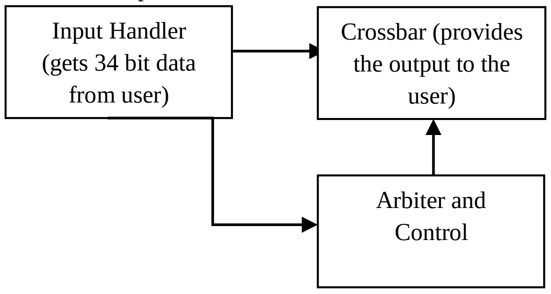

Input Handler (gets 34 bit data from user)
Crossbar (provides the output to the user)
Arbiter and Control

Figure 4: Main components of Router

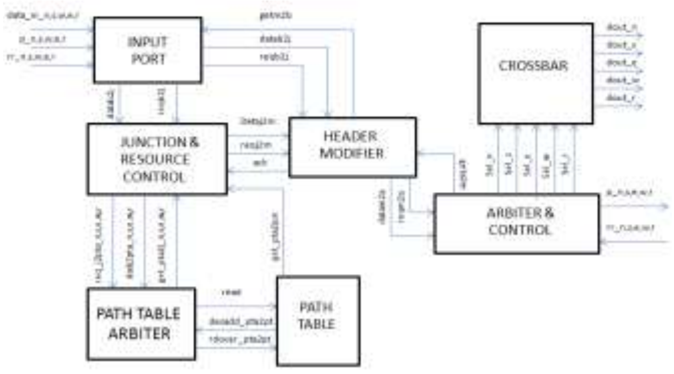


Figure 5: Block Diagram depicting all blocks of Junction based Router

1.3 Flit Format:

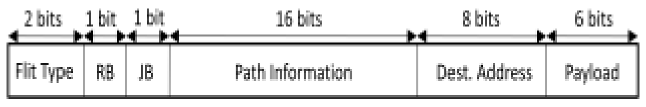


Figure 6: Flit Format.[1]

RB- Resource Bit.
JB- Junction Bit.
Dest. Address- Destination Address
The above diagram represents the Flit Format or the distribution of bits of the 34 bit data.
Flit type:
00- Head Flit.
01- Body Flit.
10- End Flit.
11- Full Flit.

Head Flit:

When the Flit type is "00", its Head Flit. This flit is initially sent to lock a particular path for transmission of data. This contains the destination address, direction bits, etc.

Body Flit:

When the Flit type is "01", its Body Flit. This flit follows the path reserved by the Head flit. It contains the main data which needs to be transmitted.

End Flit:

When the Flit type is"10", its End flit. When this flit is sent, it means that the data has been sent and no more data needs to be transmitted. This indicates end of transmission of data.

Full Flit:

When the Flit type is "11", its full flit. When this flit is sent, the path which was locked for transmission of data is unlocked for next data to be transmitted.

2. DESCRIPTION OF COMPONENTS PRESENT IN ROUTER:

2.1 Input Handler:

This block receives the data and stores it temporarily in a stack present. By using a stack of width 34bits and depth 4bits. The outputs are sent to different parts of Crossbar and Arbiter .This block consists of 3 subparts:

i. Input Port
ii. Header Modifier
iii. Junction and Resource Control.

2.1.1. Input Port

- The main function of this block is to store the incoming flits and send them to header modifier block.
- For the storing the data temporarily, stack is used, which operates on 'First In First Out' (FIFO) concept.
- There are many temporary registers used to control the stack
- As the data is sent into the router, it becomes the data for this block, signal is generated and the data is written into the stack.
- Stack consists of a depth of 4, which means it can save maximum of 4 data at a time. When the stack is full, other data go into the waiting state for stack to empty at least one row.

2.1.2 Header Modifier

- When the block receives request signal from the input port, data is sent from the port.
- When the data is received, flits are checked first. If the flits are Head flit or Full flit, then the RB (Resource Bit) is checked.
- If Resource Bit=1, then wait for request signal from junction to the Header modifier. When the signal is received, along with this the data from Junction and Resource control which is of 34bits is also received.
- First 4 bits of Header Modifier is sent to the Arbiter, these become the inputs of Arbiter block, which is discussed later.
- The 4 bits of Header Modifier are divided in following manner:
  - ➢ 2 bits for Flit type, first two bits.
  - ➢ 2 bits for direction i.e. from where the data is coming, next two bits.
- The entire 34bit data is sent to the Crossbar as its input from this block.
- If RB=0, then concatenation operation of data is performed. Here the path information is modified, first two bits are sent behind the last 2 bits of path information. This modified data is then sent to the crossbar as 34bits data and 4 bit data is sent to the Arbiter and control block.
- If the Flit type is body flit or end flit, data in the modifier is directly sent to the crossbar and the 4 bit data is sent to the Arbiter and control block.

