**1. INTRODUCTION**

Network on chip or network on a chip (NoC) is a communication subsystem on an integrated chip, typically between IP cores in a System on Chip. NoC technology applies networking theory and methods to on chip communications and brings notable improvements over the conventional bus and crossbar interconnections. NoC improves the scalability of system on chips, and the power efficiency of complex system on chips compared to other designs. A packet in NoC is injected by the following cases:-

1) L1,L2 Cache misses.

2) Coherence Transactions.

3) Synchronisation.

L1,L2 cache miss occurs when a processing core is not able to find the required memory item in its L1 cache or its shared L2 cache. In this case the processing core tries to find the item in the shared memory of other nodes in the network. Coherence transactions are required when more than 1 node holds the value of a variable from a different node , and one of the nodes updates the value of variable , then this value has to be updated with all other nodes holding this value. Synchronisation between two nodes is required when a node will be using a value which is generated by another node , so both these nodes have to be synchronised.

Network on chip is an emerging paradigm for communications within large VLSI systems implemented on a single Silicon Chip. A NOC is constructed from multiple point-point data links interconnected by switches, such that messages can be relayed from any source module to any destination module over several links, by making routing decisions at the switches.

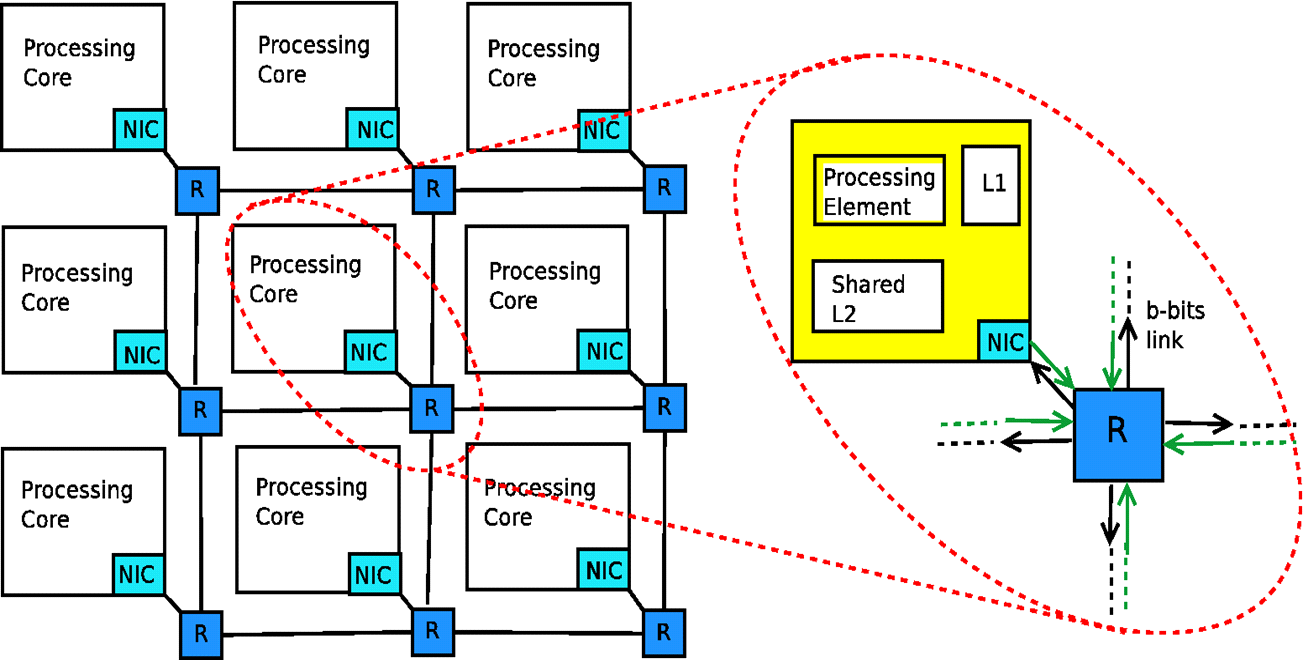


Fig 1. NoC Architecture

To meet the growing computation-intensive applications and the needs of low-power, high-performance systems, the number of computing resources in single-chip has enormously increased, because current VLSI technology can support such an extensive integration of transistors. By adding many computing resources such as CPU, DSP, specific IPs, etc to build a system in System-on-Chip, its interconnection between each other becomes another challenging issue. In most System-on-Chip applications, a shared bus interconnection is required. However, such shared bus interconnection has some limitation in its scalability because only one master at a time can utilize the bus which means all the bus accesses should be serialized by the arbitrator. Therefore, in such an environment where the number of bus requesters is large and their required bandwidth for interconnection is more than the current bus, some other interconnection methods should be considered.

