

# **Performance Evaluation of XY and XTRANC Routing Algorithm for Network on Chip and Implementation using DART simulator**

*A Thesis submitted in partial fulfilment of the requirements for the degree of*

*Bachelor of Technology in  
Electronics and Instrumentation Engineering*

by  
MANISHA PANDA  
ROLL NO-111EI0144

Under the guidance of  
PROF. AYAS KANTA SWAIN



Department of Electronics and Communication Engineering  
NATIONAL INSTITUTE OF TECHNOLOGY, ROURKELA

राष्ट्रीय प्रौद्योगिकी संस्थान, राउरकेला

PIN-769008  
ODISHA, INDIA

DEPT. OF ELECTRONICS AND COMMUNICATION ENGINEERING  
NATIONAL INSTITUTE OF TECHNOLOGY, ROURKELA  
ROURKELA – 769008, ODISHA, INDIA



## CERTIFICATE

This is to certify that the Thesis entitled, " **Performance Evaluation of XY and XTRANC Routing Algorithm for Network on Chip and Implementation using DART simulator** " submitted by "**Manisha Panda**" bearing **Roll No. 111EI0144** to the National Institute of Technology Rourkela is a bonafide research work carried out by him under my guidance and is in partial fulfilment of the requirements for the award of the degree of "**Bachelors of Technology**" in Electronics and Instrumentation Engineering . The embodiment of this thesis is not submitted in any other university and/or institute for the award of any degree or diploma to the best of our knowledge and belief.

**Date: May 7, 2015 Prof. Ayas Kanta Swain**  
**Place: Rourkela Department of E. C. E**  
**National Institute of Technology ,Rourkela**

# ACKNOWLEDGEMENT

There are many people who are associated with this project directly or indirectly whose help and timely suggestions are highly appreciable for completion of this project. I would like to express my sincere gratitude to my supervisor Prof. Ayas Kanta Swain, for his supervision, encouragement, and support which has been instrumental for the success of this project. It was an invaluable experience for me to be one of his students. Because of him, I have gained a careful research attitude.

Thanks to those who are also the part of this project whose names could have not been mentioned here.

**Manisha Panda**

**Roll No. 111ei0144**

# ABSTRACT

In today's world Network on Chip(NoC) is one of the most efficient on chip communication platform for System on Chip where a large amount of computational and storage blocks are integrated on a single chip. NoCs are scalable and have tackled the shortcomings of SoCs . In the first part of this project the basics of NoCs is explained which includes why we should use NoC , how to implement NoC ,various blocks of NoCs .The next part of the project deals with the implementation of XY routing algorithm in mesh (3\*3) and mesh (4\*4) network topologies. The throughput and latency curves for both the topologies were found and a through comparison was done by varying the no of virtual channels.

In the next part an improvised routing algorithm known as the extended torus(XTRANC) routing algorithm for NoCs implementation is explained. This algorithm is designed for inner torus mesh networks and provides better performance than usual routing algorithms. It has been implemented using the CONNECT simulator. Then the DART simulator was explored and two important components namely the flitqueue and the traffic generator was designed using this simulator.

# Contents

CERTIFICATE .....	2
ACKNOWLEDGEMENT .....	3
ABSTRACT .....	4
TABLE OF FIGURES .....	6
INTRODUCTION .....	7
CHAPTER 1: Evolution of On chip Networks .....	8
1.1 Point to Point Communication .....	9
1.2 Shared Bus System .....	9
1.3 Network on Chip .....	10
Literature Review .....	11
Motivation .....	12
CHAPTER 2: Basics of Network on Chip .....	12
2.1 Communication Layers .....	13
2.2 Some Network communication related terms .....	13
2.3 Main Components of NoC .....	14
2.3.1 Resources .....	14
2.3.2 RNI .....	15
2.3.3 Router .....	15
2.4 Design Concepts of NoC .....	16
2.4.1 Topology .....	16
2.4.2 Switching Techniques .....	17
2.4.3 Routing Algorithm .....	17
2.5 Problems in routing .....	18
2.5.1 Deadlock .....	19
2.5.2 Livelock .....	19
2.5.3 Starvation .....	19
2.6. Performance parameters for Routing algorithm .....	19
CHAPTER 3- ROUTING ALGORITHM .....	20
3.1 XY Routing Algorithm .....	20

3.1.1 SIMULATION RESULTS OF XY ALGORITHM IN 3*3 MESH TOPOLOGY.....	22
3.1.2 SIMULATION RESULTS OF XY ALGORITHM IN 4*4 MESH TOPOLOGY.....	24
3.2 eXtended Torus routing algorithm for networks-on chip.....	25
3.2.1 Implementing the 1D XTRANC .....	25
3.2.1 2D IMPLEMENTATION OF XTRANC ALGORITHM IN A 10*10 MESH HAVING 4 INNER TORUS NETWORKS.....	28
CHAPTER 4: XTRANC ALGORITHM IMPLEMENTATION USING CONNECT SIMULATOR .....	30
4.1 TAILORING TO FPGAS .....	30
4.1.1 “Free” Wires.....	30
4.1.2 Storage Shortage.....	30
4.1.3 Frequency Challenged.....	31
4.1.4 Reconfigurability .....	31
4.2 CONNECT Router Architecture .....	31
4.3 Xtorus Topology Implementation .....	32
4.3.1 Results .....	33
CHAPTER 5: DART SIMULATOR .....	35
5.1 Introduction .....	35
5.2 DART ARCHITECTURE .....	35
5.3.1 Flit Queue (FQ) .....	35
5.3.2 Traffic Generator (TG).....	36
5.3.3 Router .....	37
5.3.4 Global Interconnect. ....	38
5.4 RESULTS.....	40
5.4.1 RTL Schematic of flitqueue: .....	40
5.4.2 RTL Schematic of traffic generator: .....	41
CHAPTER 6: CONCLUSION.....	42
CHAPTER 7: REFERENCES .....	43

## TABLE OF FIGURES

Figure 1: Point to point communication system.....	9
Figure 2: Shared Bus System for SoC .....	10
Figure 3: NoC for SoC .....	11
Figure 4: Message composition .....	14
Figure 5: Virtual Channel Router.....	16
Figure 6: Example of DOR, Oblivious and Adaptive routing.....	18

Figure 7: Allowed turns in XY routing .....	21
Figure 8: end to end latency vs load graph.....	23
Figure 9: throughput vs load graph.....	23
Figure 10: end to end latency vs load graph.....	24
Figure 11: throughput vs load graph .....	24
Figure 12: Typical 10 × 10 network which is partitioned to inner-torus networks.....	28
Figure 13: input .....	29
Figure 14: output path .....	29
Figure 15: Xtorus topology.....	32
Figure 16: Xtorus topology generated by connect simulator .....	32
Figure 17: Xtorus topology generated by connect simulator .....	33
Figure 18: testbench output .....	33
Figure 19: testbench output .....	34
Figure 20: testbench output .....	34
Figure 21: Algorithm to calculate the dequeue timestamp in a FQ.....	36
Figure 22: Flit Queue Data Path .....	36
Figure 23: Traffic generator datapath.....	37
Figure 24: Router datapath .....	37
Figure 25: VC allocator implementation: (a) classic router, (b) DART Router .....	38
Figure 26: DART global interconnect .....	39
Figure 27: Block diagram of flitqueue.....	40
Figure 28: RTL schematic displaying the inner blocks.....	40
Figure 29: Block diagram of traffic generator .....	41
Figure 30: RTL schematic of inner blocks.....	41

## INTRODUCTION

The demand of the human race is gradually increasing day by day. People always prefer a small electronic device having many more features in it. Thus VLSI industry found a new a paradigm

i.e. System on Chip (SoC). According to this paradigm different electronic or computing systems are embedded on a single chip. Those computing or electronic systems are also called as intellectual property cores. Since the introduction of research into multi-core chips more than a decade ago, on-chip networks have emerged as an important and growing field of research. As core counts increase, there is a corresponding increase in bandwidth demand to facilitate high core utilization and a critical need for scalable interconnection fabrics such as on-chip networks. On-chip networks will always be prevalent in computing domains ranging from high-end servers to embedded system-on chip devices. This diversity of platforms has led to research in on-chip networks spanning a variety of disciplines from computer architecture to computer-aided design, embedded systems, VLSI and more. In this chapter evolution of on-chip networks

## CHAPTER 1: Evolution of On chip Networks

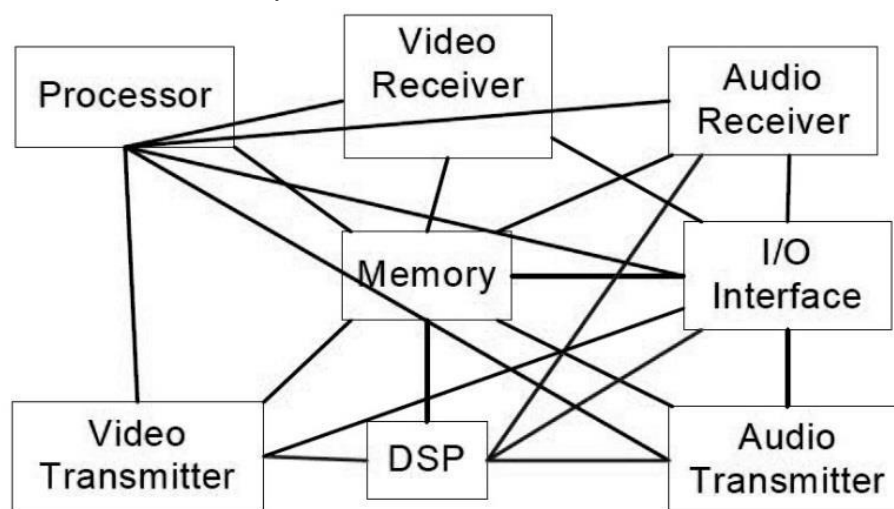
Here the discussion will start from various communication infrastructure for a System on Chip and how NoC became so popular. Here there is some discussion about some common terms in data networks and its relation to the NoC. This discussion will go on with main components and important design notions of NoC. There are three common communication systems for system on chip i.e. point to point communication and shared bus system and network on chip.



## 1.1 Point to Point Communication

Previously the designers prefer the direct point to point connection for the communication in system on chip. Here the resources are allowed to communicate directly through wires which are connected to each other. It doesn't need any priority providing system or arbitration unit. For a system on chip having more number of cores, this communication system requires large routing area, large routing delay and large number of pins for each core and becomes very complex. When direct point to point interconnections are used for communication, in this kind of communication system we can detect the quality of signal and delays occurred for routing. So testing of that system is a very tedious job. Due to these above problems, direct point to point interconnection system shows some drawbacks like underutilization of cores or resources, high complexity very poor reusability and poor scalability. A System on Chip which has less number of cores can use this communication infrastructure and can give best performance as compared to other systems.

## 1.2 Shared Bus System



*Figure 1: Point to point communication system*

Most of the SoCs uses shared bus system as their inter-core communication system. Here all the cores are connected to one or more than one bus. An interface is used for the connection of the bus to the cores. In this system the communication is managed by a bus arbiter system. Shared bus communication infrastructure requires less input output pins as compared to direct point to point communication system. So wiring cost and area is critically reduced. There are different kinds of buses present in literatures such as segmented, hierarchical, pipelined buses

etc. Shared bus system has many advantages. But still it has some disadvantages like data movement is slow due to contention and arbitration. In scalability point of view this system is not a fair choice as it can be scaled up to certain limits otherwise its efficiency will be very bad. A shared bus is shown in Figure 2.