2.1.3 Junction and Resource Control

The main function of this block is to receive flits from input port, send destination address to Path Table Arbiter Component and receive new path information , To send flit with new path information to Header Modifier.

- When the block receives request signal from the input port, data is sent from the port.
- As soon as the data is received , flit type is checked. If the flit type is Head or Full type , then it checks for Resource Bit condition.
- If resource bit is 1, then it loads the data in destination address which goes from junction to path table arbiter else it goes into waiting state till it receives the address with resource bit as 1 .
- Further, if junction and resource control block is getting signal from path table arbiter then this block gets new path information from path table.
- Junction and resource control concatenates the new path information such that the first two bits are sent behind the last 2 bits of path information.
- If the flits are Body or End type, then it goes into waiting state till it receives Head or Full as flit type.

2.2 Arbiter and control

2.2.1 Path Table Arbiter

- The main function of this block is to handle all the new path information requests and forward them to the path table, where the new path information is generated.

- Firstly, there is a request signal from Junction and Resource Control block. Run an FSM to set priority if data is coming from more than one direction:
  - North -------- First
  - South ------- Second
  - West -------- Third
  - East --------- Fourth
  - Resource -- Last

  Resource is given lowest priority to avoid entrance of a new flit when present data is being processed.
- As the request is accepted to be serviced, the destination address is forwarded to the path table to obtain new path information. Simultaneously, a read signal is generated and forwarded to the path table.
- The block remains in waiting state until a signal is obtained from the path table.
- As soon as this signal is obtained, signal is generated and sent to Junction Resource Control block.
- After this, the block enters the initial state of scanning requests.

2.2.2 Path Table

- Main function of this block is to forward the new path information to the Junction and Resource Controller.
- For this purpose, there is a new path information created in the initial steps, input of destination address from the Path Table Arbiter, payload from the Input Port and an FSM is run to decide the direction bits i.e. 001 for north, 010 for south, 011 for west and 100 for east.
- The above three are concatenated and stored in a temporary register of 17 bits length.
- This data is then written into a stack. Its operation is similar to that of stack used in Input Port.
- At positive transition of clock, the path info is concatenated and written in the fifo,
- As soon as a read signal is received, the data is sent to the Junction and Resource Control block. After this, a signal is sent to the Path Table Arbiter indicating the completion of the task.

2.2.3 Arbiter using Round RobinAlgorithm:

Arbiter & Control is considered as the Brain of the router because of its following functions described below:

- ➢ handles request for output direction.
- ➢ takes decision for output direction by decoding the Head flit.
- ➢ locking and unlocking the path.
- ➢ checks the space in the buffer of next router.
- ➢ To send signals for selecting the output direction.
- ➢ To send signal to the next router or resource to save data in buffer. [1]

Arbiter and Control makes use of Round robin Algorithm to assign priority to directions/ routing of the data. There are 5 directions which needs to be considered while making this router, North, South, West, East and Resource.