A router is a device that forwards data packets between computer networks. This creates an overlay internetwork, as a router is connected to two or more data lines from different networks. When a data packet comes in one of the lines, the router reads the address information in the packet to determine its ultimate destination. Then, using information in its routing table or routing policy, it directs the packet to the next network on its journey. Routers perform the "traffic directing" functions. A data packet is typically forwarded from one router to another through the networks that constitute the internetwork until it reaches its destination node.

The wires in the links of the NOC are shared by many signals. A high level of parallelism is achieved, because all links in the NOC can operate simultaneously on different data packets. Therefore, as the complexity of integrated systems keeps growing, a NOC provides enhanced performance such as throughput and scalability in comparison with previous communication architectures,e.g.,dedicated point-to-point signal wires, shared buses, or segmented buses with bridges.

**1.1 Deadlock**

A deadlock is a situation in which two or more competing actions are each waiting for the other to finish, and thus neither ever does.Deadlock is a common problem in multiprocessing systems, parallel computing and distributed systems. A deadlock situation can arise if a few nodes in the system are buffer full and when another node which is buffer full is not able to send because there is no free buffer. This situation occurs in a circular situation as shown below.



Fig 2. Deadlock

**1.2 Livelock**

A livelock is similar to a deadlock, except that the states of the processes involved in the livelock constantly change with regard to one another, none progressing. Livelock is a special case of resource starvation. Livelock is a risk with some algorithms that detect and recover from deadlock eg. buffer less routing. If more than one process takes action, the deadlock detection algorithm can be repeatedly triggered. This can be avoided by ensuring that only one process takes action.

**2. LITERATURE SURVEY**

Routing is the process of selecting best paths in a network. In the past, the term routing was also used to mean forwarding network traffic among networks. However this latter function is much better described as simply forwarding. Routing is performed for many kinds of networks, including the telephone network, circuit switching, electronic data networks such as the Internet, and transportation networks.

Routing algorithms can be classified according to the three different criteria: (i) where the routing decisions are taken.

(ii) how a path is defined

(iii) the path length.

According to where routing decisions are taken, it is possible to classify the routing in source and distributed routing. In source routing, the whole path is decided at the source switch, while in distributed routing each switch receives a packet and defines the direction to send it. In source routing, the header of the packet has to carry all the routing information, increasing the packet size . In distributed routing, the path can be chosen as a function of the network instantaneous traffic conditions. Distributed routing can also take into account faulty paths, resulting in fault tolerant algorithms.

Depending how a path is defined, routing can be classified as deterministic or adaptive. In deterministic routing, the path is completely specified from the relative position of source and target addresses. In adaptive routing, the path is a function of the network instantaneous traffic . Adaptive routing increases the number of possible paths usable by a packet to arrive to its destination. However, deadlock and livelock situations can happen in fully adaptive algorithms , which limit its usage.

Regarding the path length criterion, routing can be minimal or nonminimal . Minimal routing algorithms guarantee shortest paths between source and target addresses. In nonminimal routing, the packet can follow any available path between source and target. Nonminimal routing offers great flexibility in terms of possible paths, but can lead to livelock situations and increase the latency to deliver the packet.

**2.1 General Routing Algorithms**

**2.1.1 XY Routing**

This is an routing algorithm that can be established on network on chip for transmission of packets from source to destination. It follows a simple approach where in the sender first sends the packet and it moves in horizontal direction (x-direction) and reaches a point which is either the destination or a node that lies in the same column where the destination node resides, then the packet traverses in the vertical direction (y-direction) until it reaches the destination.