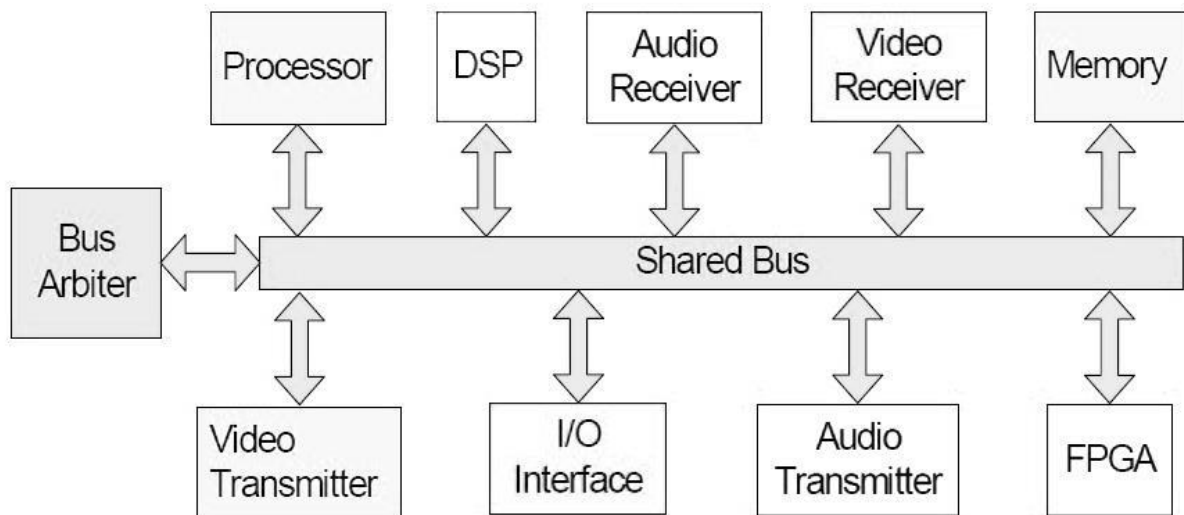


Figure 2: Shared Bus System for SoC

### 1.3 Network on Chip

The various disadvantages of the above two communication systems are less scalability, non-adaptive nature, underutilization of resources and less reuse factor. Researchers proposed a communication system which can avoid above problems which is termed as Network on Chip. It consists of three important components i.e. Routers, Resource Network Interface (RNI) and IP cores or resources. IP cores in the NoC are connected to the network switches. RNI (Resource Network Interface) is the communication bridge between the routers and IP cores as routers and IP cores have different communication protocols. For this on chip packet switched network data is converted into some formatted packets and those packets traverse from source to destination with the help of one or more routers in the network. Scalability of this communication system is sufficiently high. It also provides high reusability factor, reduced cost and less complexity.

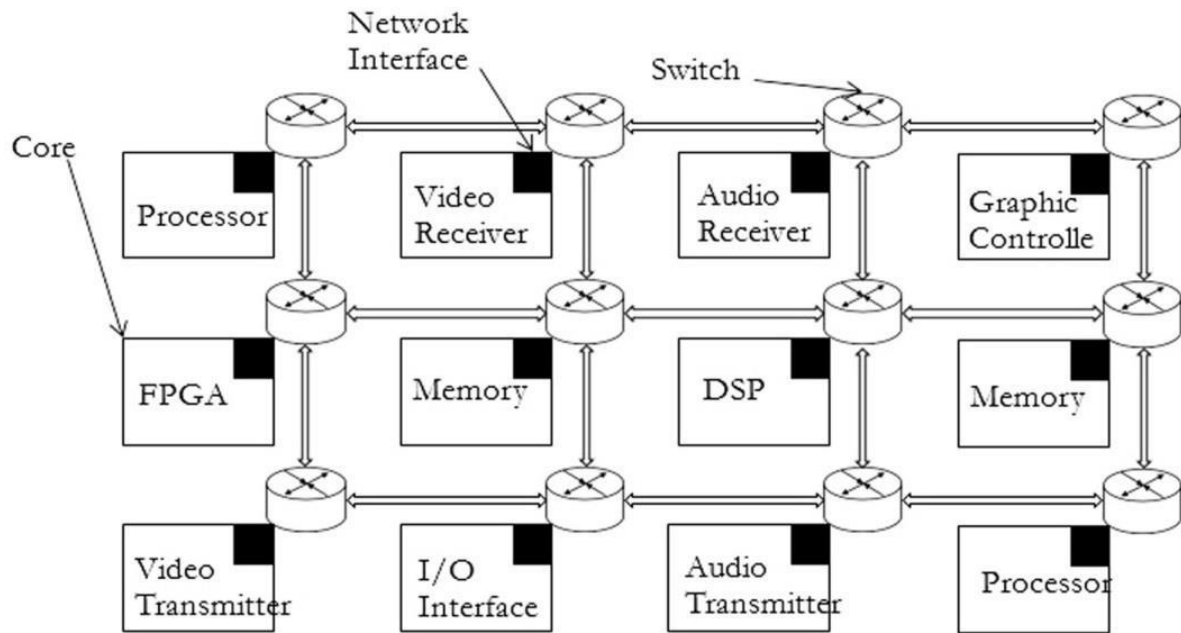


Figure 3: NoC for SoC

## Literature Review

Routing algorithm is an important design concept of Network on Chip. The purpose of routing algorithm is to determine an efficient route for the data or packets . The routing algorithms can be classified in various basis i.e. i) Fault tolerance ii) Adaptivity iii) Number of destinations. The routing algorithm is a crucial task in network layer. Gratz et al. propose a regional congestion aware routing which calculates the congestion level of the regions near the router and finds a low congested path by selecting less congested links. Ascia et al. introduce Neighbors-an-Path adaptive routing algorithm which uses immediate neighbor's congestion level for adaptive routing and the routers don't use any virtual channels which results in less routing area. A methodology called application specific routing algorithm (APSRA) has been proposed by Maurizio Palesi et al to develop deadlock-free efficient and routing algorithms for Network-on-Chip (NoC) platforms that are specialized for a particular application or a set of parallel running applications to maximize communication performance and adaptivity . The region-based routing (RBR) mechanism was proposed by Mejia et al in which destinations are grouped into network regions allowing an implementation which is efficient and also be viewed as a method to reduce the count of entries in the routing tables. Palesi et al. believed

that it is possible to design very efficient application specific routing algorithms which by using information regarding applications communication allocate traffic more uniformly.

## Motivation

Due to rapid development in VLSI industry millions of systems or features can be introduced in an electronic device by fabricating millions of transistors on a single silicon wafer or chip. Now a days chip producers are trying to release the multi-core products with many more cores in the system. This multi-core wave may lead to hundreds and even thousands of cores integrated on a single chip. Along with the integration of many general-purpose cores on a single chip, increasing the no of transistor will lead to greater system integration for multiprocessor systems-on-chip (MPSoCs). A system on chip with large number of IP cores definitely needs a compatible, scalable and high bandwidth communication system. Bus system and crossbar systems came into picture but they failed to provide an efficient communication system. So on chip network came into existence and replaces those two systems with ease. This on-chip network has some switches technically called as routers and routing wires. For various computing domains multi core architecture are very necessary requirement. These architectures will increase the levels of linking capability to the data centers. The applications which are throughput oriented will definitely require a high bandwidth communication. Communication delay or latency can have a substantial role on the performance of multi-threaded systems. Synchronization between threads will require a communication having low overhead for scalability purpose. In MPSoCs, Utilization of an on-chip network can help enable design isolation: MPSoCs utilize heterogeneous IP blocks from a variety of vendors; with standard interfaces, these blocks can communicate with the help of an on-chip network in a plug-and-play fashion.

## CHAPTER 2: Basics of Network on Chip

NoC follows the simplified rules or protocols of general data communication network. In this section the discussion is about the relation of NoC to the layered communication of OSI model.

Some general data network related term are also discussed in this section which have important role in NoC.

## 2.1 Communication Layers

Network on Chip uses layered communication system of OSI model like general data networks. OSI models consists of seven layers i.e. Data link layer ,Physical layer, Network layer, Transport layer, Session layer, Application layer, Presentation layer. Here each layer consists of some software with hardware components to perform certain functionality. Here each layer performs a task alone and independently. The layers provides various service to their upper layer and acquires service from their bottom layer.

## 2.2 Some Network communication related terms

### **Message**

Message is the information or data which are transmitted from source to destination resource. It is defined in application layer. A message can be of fixed length or can be of variable length according to requirement. Those messages are travelled in the networks in various forms described below.

### **Packet**

In packetization process message is divided into certain number of packets. Packets of the same message are independent of each other. Each packet has enough info to travel throughout the network. Generally a packet has three parts i.e. i) Header ii) Payload or Body iii) Trailer. Header contains controlling and routing information such as source and destination address. Sometimes it contain the whole route for data transmission. Payload contain actual data or information. Trailer indicates the end of packet.

### **Flit**

A packet can be divided into smaller elements. That small elements are called flits. They are flow control digits. Flits also has three parts i.e. Header flit, Body flit and Tail flit. The size of the flit is always fixed. Due to flits the storage devices required in the switches or routers will be very small. It is a good advantage.

## Phit

A flit can be divided into small units known as phits. They are physical transfer digits. It is travelled across a channel among the network switches but as one unit. It can be considered as link width Phit can be considered as the measuring parameter of link width as it indicates the number of wires need for data transfer between network routers. Size of a phit can be same as the size of flit or may be not.

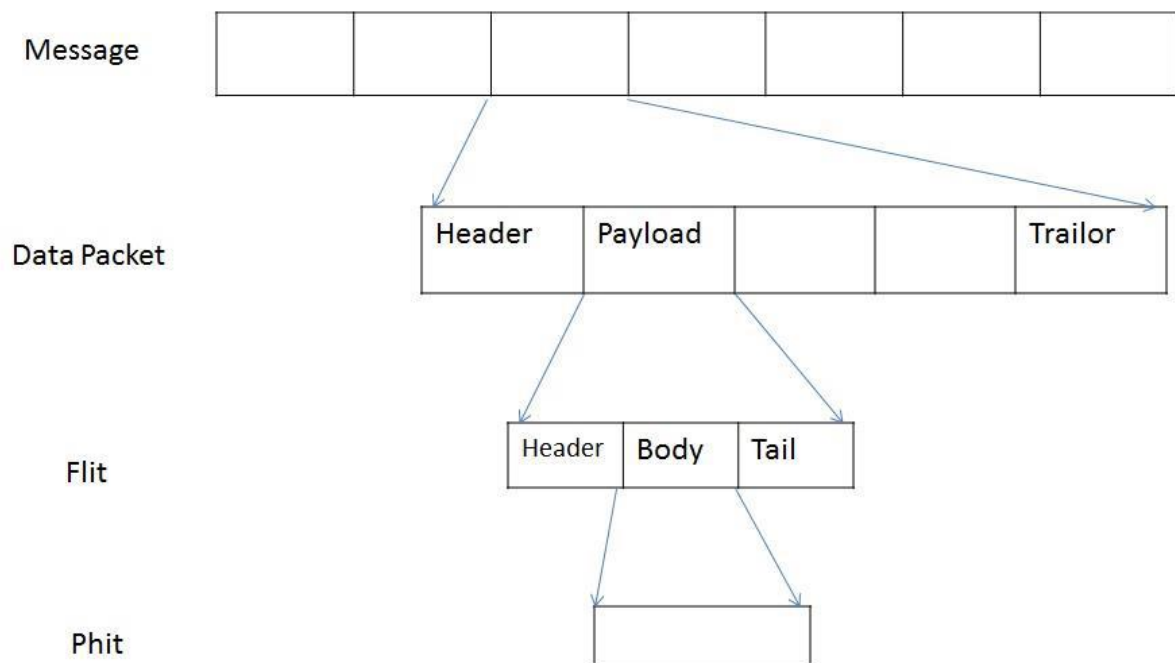


Figure 4: Message composition

## 2.3 Main Components of NoC

A NoC has three main and basic components i.e. i) Network switches technically called as routers ii) Resources or IP cores iii) Resource to Network Interfaces (RNI).

### 2.3.1 Resources

Here the clients (e.g., IP cores or Resources) are placed on city blocks separated by wires and the routers and wires are placed similar to grids in streets . The IP cores can be General Purpose

Processors, Amplifiers, FPGAs, memory ,ADCs, DSP, Graphic controller, Mixed signal Module, RF unit, I/O controller etc. Resource must have the same technology implementation as that of used in NoC. A designer can use own resources rather than buying from different vendors.

### 2.3.2 RNI

A Resource Network Interface is used to connect an IP core or resource to a router in NoC . Like that IP cores can transmit message packets to the network switch. Resource Network Interface has two parts which are i) Resource Dependent ii) Resource Independent . Design of Resource independent part is done in such a way that Resource Network Interface acts as another network switch to the connected network switch.

### 2.3.3 Router

Router is nothing more than a switch used in the network. It is a very important part of the on chip network like any other network infrastructure. They are just like back bone of Network on Chip. In an on chip network, the primary task of a router is to transmit the incoming data to the destination IP core if the router is directly connected to the destination resource otherwise that router has to send it to another router. A router implementation is based on three layers of communication in OSI model i.e. Physical, Datalink and Network layer. A designer must consider the simplicity of a router and design it like wise so that he can avoid some overheads like cost, area and power. Routing function implementation is the sole purpose of router for distributed routing. A generic router consists of five ports i.e. north, east, south, west, local port and a central matrix. The first four ports are used to connect to other routers and the local port is used to connect the IP core. In the router each port has an input channel and an output channel. The data packets move into the input channel of a port of router by which it is moved to the output channel of some other port.