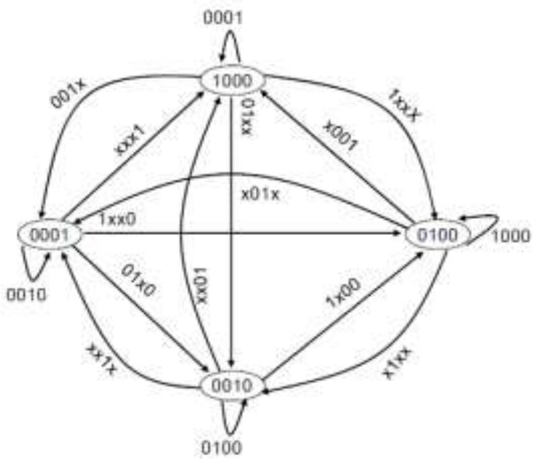


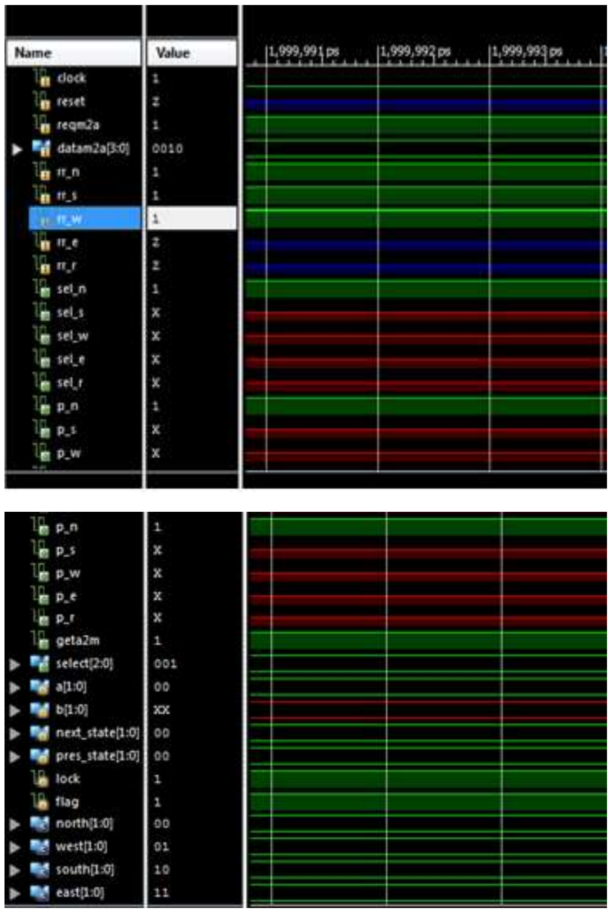
Figure7: FSM for using Round Robin Algorithm[2]

North- 1000
West- 0100
South- 0010
East- 0001

While using this algorithm assume a 3*3 mesh, resource will be the 5th router or the centre router, as discussed diagram above. Hence resource direction has not been taken into considerations. By using round Robin algorithm each state checks for all signals from all directions and then passes the signal according to their priority.
Highest priority is given to north, then west, south and east. The direction into consideration is anti-clockwise.
Arbiter works in following way:

- When it receives the signal from Header Modifier which is a 4 bit data, the first 2 bits are used for checking Flit type.
  - ➢ If the Flit type is head, it means that arbiter needs to assign a direction for the flow of data i.e. lock the path for data.
  - ➢ If the flit type is body or end, data follows the path assigned/ the path which is locked.
  - ➢ If the flit type is full, it means that the data transmission is over and the path should be unlocked.
- It checks from which direction Arbiter has received the signal and accordingly assigns the next state. Directions are assigned and used to avoid congestion of Data.
- When the direction the data is in the present state then the path is locked. Locking of path takes place only for flit type head.
- Along with this select is also assigned some value of 3bits. These values are given to crossbar for deciding the direction through which the output is expected.
  For example if the Present State is 'North' then select= 001 and select line for north=1, others are assigned as 0.

- This cycle continues till the data does not reach destination.



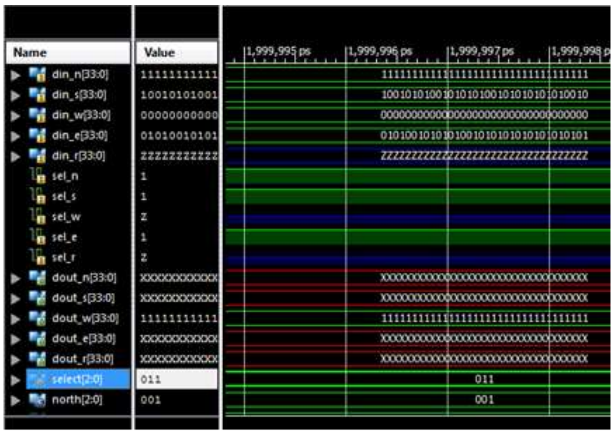
Name	Value
clock	1
reset	z
reqm2a	1
datam2a[3:0]	0010
rr_n	1
rr_s	1
rr_w	1
rr_e	z
rr_r	z
sel_n	1
sel_s	x
sel_w	x
sel_e	x
sel_r	x
p_n	1
p_s	x
p_w	x
p_e	x
p_r	x
geta2m	1
select[2:0]	001
a[1:0]	00
b[1:0]	xx
next_state[1:0]	00
pres_state[1:0]	00
lock	1
flag	1
north[1:0]	00
west[1:0]	01
south[1:0]	10
east[1:0]	11

Figure 8: Simulation Result of Arbiter. Flit type is Head, North is given higher priority hence select=001 and, signal for north goes high(p_n) and select line for north (sel_n) goes high.