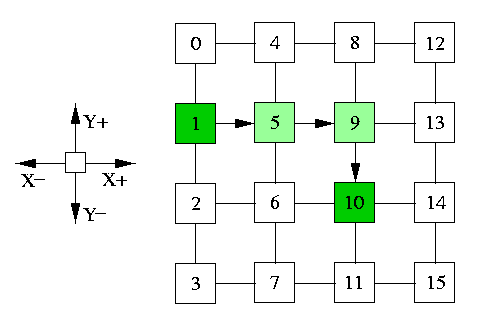


Fig 3. X-Y Routing

**2.1.2 Odd-Even Routing**

The four sides of a 2-D mesh are labelled as North, South, East and West. All the nodes that have the same coordinates of dimension 0 constitute a column, and all the nodes that have the same coordinates of dimension 1 constitute a row. In 2D mesh is called an even (respectively,odd) if the dimension-0 coordinate of the column is an even (respectively,odd) number.

Rule 1: Any packet is not allowed to take an EN turn at any nodes located in an even column, and it is not allowed to take an NW turn at any nodes located in an odd column.

Rule 2: Any packet is not allowed to take an ES turn at any nodes located in an even column, and it is not allowed to take an SW turn at any node located in an odd column.

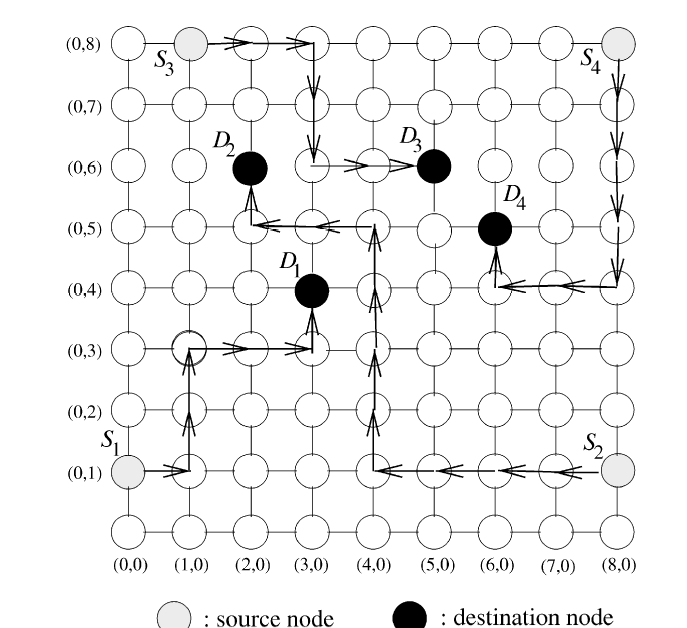


Fig 4. Odd-Even Routing

**2.1.3 Source Routing**

When source routing is used in a NoC , each core contains a table that includes complete route information to reach all the other cores in the network to which it needs to communicate.

Inorder to route a packet through the network, a sender resource looks up the table and adds complete path from source to destination in the packet header.Each router reads the path field in the packet header and forwards it to the destined output port.

Source routing is not perhaps suitable for dynamic networks where network size and topology are changing. But in a NoC with fixed size and regular topology like mesh, the path information can be efficiently encoded with small number of bits.

Since the packet entering a router contains the pre-computed decision about the output port, the router design is significantly simplified. Since NoCs used in embedded systems are expected to be application specific, we can have a good profile of the communication traffic in the network.

This allows us to analyze the traffic and compute efficient paths giving the desired performance properties, like uniform link load distribution. Source routing also provides possibility of mixing minimal and non-minimal paths for this purpose.

Advantages

* Simpler,smaller and faster router design.
* Topology independence.
* Network Size Independent Router.
* Guaranteed throughput.
* In-order delivery of packets.

Disadvantages

* Routing overhead
* Static and Non adaptive nature of source routing
* Limitation of size of source table

**3. HARDWARE AND SOFTWARE SPECIFICATIONS**

**3.1. Hardware Specifications**

Processor : Pentium III or Above.

Main Memory : 512 MB RAM.

Cache Memory : 512 KB.

Operating Frequency : 1 GHz.

Hard Disk Capacity : 4GB or Above.