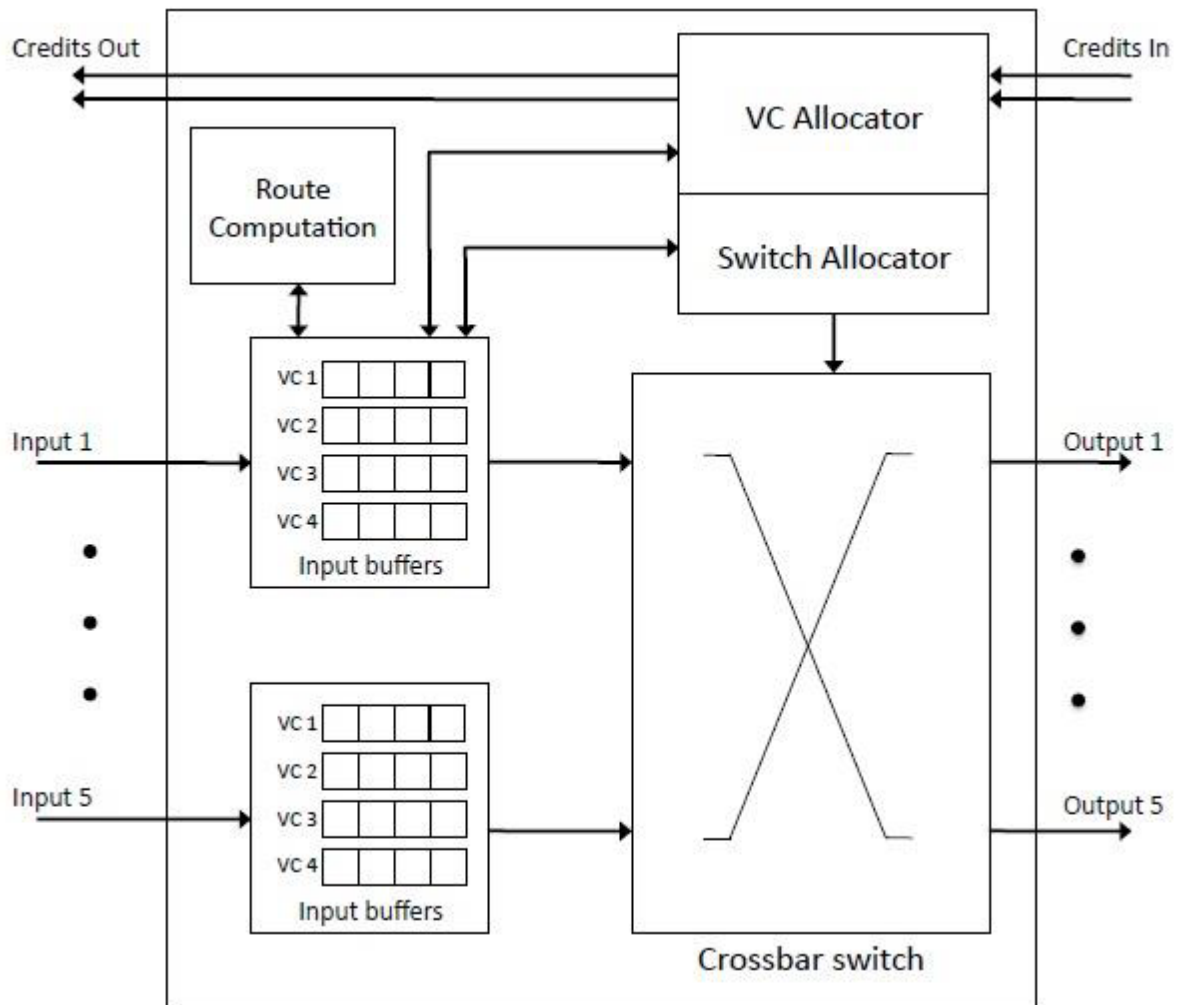


Figure 5: Virtual Channel Router

## 2.4 Design Concepts of NoC

In this section the important design concepts of NoC is described which are backbone of NoC. They are i) Topology ii) Switching techniques iii) Routing algorithms. The performance of NoC depends upon these above concepts. So the researchers keep on researching in these areas from past few years.

### 2.4.1 Topology

Most of the pattern of interconnection used for Network on Chip used for parallel computing field . According to the SoC paradigm these architectures are different for on chip and off chip networks. An ideal architecture or topology should provide less latency, less power consumption and more throughput, less routing area and less complexity. It is definitely



impossible to put all these features in a system because there is tradeoff between these features. So some researches has to sacrifice some advantages for these architectures to gain another one. So there is no architecture which can provide the required and desired performance of researchers. Topology are nothing but arrangement of nodes and channels inside the network. An efficient topology should be selected for a network so that the performance will be improve and it can fulfill the bandwidth and latency requirement with low cost. Topology can be regular or irregular. A topology can be called non-blocking if it can manage and serve all the requests coming to it.

#### 2.4.2 Switching Techniques

In this sub section the discussion will be on various flow control mechanism. It is also called as switching techniques . This technique operates in message level. The main task of switching technique is to establish connection between input and output channel inside the router. There are two types of popular switching techniques i.e. a) Circuit Switching Technique b) Packet Switching Technique. Packet switching has some classifications i.e. i) S&F (Store and forward) switching ii) VCT (Virtual Cut Through) switching iii) Wormhole switching iv) Virtual channel based switching.

#### 2.4.3 Routing Algorithm

First we need to determine the topology for the on-chip network and then a specific routing algorithm should be chosen. A routing algorithm determines the entire path for the message or data packets to reach the destination. Main task of a routing algorithm is to distribute the traffic from different nodes evenly throughout the network. Like that it will avoid hotspots and improve the network latency and throughput value by minimizing contention. A particular routing algorithm affects the router design complexity, area and hence affects the power consumption in whole network to achieve all performance requirements. There are various classification of routing algorithm. Generally routing algorithms are classified into three types. Those are i) Deterministic routing ii) Oblivious Routing iii) Adaptive routing. Researchers proposed various routing algorithms. But there is a Dimension order routing (DOR) which is very simple to implement. So it is preferable for the networks. DOR is an example of deterministic routing. XY routing comes under this DOR. Deterministic routing is a subset of oblivious routing. In deterministic routing one path is calculated between source and

destination and routing in that path is done throughout the process. In oblivious routing is done in different paths but it doesn't bother about the congestion in network. In adaptive routing more than one path is calculated between source and destination but only path is selected according to the congestion in the network. There is another classification of routing algorithms i.e. Minimal and Non-minimal routing. In minimal routing a path having smallest number of hops is selected. Non-minimal routing is just the opposite. Without network congestion non-minimal routing will increase the latency value. But with congestion it's performance is acceptable. A minimal routing which can avoid congested links will give satisfactory result by reducing the network latency. An example of DOR, oblivious and adaptive routing is shown. All the nodes are identified by (x,y) coordinates. Here (0,0) is chosen as the source and (2,3) is chosen as the destination. In DOR routing is done in that one path throughout the network. In oblivious routing two paths are calculated for routing. In adaptive routing algorithm the routing is done with that path avoiding congested links.

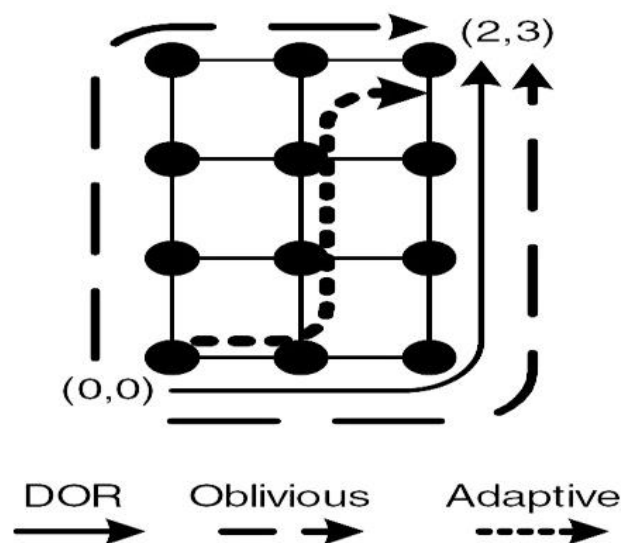


Figure 6: Example of DOR, Oblivious and Adaptive routing

## 2.5 Problems in routing

There are different kinds of problems arise during the routing process. Especially in oblivious routing this type of problem arises which results in blockage of traffic. So routing devices has to wait for the reduction of traffic and then try sending repeatedly. Deadlock, livelock and

starvation are potential problems on both oblivious and adaptive routing. These problems are discussed as follows in the next subsection.

### 2.5.1 Deadlock

When the data packets are moving around the network, they usually reserve some resource in between the path. When all the packets are waiting for each other to release the resources in a cyclic manner then this kind of situation is called as deadlock. As a result the all packets will be blocked inside the deadlock condition and they can't be routed to their destination and that is a huge loss in the on chip networks.

### 2.5.2 Livelock

This kind of problem happens when the packets are moving around a destination without reaching there. So data can't be routed to the destination. This kind problem will happen in non-minimal routing algorithm where the routing algorithm choose the longest path whether the shortest path exist or not by observing the network congestion. For throughput improvement this kind of problem should be avoided.

### 2.5.3 Starvation

This kind of problem is very similar to the real life situations. Different priority assignment to the data packets in NoC can cause this kind of problem. So high priority packets can reach their destination easily but the low priority elements will never reach their destination. This kind of situation arises because the high priority packets reserve the resources and the low priority packets get starved for the resources. This kind of problem can be avoided by using an appropriate routing algorithm which has some bandwidth reservation for the low priority packets.

## 2.6. Performance parameters for Routing algorithm

Performance requirements that every NoC must satisfy

- Small latency

- Guaranteed throughput
- Path diversity
- Sufficient transfer capacity
- Low power consumption
- Fault and distraction tolerance
- Architectural requirements of scalability and programmability

**Network Latency** is measured from the time its head flit is generated by the source to the time its tail flit is consumed by the destination. Let  $L_{ij}$  be the packet  $j$  and  $N_i$  be the number of packet received by processor  $i$  (After warm-up time).  $N$  is the number of processors in the platform.

Average Network Latency =  $L_{avg} = 1/N \sum (1/N_i \sum L_{ij} \forall j)_{i=1 \dots N}$  for  $i = 1, 2 \dots N$  and for all  $j$

**Network Throughput** is defined as the rate at which the network can successfully accept and deliver the injected packets. Let  $T_{sim}$  and  $T_{warm}$  be the simulation time and warm-up time respectively.

Average Network Throughput (in packets per unit time per node) is given by

$$T_{avg} = 1 / (T_{sim} - T_{warm}) (\sum N_i)_{i=1 \dots N} \text{ where } i = 1, 2, \dots, N$$

## CHAPTER 3- ROUTING ALGORITHM

### 3.1 XY Routing Algorithm

As discussed earlier a deterministic routing algorithm uses a fixed routing path throughout the process. XY routing algorithm comes under deterministic routing algorithm. This algorithm can be implemented for both for regular and irregular network topology. It is called dimension order routing (DOR). It follows the concept of minimal turning routing. In this routing each node or router of NoC is identified by the (x, y) co-ordinates of that node for a 2D mesh.

According to this algorithm the data packets will traverse in X-direction towards the destination column. After finding the destination column the data packets will traverse to the destination node. Data movement in this algorithm is described in an example in Figure 4.2. This algorithm simply states that “First the data will move in X-direction and then in Y-direction”. That is why the name of the algorithm is XY Routing algorithm. According to this algorithm the packets can’t move first in Y-direction then in X-direction. So it has some routing or turning restrictions. Due to which it becomes deadlock free. According to this algorithm, (x ,y) coordinate of Current router is compared to the (x,y) coordinate of Destination router . If they are equal, this will indicate data packets have reached its destination and they will be routed to the resource through the local port. If they are not equal, the x-coordinate of destination is initially compared to the x-coordinate of current router. When x-coordinate of current router is less than x-coordinate of destination router, the data packets will be transmitted to the East direction, when x-coordinate of current router is greater than x-coordinate of destination router, then packets will be routed to West direction. Then check if the data packet is already on the destination column or not. If this last condition is true, the vertical address of destination is compared to the vertical address of current router. Data packets will be traversed to South direction when vertical address of current router is less than vertical address of destination router, to North when vertical address of current router is greater than vertical address of destination router.