2.3 Crossbar

The main function of the crossbar is to send the data from one path to the assigned path.

- Select lines are assigned for choosing the direction or path of data flow.
- There are 5 select lines for 5 directions which are controlled by Arbiter.
- When the select line is known then data is collected from the Header Modifier and sent out through the same line.
- For example, if select=001, which means North state and in this state if west select line is high, then data is collected from modifier and sent as data out from west direction.



Name	Value
din_n[33:0]	11111111111
din_s[33:0]	10010101001
din_w[33:0]	00000000000
din_e[33:0]	01010010101
din_r[33:0]	zzzzzzzzzzz
sel_n	1
sel_s	1
sel_w	z
sel_e	1
sel_r	z
dout_n[33:0]	xxxxxxxxxxx
dout_s[33:0]	xxxxxxxxxxx
dout_w[33:0]	11111111111
dout_e[33:0]	xxxxxxxxxxx
dout_r[33:0]	xxxxxxxxxxx
select[2:0]	011
north[2:0]	001

Figure 9 : Simulation report of Crossbar

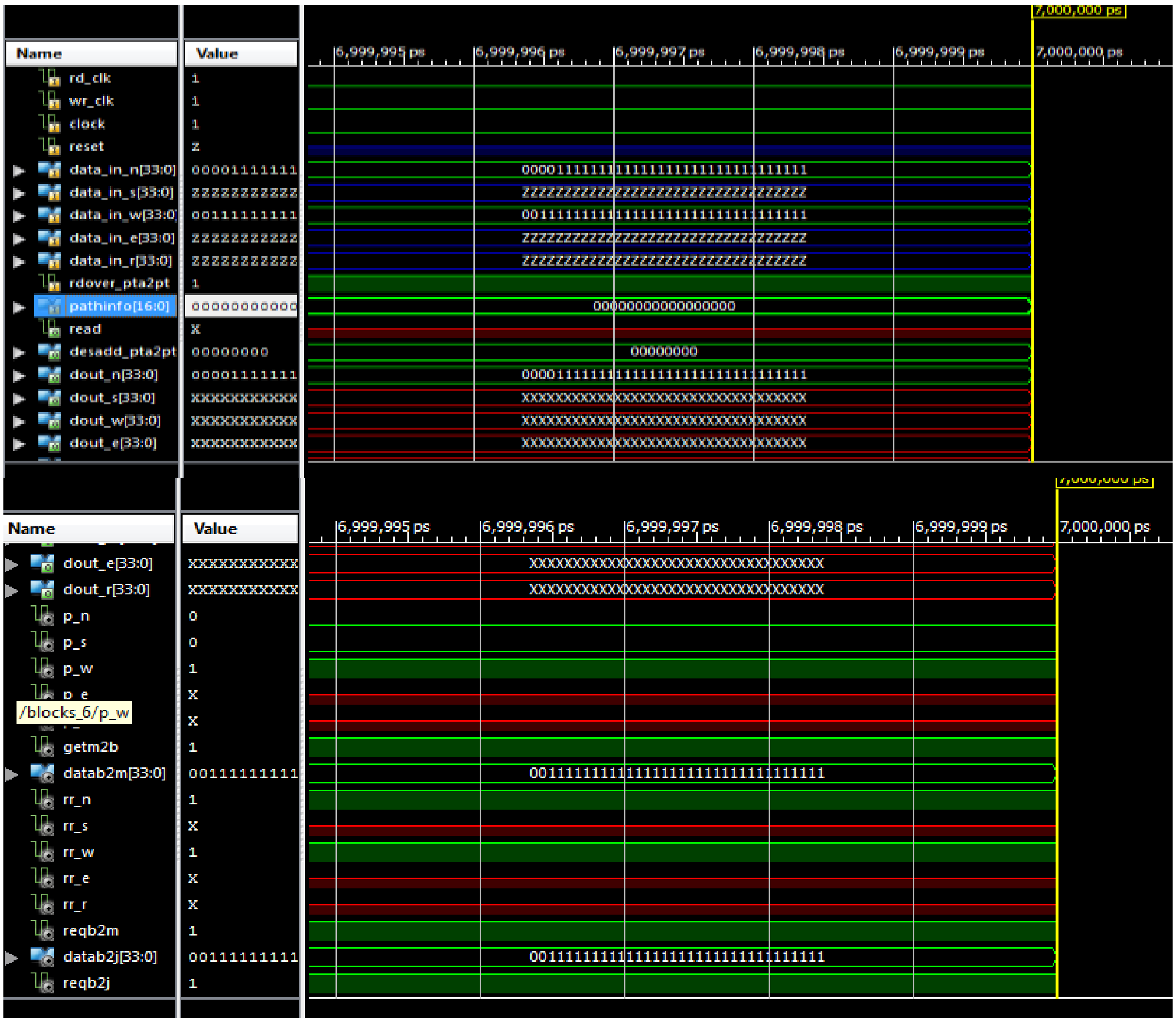
2.4 Interfacing of the blocks

Verilog coding is usually done in segments and later a number of smaller segments is integrated together to give a common output. The process of interfacing involves the following mentioned steps:

- Writing the codes individually.
- Creating a top module and calling the above programs as instances in the top module.
- Instantiating a program is calling the code is a defined format, that is, module ‘name’ ‘instance’(ports).
- After all the codes are instantiated and added in the top module group and a common simulation is checked.

2.5 Conclusions

This work presented the implementation of router which is divided into small blocks input port, header modifier , junction and resource control , path table, path table arbiter, crossbar and arbiter and control. The blocks were implemented through Verilog codes using ISE XILINX version 14.6 software. The simulation facilitates clear understanding of the functionality of each block in the router for a network on chip.
Six blocks have been interfaced satisfactorily which includes input port, header modifier, junction and resource control, path table arbiter, arbiter and crossbar.
Interfacing of the 7th block, Path Table is in the process.



Name	Value