Monitor : SVGA Color

Speaker : NIL

Microphone : NIL

**3.2. Software Specifications**

Operating System : Windows XP/Windows 7/8(for Xilinx),

Ubuntu 12.0 (for BookSim simulator)

Software Required: BookSim 2.0, Xilinx 14.1

**4 .SYSTEM DESIGN AND IMPLEMENTATION**

The project focuses on the implementation of source routing for Network on Chip(NoC). Analysis of X-Y routing algorithm gave us the insight that the same path is taken for the same source and destination independent of the traffic. So the time taken for processing can be reduced if the path is appended to the flit. This makes the process of routing simpler and more efficient.

Therefore to implement source routing we append an additional field to each packet which contains the path taken by the packet. The field has the following structure.

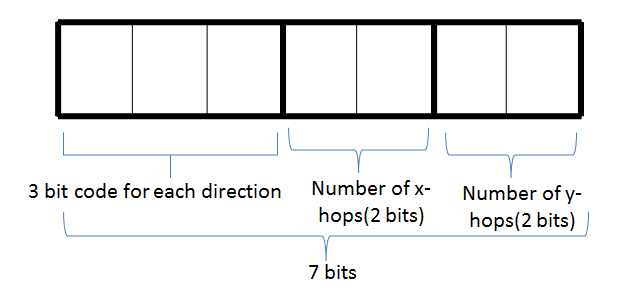


Fig 5. Packet Header for proposed Source Routing

The destination of the packet is assigned a three bit direction value depending upon the orientation of the source and destination. The code may take values from 000 to 111 which may represent the directions East, East North, North, West North, West, West, West South, South and East South respectively.

The next four bits represent the number of x-hops and y–hops respectively. The number of x-hops is the number of movements made in the x direction while travelling to the destination. The number of y-hops is the number of movements made in the y-direction while travelling to the destination.

Each router in the implementation of source routing can operate in any of the two modes, mode 0 and mode1.

Mode 0 is the case when the node is the source. That is two operations take place.

* The path taken by the node is computed and is inserted into the flit.
* The path is read from the flit and the corresponding next router is taken.

Mode 1 is the case when the node is not the source. Only one operation takes place, which is reading the path from the packet and taking the correct path.

Whenever a packet reaches a router, the router processes the packet and produces an output which is a port number to indicate the corresponding router.

**4.1 Pseudo Code for Source routing on NoC**

**4.1.1 Psedocode for mode 0**

Initialize the string variable str to 0 //used to store the path

1. Use the variable e to store the value of the destination router

2. Use the variable cur to store the value of the current router.

3. If the value of e[3:2] > cur[3:2] & e[1:0] > cur[1:0], assign 1 to quadrature value.//

value of x and y co-ordinate of the destination are greater

4. If the value of e[3:2] > cur[3:2] & e[1:0] < cur[1:0], assign 3 to quadrature value.//

value of y coordinate is greater and x coordinate is lesser

5. If the value of e[3:2] < cur[3:2] & e[1:0] > cur[1:0], assign 7 to quadrature value.//

value of y coordinate is lesser and x coordinate is greater

6. If the value of e[3:2] < cur[3:2] & e[1:0] < cur[1:0], assign 5 to quadrature value.//

value of y coordinate and x coordinate is lesser

7. If the value of e[3:2] > cur[3:2] & e[1:0] = cur[1:0], assign 2 to quadrature value.//

value of y coordinate is greater and x coordinates are equal

8. If the value of e[3:2] < cur[3:2] & e[1:0] = cur[1:0], assign 6 to quadrature value.//

value of y coordinate is lesser and x coordinate are equal

9. If the value of e[3:2] = cur[3:2] & e[1:0] > cur[1:0], assign 0 to quadrature value.//

value of y are equal and x coordinate is greater

10. If the value of e[3:2] = cur[3:2] & e[1:0] < cur[1:0], assign 3 to quadrature value.//

value of y coordinates are equal and x coordinate is lesser

// Quadrature value obtained, now to obtain the value of x hops and y hops

11. Find the difference between e[3:2] and cur[3:2] to find y hops.

12. Find the difference between e[1:0] and cur[1:0] to find x hops.

13. Assign the values to the string.

**4.1.2 Pseudo code for Mode 1.**

1. If the number of x-hops > 0,
   1. If the quadrature value is 1, 7 or 0

Return output port as 1. // port number for East.

Reduce the number of x hops.

Else,

Return output port as 3. // port number for West.

Reduce the number of x hops.