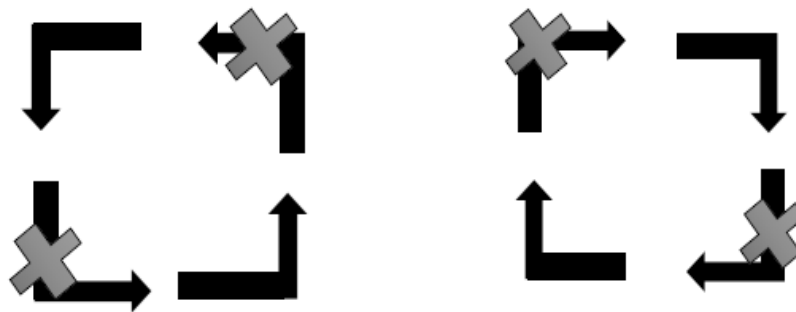


Figure 7: Allowed turns in XY routing

Some common term are

Source router: ( $S_x$ ,  $S_y$ )

Destination router: ( $D_x$ ,  $D_y$ )

Current router: (Cx , Cy)

Xoffset = Dx - Cx

Yoffset = Dy - Cy

Algorithm:

begin

if (Xoffset > 0 ) then

Select EAST port;

else

if (Xoffset <0) then

Select WEST port ;

else

if (Xoffset = 0) then

{

if (Yoffset < 0)

Select SOUTH port ;

else

if (Yoffset > 0)

Select NORTH port;

else

if (Yoffset = 0)

Select LOCAL port;

}

End

### 3.1.1 SIMULATION RESULTS OF XY ALGORITHM IN 3\*3 MESH TOPOLOGY

Taking various loads the end to end latency was calculated two virtual channels(1 and 2).It is represented in the same graph for proper comparison.as we can observe by increasing the no of virtual channels the latency is decreasing and the throughput is increasing.

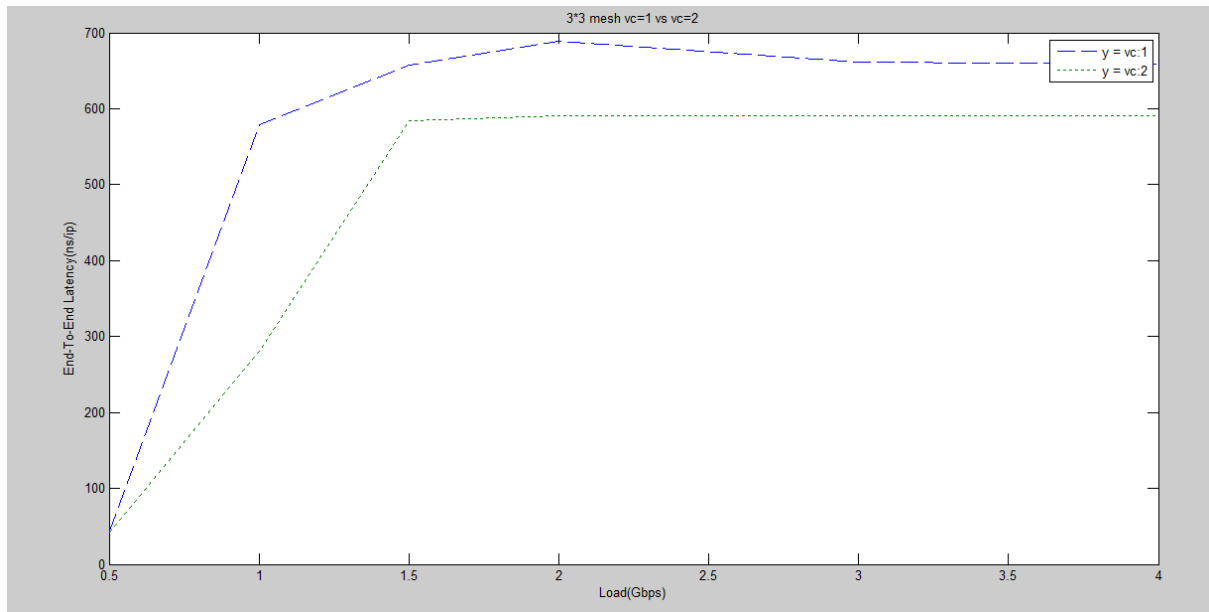


Figure 8: end to end latency vs load graph

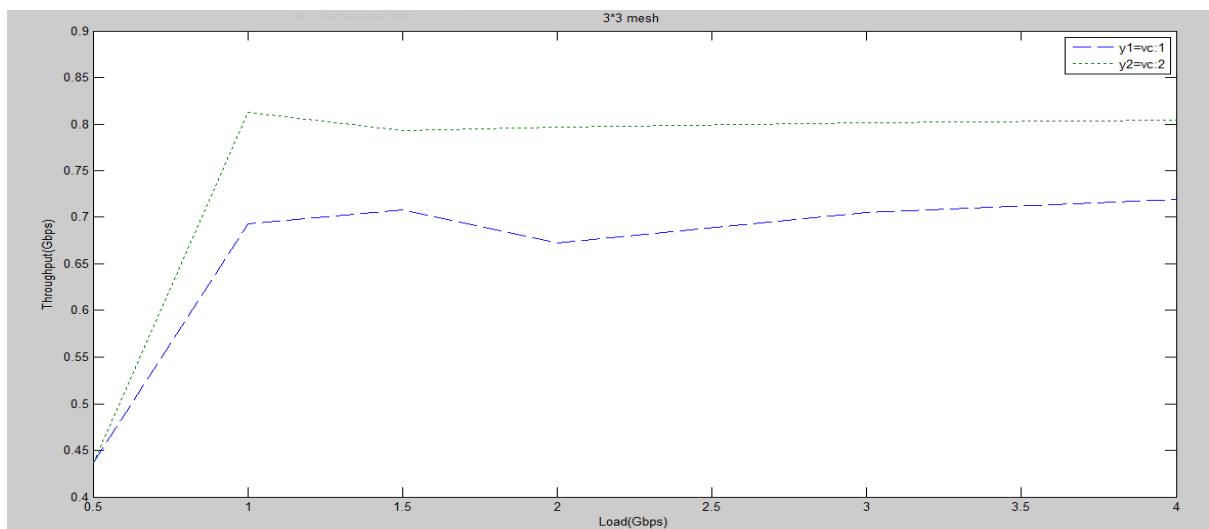


Figure 9: throughput vs load graph

### 3.1.2 SIMULATION RESULTS OF XY ALGORITHM IN 4\*4 MESH TOPOLOGY

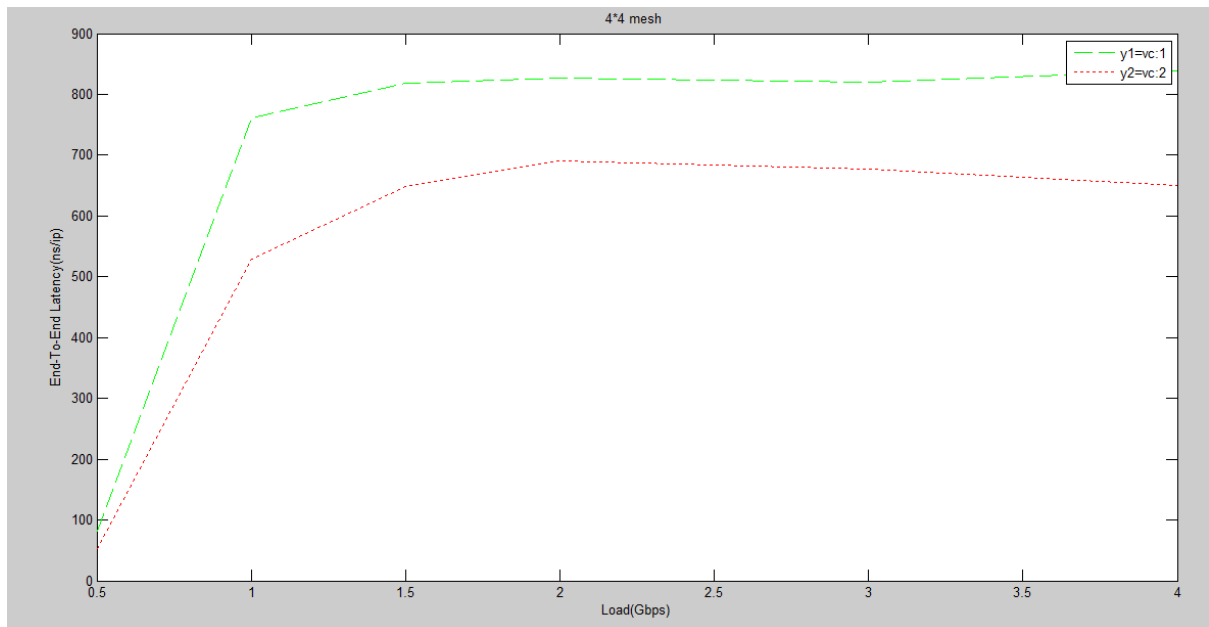


Figure 10: end to end latency vs load graph

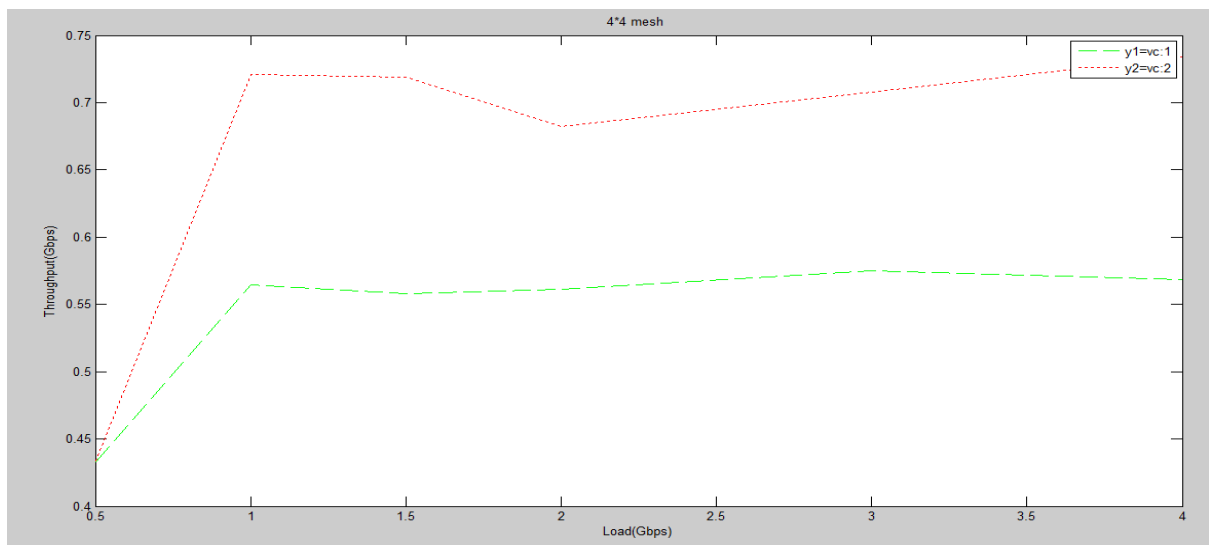


Figure 11: throughput vs load graph



## 3.2 eXtended Torus routing algorithm for networks-on chip

The eXtended Torus routing algorithm for networks-on-chip (XTRANC) supports topologies based on a variable size and number of inner-torus building blocks. A mesh topology is partitioned into various sub-networks to improve the performance of the mesh efficiently. Here depending on the behaviour of the application and resource availability additional links are added to the mesh topology at run-time to reduce traffic congestion. XTRANC allows the addition of links as requested by different parts of the application without centralized control and in spite of its dynamic behaviour this algorithm remains deadlock free.