rd_clk	1
wr_clk	1
clock	1
reset	z
data_in_n[33:0]	00001111111
data_in_s[33:0]	zzzzzzzzzzz
data_in_w[33:0]	00111111111
data_in_e[33:0]	zzzzzzzzzzz
data_in_r[33:0]	zzzzzzzzzzz
rdover_pta2pt	1
pathinfo[16:0]	00000000000
read	x
desadd_pta2pt	00000000
dout_n[33:0]	00001111111
dout_s[33:0]	xxxxxxxxxxx
dout_w[33:0]	xxxxxxxxxxx
dout_e[33:0]	xxxxxxxxxxx

Name	Value
dout_e[33:0]	xxxxxxxxxxx
dout_r[33:0]	xxxxxxxxxxx
p_n	0
p_s	0
p_w	1
p_e	x
/blocks_6/p_w	x
getm2b	1
datab2m[33:0]	00111111111
rr_n	1
rr_s	x
rr_w	1
rr_e	x
rr_r	x
reqb2m	1
datab2j[33:0]	00111111111
reqb2j	1

Figure 10: Simulation Report of 6 blocks(Input port, Header Modifier, Junction and Resource Control, Path Table Arbiter, Arbiter and Crossbar) have been interfaced together. The inputs are given to data_in pin in any direction required and the output is obtained from data_out in the same direction as the input given.

3. ACKNOWLEDGMENTS

We would like to thank our mentor Ms. Priti Shahne for her continuous support and giving us the opportunity to work on this project.

4. REFERENCES

1. Muhammad Awais Aslam,” Router Architecture for Junction Based source routing: design and FPGA Prototyping”, Tekniska Hogskolan, 2011.
2. Jer-Min Jou and Yun-Lung Lee,” An Optimal Round-Robin Arbiter Design for NoC”, Journal of Information Science and Engineering,2010.
3. Tobias Bjerregaard and Shankar Mahadevan,” A Survey of Research and Practices of Network-on-Chip”, ACM computing surveys, 2006.
4. Ville Rantala, Teijo Lehtonen and Juha Plosila, “Network on Chip Routing Algorithms”, TUCS technical Report, 2006.
5. Stephen L. Chamberlin,” Design and implementation of Router using Xilinx FPGA”, Massachusetts Institute of Technology.
6. Sunil, Shaik Khadar Sharif, 3praveen Vanaparthy, “Fpga Implementation of Five Port Router Network”, International Journal of Engineering Development and Research.
7. Adrijean Adriahantenaina, Hervé Charlery, Alain Greiner, Laurent Mortiez, Cesar Albenes Zeferino, “SPIN: a Scalable, Packet Switched, On-chip Micro-network”.
8. Samir Palnitkar, “Verilog- HDL, a guide to Digital and Synthesis”.
9. J. Duato, S. Yalamanchili, and L. Ni, Interconnection Networks: An Engineering Approach. IEEE Computer Society Press, 1997.