1. If the number of y hops > 0,
   1. If the quadrature value is 1, 2 or 3

Return output port as 0 // port number for North.

Reduce the number of y hops.

Else,

Return output port as 2// port number for South.

Reduce the number of y hops.

1. When the value of x hops and y hops are 0, return output port as 4

// Local port

**4.1.3 Packet Header when source is 0 and destination is 10.**

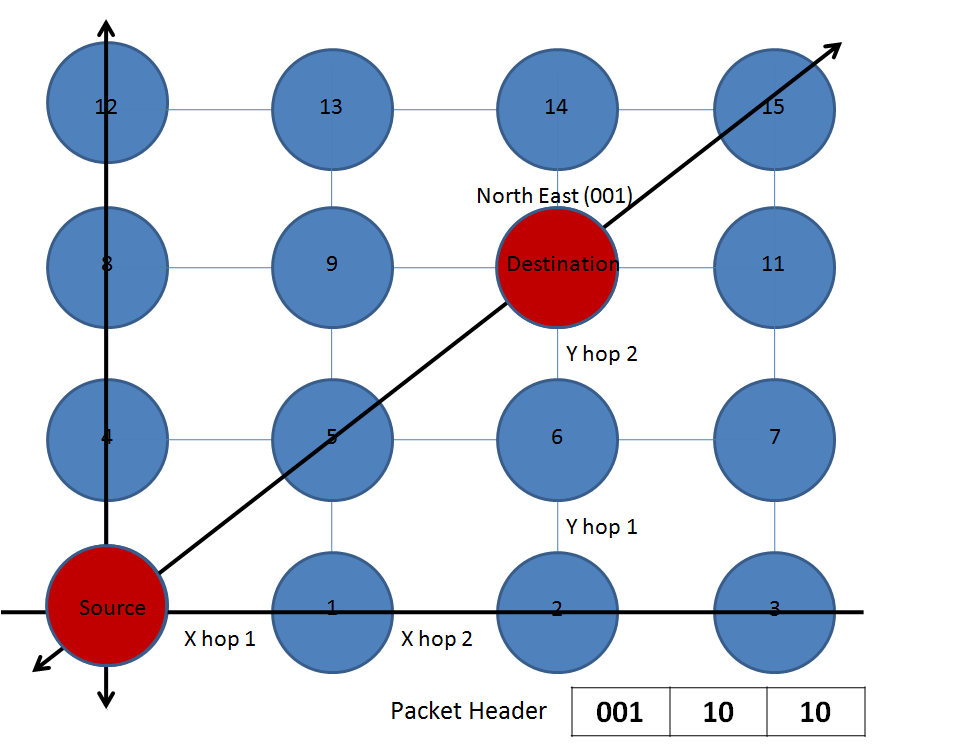


Fig 6. Example 1 for Packet Header Generation

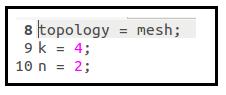
**4.1.4 Packet header when source is 2 and destination is 14.**

Fig 7. Example 2 for Packet Header Generation

**5. TESTING STRATEGIES AND RESULTS**

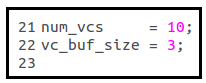
The results of the network system was obtained using the simulator known as BookSim 2.0 and the following are some of the parameters of the network at the time simulation.

a) Topology



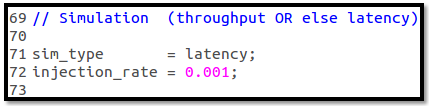
The topology of the network is set as mesh and the mesh is set as 4x4.

b) Virtual channel and Buffer size



The number of virtual channels for each router is set as 10 and the buffer size of each virtual channel is 3.

c) Injection Rate



The Injection rate specifies the amount of packets injected into the network in one clock cycle. Here we set the injection rate as 0.001 .

The BookSim simulator returns various parameters such as latency, throughput etc. On running the distributed direction order routing algorithm we obtained the result as follows.

Fig 8. Booksim Output for Distributed Routing

Here we obtained the overall average latency as 15.1429 .

On running the Source routing algorithm we obtained the result as follows.

Fig 9. Booksim result for Source Routing

Here also we obtained the overall average latency as 15.1429 .

The above latency obtained is based on the terms of average number of clock cycles. But since it was evident that there would be a considerable timing difference the similar logic was implemented in a Hardware Description Language called Verilog. The results obtained from Verilog were as follows.