### 3.2.1 Implementing the 1D XTRANC

One dimensional XTRANC topology consists of node and rings. These two parameters are used for the router initialization and implementation of the routing algorithm. Here when a particular node is a regular node its ring and node parameters are set to be zero. If a node is a boundary node then the ring parameter is set to 1 and if the node lies inside the inner ring then the node parameter is set to be 1. Nodes which are not in an inner-ring employ the 'X' routing algorithm and to implement the IRN map. When the node is located in the inner ring it can actually be in two positions i.e inside node or border node of the inner ring. When the destination nodes are outside the inner-ring and the source node is inside, the packet will be routed to the inner-ring border which is the same side as the destination node. When the source and destination nodes are in the inner-ring, the packet will be routed according to the IRN Map

```

Inputs: X Coordinates of current node (Xcurrent, Xstart, Xend),
        X destination node (Xdest), and radix n;
Output: Selected output Channel;
n:= Xend-Xstart+1;
if (Xdest=Xcurrent) then return Local port;
else
{if ( Xdest<=Xend and Xdest>=Xstart )then
  { if (Xdest-1=Xcurrent-n) or
    ((Xcurrent = Xstart+n-4) and (Xdest = Xstart+n-2)) or
    ((Xcurrent>= Xstart+n-2) and (Xdest<= Xstart+n-4)) or
    ((Xdest-Xcurrent> 0) and (Xdest<= Xstart+n-3)) or
    (Xdest-1=Xcurrent)
    then return X+;
    if (Xdest-Xcurrent =-1) or (Xdest-Xcurrent = n-1) or
      ((Xcurrent = Xstart+n-1) and (Xdest = Xstart+n-3)) or
      ((Xcurrent<= Xstart+n-5) and (Xdest>= Xstart+n-2)) or
      ((Xdest-Xcurrent< 0) and (Xcurrent<= Xstart+n-3)) or
      ((Xdest-Xcurrent> 1) and (Xdest= Xstart+n-1))
      then return X-;}
  }else
  {if (Xdest < Xstart )then
    { if (Xstart-1=Xcurrent-n) or
      ((Xcurrent = Xstart+n-4) and (Xstart= Xstart+n-2)) or
      ((Xcurrent>= Xstart+n-2) and (Xstart<= Xstart+n-4)) or
      ((Xstart-Xcurrent> 0) and (Xstart<= Xstart+n-3)) or
      (Xstart-1=Xcurrent)
      then return X+;
      if (Xstart-Xcurrent =-1) or (Xstart-Xcurrent = n-1) or
        ((Xcurrent = Xstart+n-1) and (Xstart = Xstart+n-3)) or
        ((Xcurrent<= Xstart+n-5) and (Xstart>= Xstart+n-2)) or
        ((Xstart-Xcurrent< 0) and (Xcurrent<= Xstart+n-3)) or
        ((Xstart-Xcurrent> 1) and (Xstart= Xstart+n-1))
        then return X-;}
      }else
      {if (Xdest>Xend )then
        {
          if (Xend-1=Xcurrent-n) or
            ((Xcurrent = Xstart+n-4) and (Xend= Xstart+n-2)) or
            ((Xcurrent>= Xstart+n-2) and (Xend<= Xstart+n-4)) or
            ((Xend-Xcurrent> 0) and (Xend<= Xstart+n-3)) or
            (Xend-1=Xcurrent)
            then return X+;
            if (Xend-Xcurrent =-1) or (Xend-Xcurrent = n-1) or
              ((Xcurrent = Xstart+n-1) and (Xend = Xstart+n-3)) or
              ((Xcurrent<= Xstart+n-5) and (Xend>= Xstart+n-2)) or
              ((Xend-Xcurrent< 0) and (Xcurrent<= Xstart+n-3)) or
              ((Xend-Xcurrent> 1) and (Xend= Xstart+n-1))
              then return X-;}
          }
        }
      }
    End;

```

```

Inputs: X Coordinates of current node (Xcurrent ,Xstart, Xend),
        X destination node (Xdest);
Output: Selected output Channel;
n:= Xend-Xstart+1;
if (Xdest=Xcurrent) then return Local port;
Else
{if Xcurrent=Xend and Xdest<=Xend and Xdest>=Xstart then
  {if (Xdest-1=Xcurrent) or
    ((Xcurrent = Xstart+n-4) and (Xdest = Xstart+n-2)) or
    ((Xcurrent>= Xstart+n-2) and (Xdest<= Xstart+n-4)) or
    ((Xdest-Xcurrent> 0) and (Xdest<= Xstart+n-3))
  then return i port;
  elseif (Xdest-Xcurrent =-1) or (Xdest-Xcurrent = n-1) or
    ((Xcurrent = Xstart+n-1) and (Xdest = Xstart+n-3)) or
    ((Xcurrent<= Xstart+n-5) and (Xdest>= Xstart+n-2)) or
    ((Xdest-Xcurrent< 0) and (Xcurrent<= Xstart+n-3)) or
    ((Xdest-Xcurrent> 1) and (Xdest= Xstart+n-1))
  then return X-; }
elseif (Xcurrent=Xend and Xdest <= Xstart)
then return i port;
elseif (Xcurrent=Xend and Xdest>Xend)
then return X+;}
Else
{if Xcurrent=Xstart and Xdest<=Xend and Xdest>=Xstart
  {if (Xdest-Xcurrent = -1) or (Xdest-Xcurrent = Xstart+n-1) o
    ((Xcurrent = Xstart+n-1) and (Xdest = Xstart+n-3)) or
    ((Xcurrent<= Xstart+n-5) and (Xdest>= Xstart+n-2)) or
    ((Xdest-Xcurrent< 0) and (Xcurrent<= Xstart+n-3)) or
    ((Xdest-Xcurrent> 1) and (Xdest= Xstart+n-1))
  then return i port;
  elseif (Xdest-1=Xcurrent ) or (Xdest-1=Xcurrent-Xstart-n) or
    ((Xcurrent = Xstart+n-4) and (Xdest = Xstart+n-2)) or
    ((Xcurrent>= Xstart+n-2) and (Xdest<= Xstart+n-4)) or
    ((Xdest-Xcurrent> 0) and (Xdest<= Xstart+n-3))
  then return X+ port;}
elseif Xcurrent=Xstart and Xdest >=Xend
then return i port;
elseif Xcurrent=Xstart and Xdest<=Xstart
then return X- port;}
}
End;

```

### 3.2.1 2D IMPLEMENTATION OF XTRANC ALGORITHM IN A 10\*10 MESH HAVING 4 INNER TORUS NETWORKS

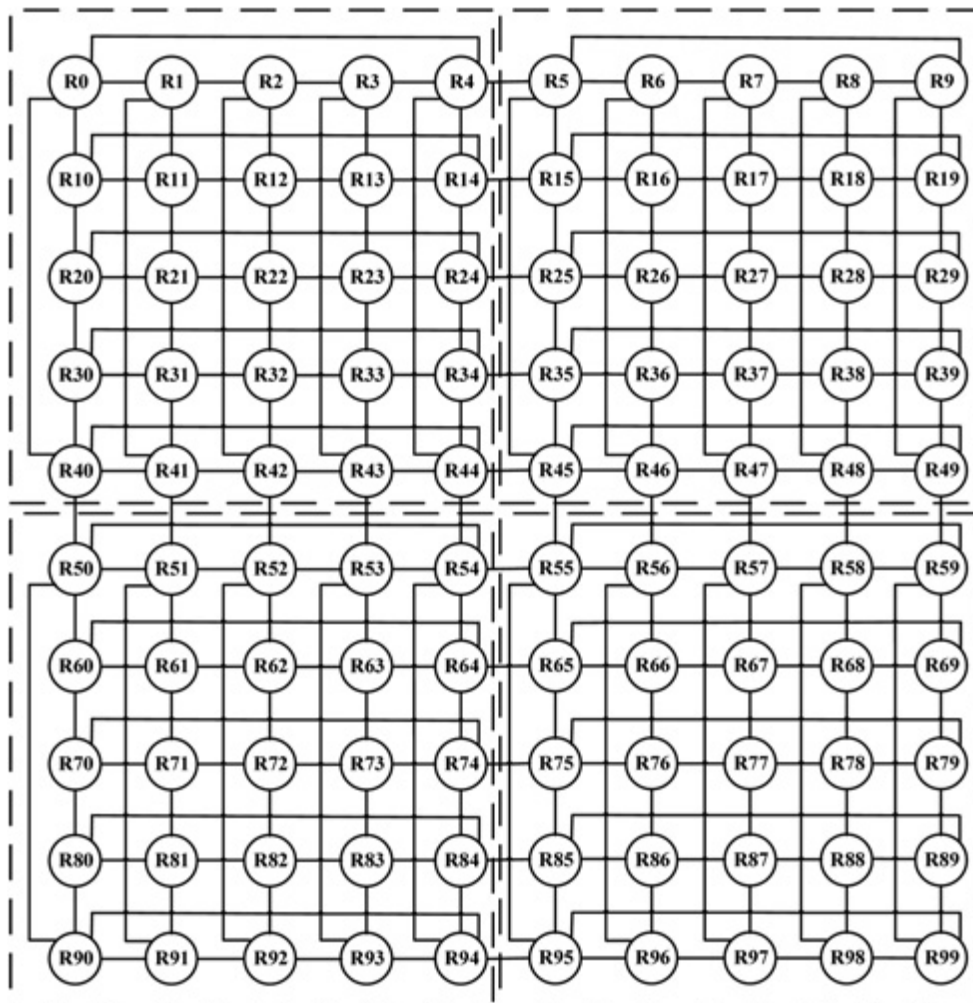
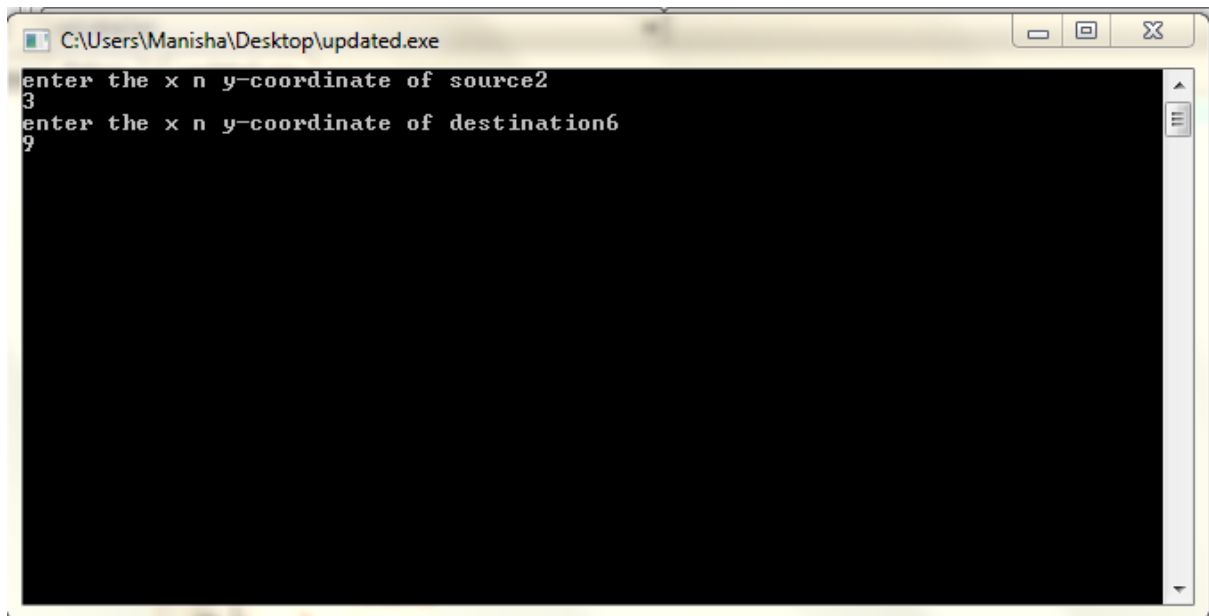


Figure 12: Typical  $10 \times 10$  network which is partitioned to inner-torus networks

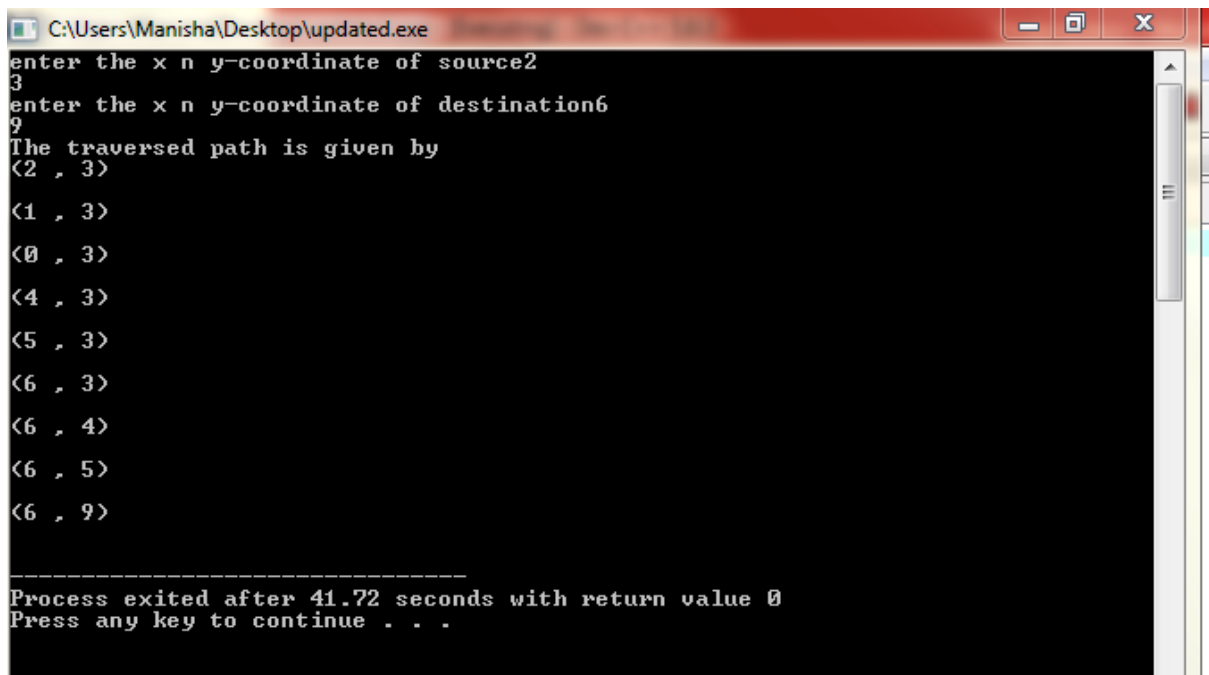
## RESULTS

The above algorithm was implemented in dev c++ compiler and the following result was found. When the user inputs the coordinates of the source and destination nodes the program gives the path travelled as the output



```
C:\Users\Manisha\Desktop\updated.exe
enter the x n y-coordinate of source2
3
enter the x n y-coordinate of destination6
9
```

Figure 13: input



```
C:\Users\Manisha\Desktop\updated.exe
enter the x n y-coordinate of source2
3
enter the x n y-coordinate of destination6
9
The traversed path is given by
<2 , 3>
<1 , 3>
<0 , 3>
<4 , 3>
<5 , 3>
<6 , 3>
<6 , 4>
<6 , 5>
<6 , 9>

-----
Process exited after 41.72 seconds with return value 0
Press any key to continue . . .
```

Figure 14: output path

# CHAPTER 4: XTRANC ALGORITHM

## IMPLEMENTATION USING CONNECT

### SIMULATOR

The rapidly growing capacity of FPGAs along with the introduction of hardwired support has made FPGAs an attractive platform for hosting a SoC design. Now designers need a flexible infrastructure to implement communication between thousands of interacting nodes. There are various factors due to which FPGAs seems to be a better option. Relative availability of wires compared to memory and logic, huge no of modest sized buffers provide the scarcity of on-die storage resources , the return on performance from deep pipelining and field configurability. A synthesizable RTL level design of multi-node NoCs can be generated using connect simulator based on a simple but flexible router architecture

#### 4.1 TAILORING TO FPGAS

FPGA is a peculiar substrate for hardware realization as compared to ASICs because it consists of a different set of design trade-offs between logic, memory , clock frequency and wires. Here we have focussed on specific FPGA characteristics and showed what influence they have on decisions related to fundamental CONNECT design .

##### 4.1.1 “Free” Wires

The main expectation from a FPGA is that it should handle a wide spectrum of designs with varying degree of connectivity. FPGAs have highly and densely connected wiring substrate. Generally for an average application the routing resources are underutilized so in these cases we can take these wires as plentiful or even “free”, as they are available in plenty relative to configurable logic blocks and on-chip storage. The following are the implications.

The data paths along with channels between routers should be made wide so as to use the largest possible fraction of the wires available and hence make minimum use of the routing substrate. Flow control mechanisms could also be adapted to use a wider interface, so that it can indirectly reduce router storage requirements.

##### 4.1.2 Storage Shortage

Storage in FPGA can be in two forms: (1)SRAMs based on logic lookup tables and (2) SRAM macros with kilo-bits of capacity. The memory macros can not be subdivided. As a full Block RAM has to be consumed, even if only a fraction of its capacity is required it leads to an inefficient design. The Distributed RAMs are expensive in comparison to Block RAMs, especially in building of large buffers. This sets up a situation where NoCs on FPGAs pay a

high cost for storage because NoCs have a typical requirement of a large number of buffers . The capacity of these buffers are smaller than the Block RAMs but bigger than Distributed RAMs.

#### 4.1.3 Frequency Challenged

A design when operated on an FPGA will work at very low clock frequency than when implemented in ASIC. This is because first of all, Look-Up Tables used to implement arbitrary logic functions are inherently very slow compared to ASIC. Secondly, in order to emulate arbitrary logic blocks, FPGAs often need long interconnect wires which are required to chain a very large number of LUT elements. The time spent traversing these wires adds to the path travelled in the design.

#### 4.1.4 Reconfigurability

FPGAs are reconfigurable in nature and this property differentiates them from ASICs and provides various opportunities as well as challenges for implementing an Noc on a FPGA.

## 4.2 CONNECT Router Architecture

CONNECT routers are heavily configurable and among other parameters, they support:

- configurable number of input and output ports
- configurable Number of virtual channels (VCs)
- Variable flit width
- Two flow control mechanisms
- Variable flit buffer depth
- Flexible user-specified routing
- Four allocation algorithms Router Datapath.

### 4.3 Xtorus Topology Implementation

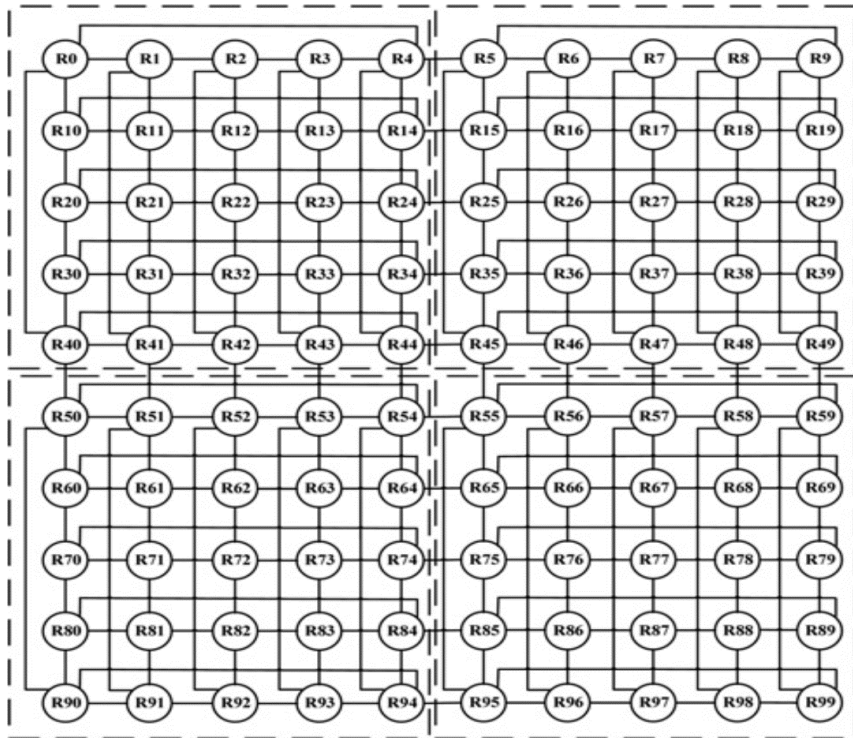


Figure 15: Xtorus topology

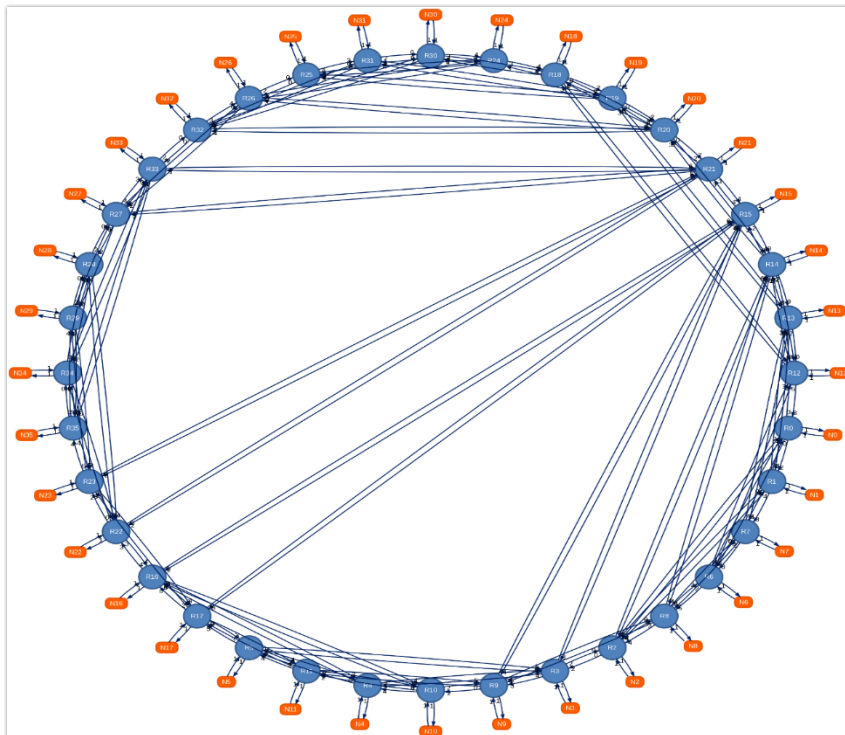


Figure 16: Xtorus topology generated by connect simulator



### 4.3.1 Results

#### Design Overview

Device Utilization Summary (estimated values)				
Logic Utilization	Used	Available	Utilization	
Number of Slice Registers	19920	1224000	1%	
Number of Slice LUTs	60753	612000	9%	
Number of fully used LUT-FF pairs	6978	73695	9%	
Number of bonded IOBs	1694	900	188%	
Number of BUFG/BUFGCTRLs	1	32	3%	