Fig 10 . Time Delay comparison using Verilog

The time delay obtained for running Node routing in a router is 6.472ns. The time delay for running Mode0 of Source routing in a router is obtained as 1.451ns and for running Mode1 of Source routing in a router is obtained as 2.665ns. The result of time delay in Source routing is 58.82% lesser than Distributed routing.

**5.1 Bar Graph Analysis**

**5.1.1 Graph 1**

Fig 11. Time Delay for a Packet

The above graph shows the comparison between the time delay for a packet in node routing and the source routing. The booksim results gave the overall average latency as 15.1429 clock cycles. The verilog results gave the time delay for the node routing and source routing as 6.472ns and 2.665ns respectively. So if we calculate the time delay for a packet, we will get the time delay for node routing as 97.08ns (15.1429 \* 6.472) and source routing as 39.975ns(15.1429\*2.665). On comparison it can be seen that there is a reduction of 57.105ns (97.08-39.95ns) in time delay between the node routing and the proposed source routing.

**5 .1.2 Graph 2**

Fig 12. Node Routing vs Source Routing

The above graph shows the comparison between the time delay for a packet to traverse from the source to the destination in a 4\*4 mesh considering the maximum hop scenario which is equal to 6 in case of a 4\*4 mesh. The time delay for the node routing can be calculated using the verilog results.The verilog showed a time delay of 6.472ns for a packet in node routing. This delay multiplied by the number of clocks per cycles, which in this case is equal to 3(one for routing unit, one for switch and virtual channel allocation and one for link traversal) which in turn multiplied by 6(that is the maximum number of hops in a 4\*4 mesh) gives the total time delay as 116.49ns. The time delay value for mode 0 and mode 1 showed by verilog is 1.451ns and 2.665ns respectively. In order to calculate the time delay for a packet to reach from source to destination in source routing we have to perform the following calculation:

Time Delay for Source Routing = [(1.451) + (2.665\*6\*3)]ns =49.421ns

Mode 0 delay= 1.451ns

Mode 1 delay= 2.665ns

Maximum number of hops=6

Time delay Reduction = [(116.49-49.421)/116.49]\*100= 57.57% compared to Node routing in a 4\*4mesh.

**5.1.3 Graph 3**

Fig 13. Time Delay comparison in a 4X4 and 8X8 mesh

The above graph shows the comparison between the time delay for a 4X4 mesh and an 8X8 mesh. The time delay reduction for a 4X4 mesh is 57.57%.

For a 8x8 mesh, the maximum number of hops will be 14. Thus the time delay can be calculated using the following computation:

Time Delay for 8x8 mesh using source routing= [1.451 +(2.665)\*14\*3]ns= 113.381ns

Mode 0 delay= 1.451ns

Mode 1 delay= 2.665ns

Maximum number of hops=14

Time delay Reduction = [(241.824-113.381)/241.824]\*100= 53.11% compared to Node routing in a 8\*8 mesh.