Figure 17: Xtorus topology generated by connect simulator

#### Test Bench Output

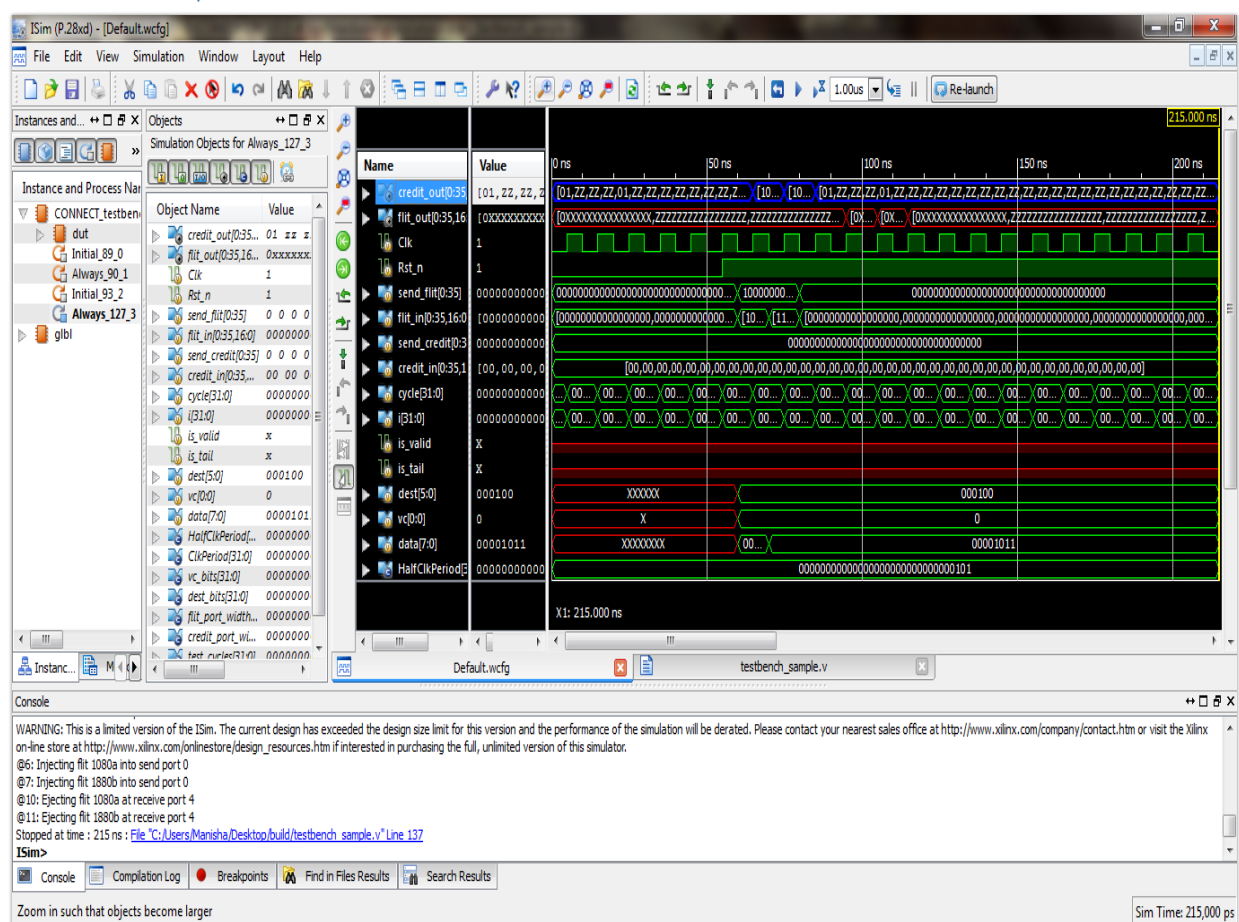


Figure 18: testbench output

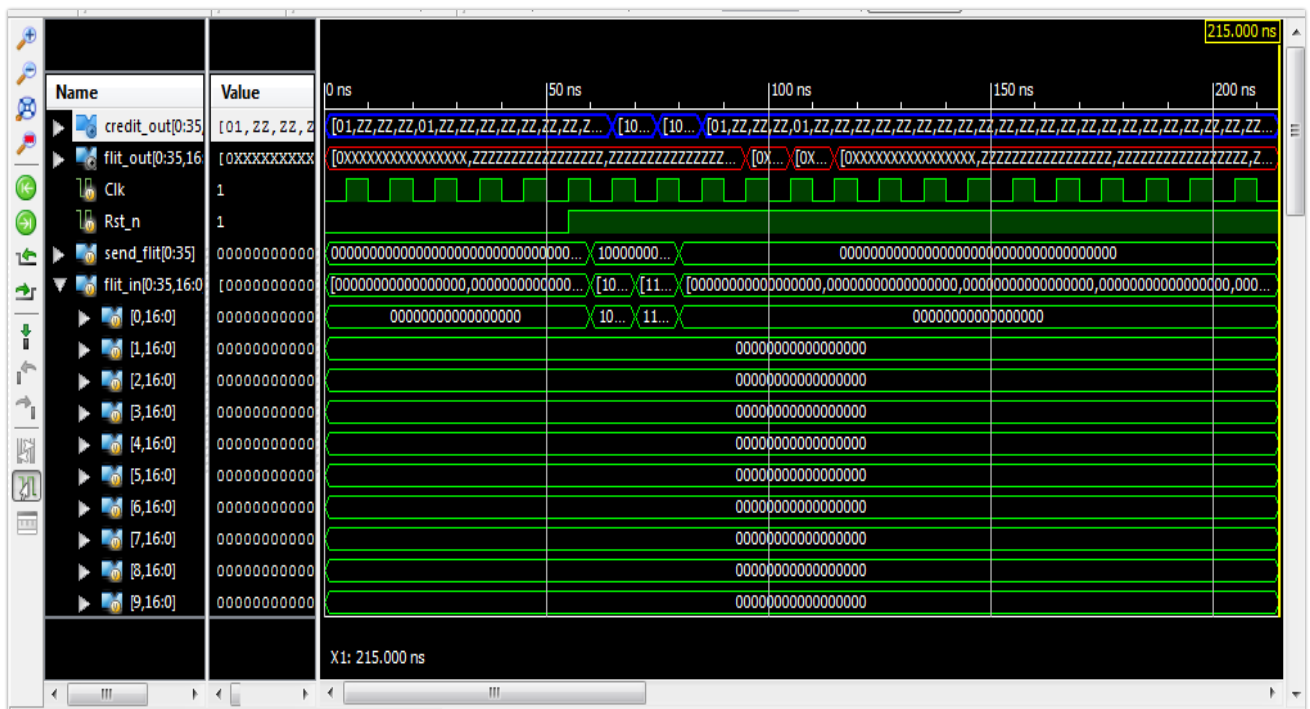


Figure 19: testbench output

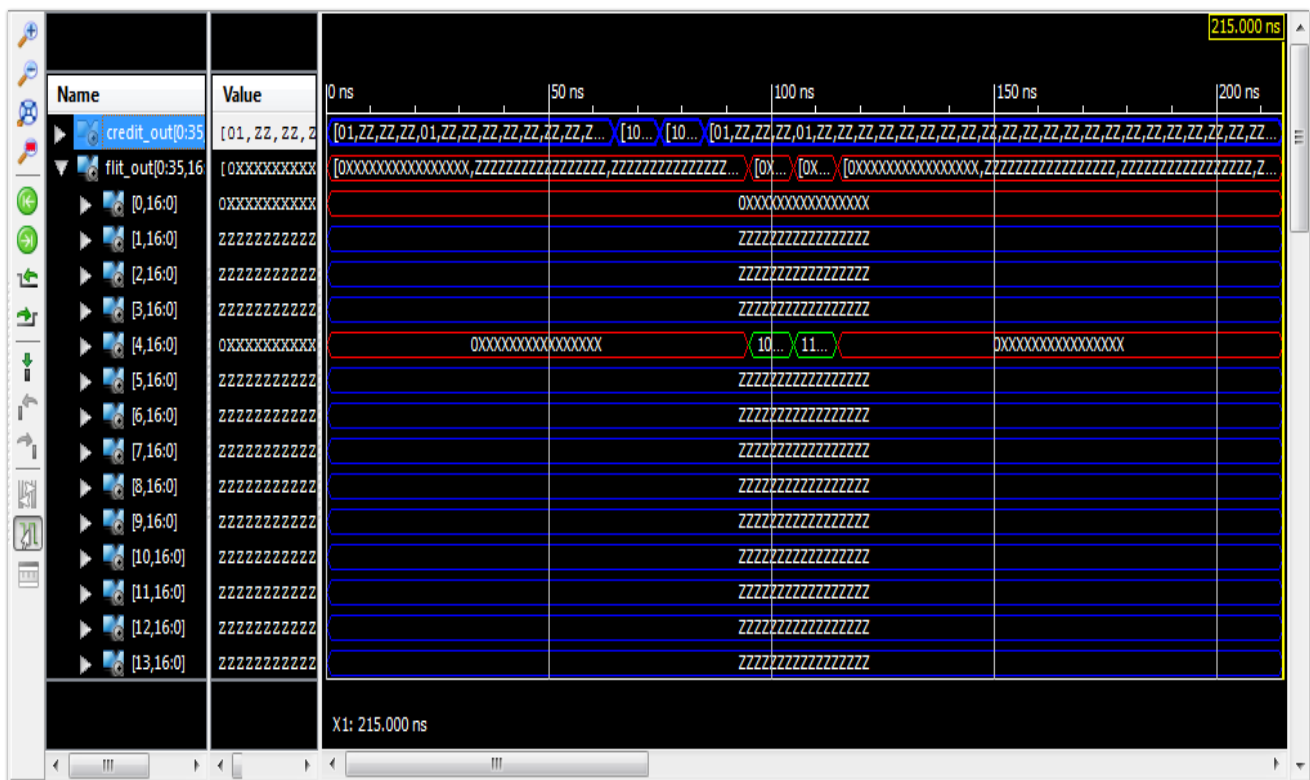


Figure 20: testbench output

# CHAPTER 5: DART SIMULATOR

## 5.1 Introduction

Nowadays systems-on-chip increasingly uses packet switched networks-on-chip (NoCs) to cater to the growing demand for on-chip communication. These designs are sensitive to many factors like buffer sizes, topology, routing algorithms, and flow control mechanisms. Software simulation is much in use. They have the advantages of being very flexible, fast compilation, easy to program, and are deterministic. However, simulation of large NoCs in software simulator is slow, which is a serious drawback. So to lessen the simulation time the level of abstraction has to be increased. Due to increase in on-chip logic and memory capability of recent FPGAs allow an entire on-chip system to be implemented on a FPGA. These emulators can reduce simulation time very much compared to software. However, this direct mapped approach has three key drawbacks relative to software simulation:

- (i) some change in the simulated NoC requires manually redesigning the emulator HDL,
- (ii) compilation of the FPGA has to be done again due to redesign, which can take hours, or up to a day for a large design, and
- (iii) The FPGA capacity determines the maximum simulatable NOC size.

## 5.2 DART ARCHITECTURE

The basic aim of the DART architecture is to provide programmability by decoupling (i) the simulator and simulated NoC architecture and (ii) from simulated cycles from DART cycles. The DART architecture has three components: Traffic Generators (TGs), and Routers and Flit Queues (FQs). The implementation of various other topologies just requires some configuration change in the global interconnect so resynthesize need not be done.

### 5.3.1 Flit Queue (FQ)

Flit Queue contains the VC buffers and the latency of the link connecting the port. The buffers are FIFOs and are independent of each other. They are implemented by dividing a single block RAM among VCs. Every coming flit is queued according to its vc after his time stamp is updated. We can configure latency and bandwidth per FQ. When a flit comes to the front of a FIFO the global simulation time is equal to the time stamp and then only it is forwarded to the next router. This is to make sure that all flits arrive in chronological order. Like a flit a credit can leave an FQ only during its scheduled dequeue time.

```

N_through ++
if(T_enqueue>T_last_flit || N_through>=bandwidth)
    T_dequeue = max(T_enqueue, T_last_flit+1)
    N_through = 1
else
    T_dequeue = T_enqueue
T_dequeue += latency

```

Figure 21: Algorithm to calculate the timestamp in a FQ.

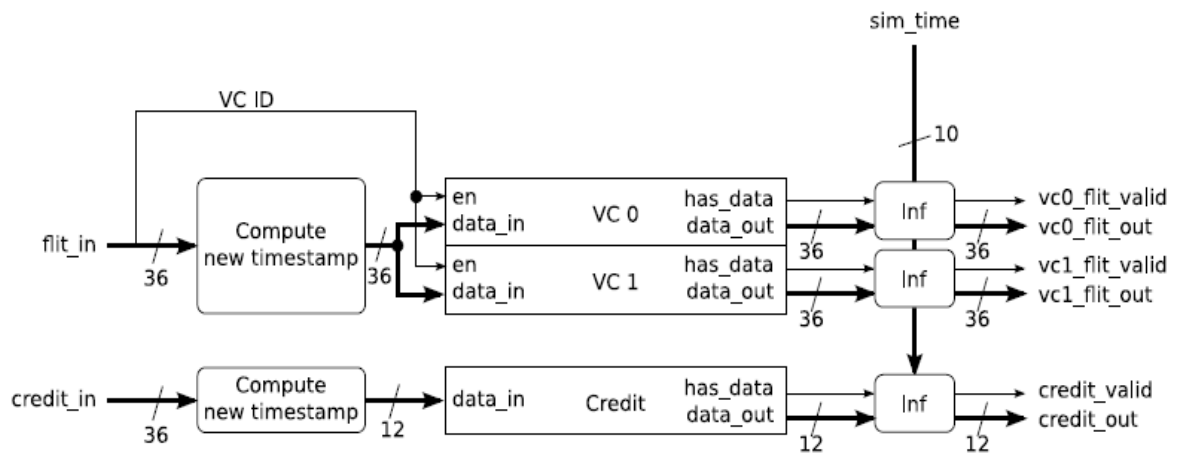


Figure 22: Flit Queue Data Path

### 5.3.2 Traffic Generator (TG)

The traffic generator can inject traffic in only two modes: synthetic or dynamic. Synthetic is used for stress testing. The dynamic traffic gives an interface to include DART into a simulator. Each mode is configurable. In synthetic mode, a TG injects flits in bursts of fixed-sized packets using a Bernoulli process. The minimum packet size is 2 flits. Among various other components the injection interval and the node address of the destination is configurable per TG. In the second mode, a TG receives packet descriptors from the host computer and injects packets accordingly. Packet size can vary from 2 to 256 flits in powers of 2. Each TG also contains two FQs. The input buffer models the last-hop delay to the traffic generator, and the source queue is modelled by the output buffer.

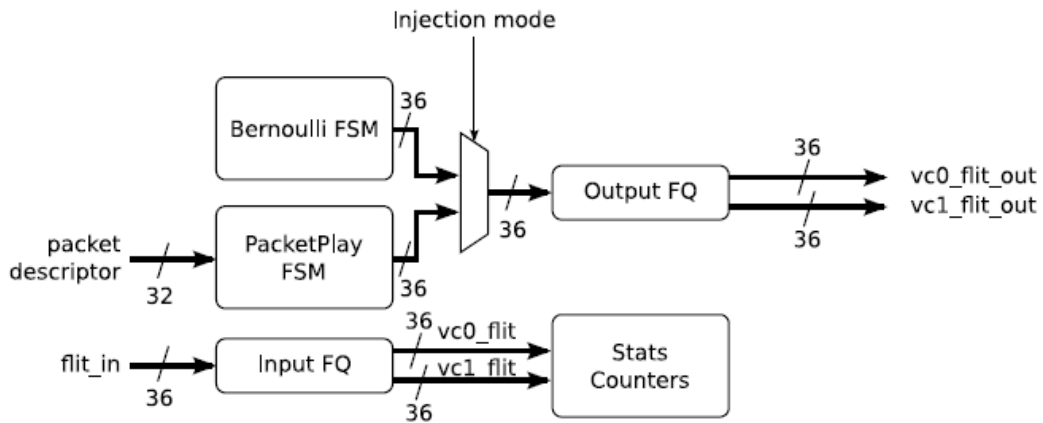


Figure 23: Traffic generator datapath

### 5.3.3 Router

Generally NoCs uses the classic wormhole router, which has per-VC flit buffers, routing logic, VC and crossbar and switch allocators. As the FQs model the flit buffers, the Router only consists of the routing logic and allocation logic. The no of ports is controlled by a Verilog parameter. A 4-bit counter for each output VC is used to implement credit-based flow control. Initial credit values represent the number of entries in the input buffer at the downstream router. When a flit is routed the counter is decremented and when a credit is received the counter is incremented. The values are configurable for each VC and Router.