**5.2 Hardware Design for Distributed Routing**

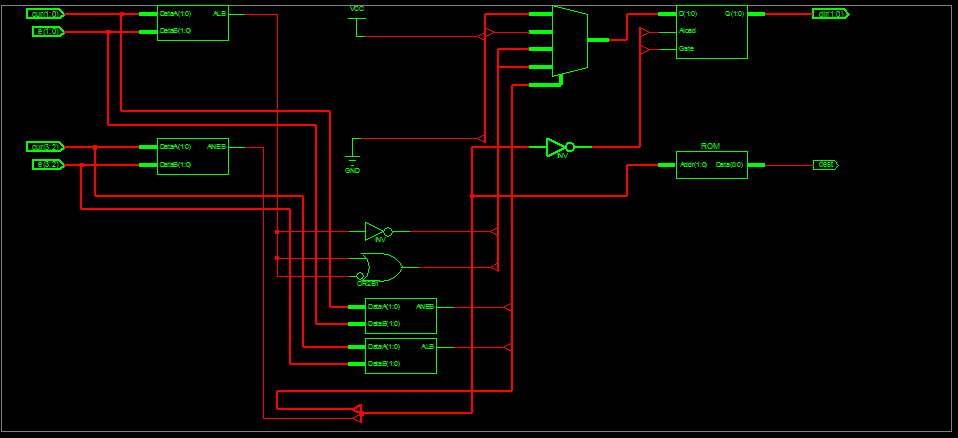


Fig 14. Hardware Design for Distributed Routing

**5.3 Hardware design for Source routing :Mode 0**

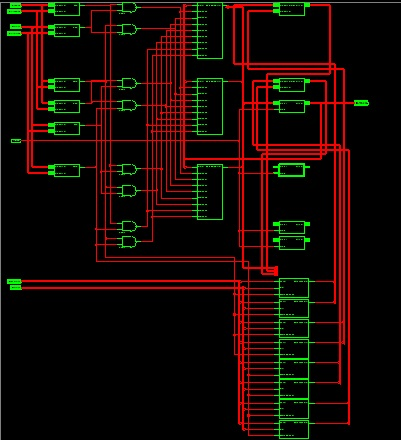


Fig 15. Hardware design for Source routing: Mode 0

**5.4 Hardware Design for Source routing : Mode 1**

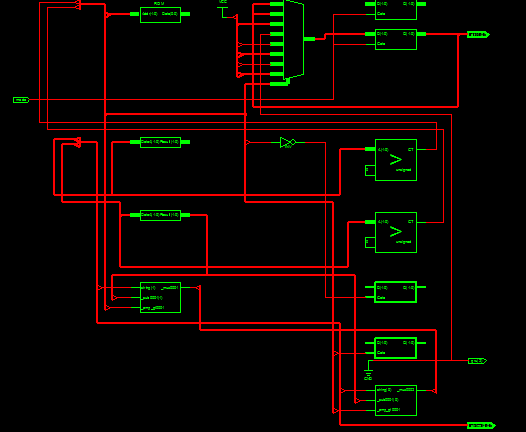


Fig 16. Hardware design for Source routing: Mode 1

**5.5 Pipeline Diagram for Source and Distributed Routing**

Fig 17. Pipeline diagram for source and distributed routing

In an NoC , each routing action takes 3 clock cycles for operation which includes one for routing algorithm, second for VC and Switch allocation and third for link traversal . So considering one such operation in source routing it takes only 7.995ns whereas it takes 19.416ns in distributed routing which is more than twice. Now considering 3 such operations in a pipelined manner , in source routing it takes 13.325 ns whereas in distributed it takes 32.36ns.

**6. Conclusion**

1. The implementation of Source Routing on Network on Chip decreased the time delay by 58.82%.
2. Source Routing reduced the speed gap between the processor and the network which increased the overall network efficiency.
3. As the time delay is been reduced , the frequency increases . Due to this we are able to match the frequency difference or the speed gap by a better factor.
4. The router becomes lighter in terms of components used and their complexities.
5. Code redundancy is removed and thereby the hardware required in each router is reduced.

**7. Future Enhancements**

The Source routing algorithm has been implemented for X-Y routing or Dimension order routing in the current stage. We would like to extend the Source routing to Odd-Even routing algorithm which is a very efficient algorithm in routing and also prevents Deadlock. Odd Even algorithm in the normal case gives more than one path . This can be done by restricting the Odd Even algorithm at many points in the Mesh so that one path is obtained .

**8. References**

1. Designing Efficient Source Routing for Mesh Topology Network on Chip Platforms by Saad Mubeen and Shashi Kumar, 2010 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools.
2. Stochastic Prefix-based Fault Tolerant SourceRouting Algorithm for Network on Chip by Gengchun Xu, Kaixin Ren, Naijie Gu, 2012 International Conference on Computer & Information Science (ICCIS).
3. Network on Chip Routing Algorithms by Ville Rantala, Teijo Lehtonen, Juha Plosila, University of Turku, Department of Information Technology Joukahaisenkatu, Turku, Finland
4. Review of XY Routing Algorithm for NETWORK-ON-CHIP Architecture by Shubhangi Chawade, Mahendra A Gaikwad and Rajendra M Patrikar, 2B. D College of Engineering, Sewagram, India.
5. Evaluation of Routing Algorithms on Mesh Based NoCs by Aline Vieira de Mello, Luciano Copello Ost, Fernando Gehm Moraes and Ney Laert Vilar Calazans .