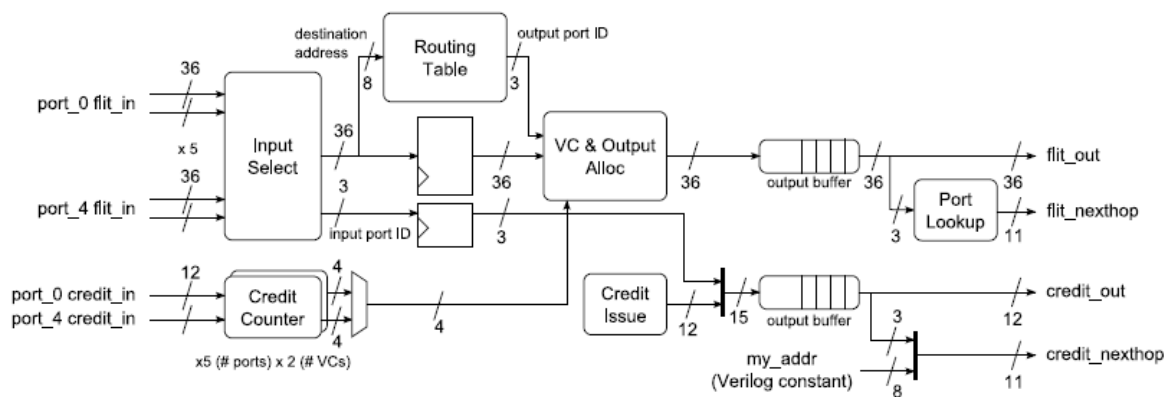


Figure 24: Router datapath

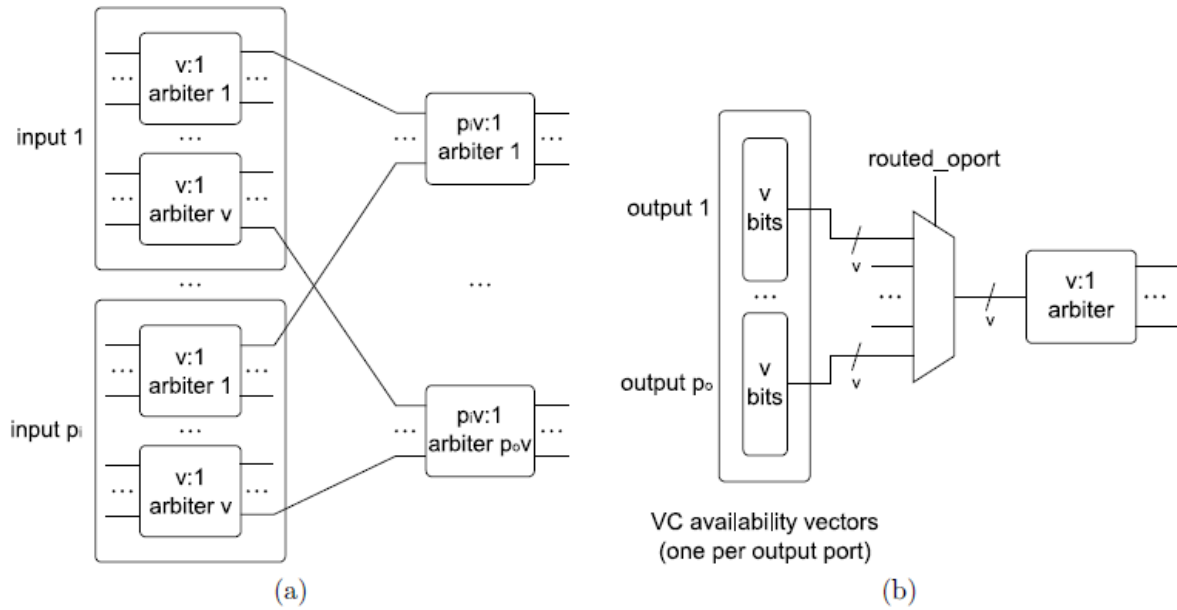


Figure 25: VC allocator implementation: (a) classic router, (b) DART Router

#### 5.3.4 Global Interconnect.

The main aim of the global interconnect is to provide uniform-latency communication within all DART nodes. DART can simulate any topology, by configuring the routing table properly. The interconnect organization is shown in the figure below, where nodes are clubbed into partitions and the partitions are connected by a crossbar. This organization is chosen to conserve area. Round-robin arbiters are used in intra-partition arbitration as well as inter-partition arbitration, flits having timestamps equal to the current simulation time are given priority. Two separate sets of arbiters are used to implement priority.

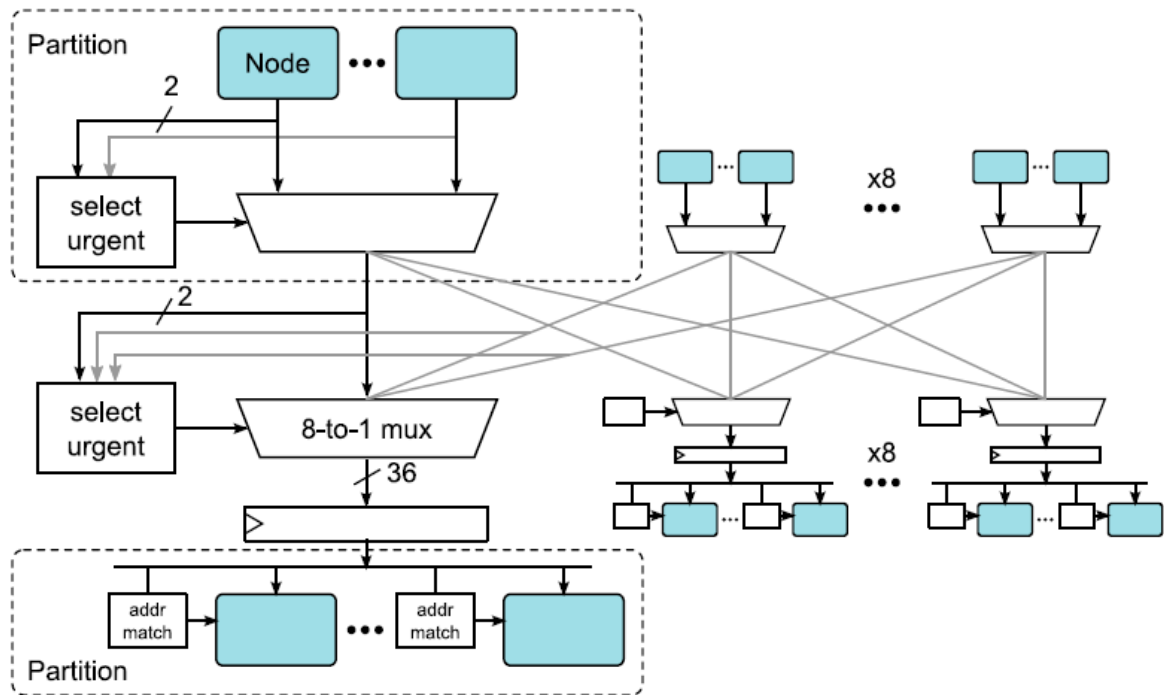


Figure 26: DART global interconnect

#### 5.4.1 RTL Schematic of flitqueue:

The diagram illustrates the FlitQueue interface with a central block labeled "FlitQueue" at the top and bottom. The interface consists of the following signals:

- Inputs (Left Side):**
  - config\_in(15,0)
  - credit\_in(10,0)
  - credit\_in\_nexthop(6,0)
  - dequeue(1,0)
  - flit\_in(35,0)
  - nexthop\_in(6,0)
  - sim\_time(9,0)
  - clock
  - config\_in\_valid
  - credit\_dequeue
  - credit\_in\_valid
  - enable
  - flit\_in\_valid
  - reset
- Outputs (Right Side):**
  - config\_out(15,0)
  - credit\_out(10,0)
  - flit\_out(1,0)
  - flit\_out(71,0)
  - flit\_out\_valid(1,0)
  - config\_out\_valid
  - credit\_ack
  - credit\_full
  - credit\_out\_valid
  - error
  - flit\_ack
  - is\_quiescent

40



### 5.4.2 RTL Schematic of traffic generator:

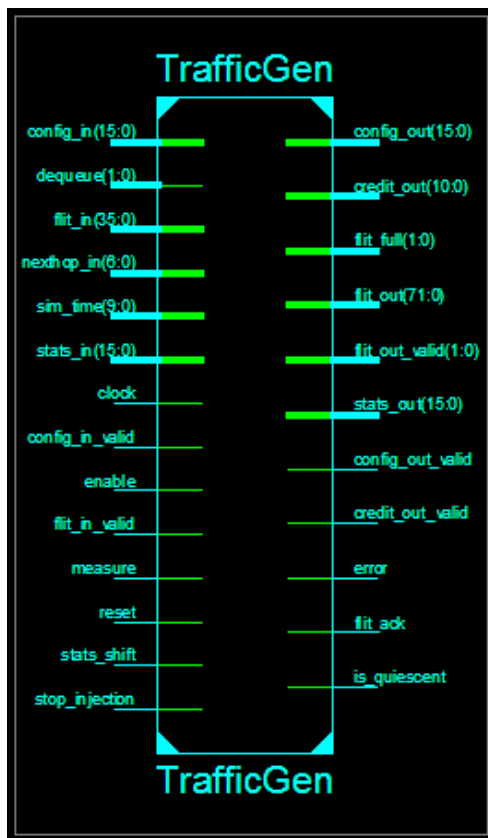


Figure 29: Block diagram of traffic generator

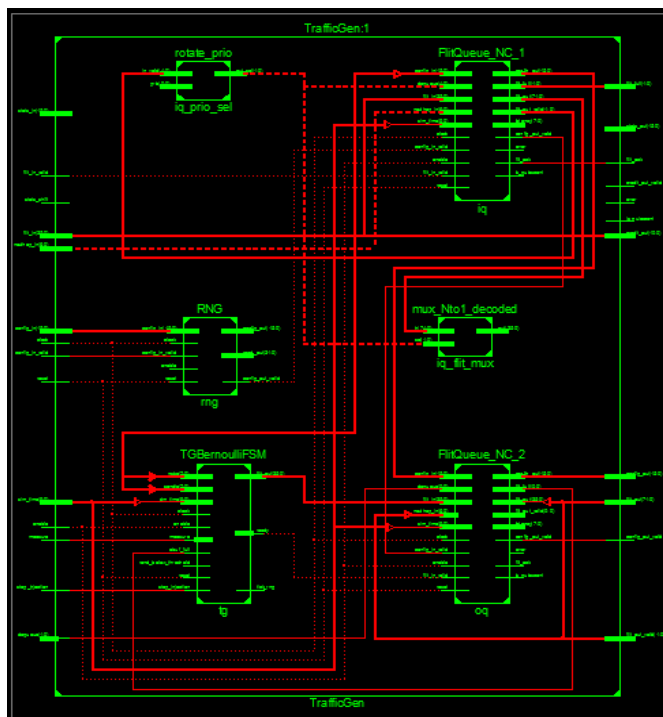


Figure 30: RTL schematic of inner blocks

## CHAPTER 6: CONCLUSION

In this thesis we have studied about different routing algorithms like XY, OE , DyAD , West first, North last, Negative first, DyXY routing algorithms. The common capability of all these algorithms are their deadlock free nature. We also discussed about a fault tolerant routing algorithm which is based on one of the deadlock free routing algorithms i.e. DyXY . We compared their performance with varying % of network load and we observed the effect on the network parameters i.e. Average Latency and Average Throughput .

We studied about the Xtranc algorithm also which was applied on Xtorus topology. It was simulated using the CONNECT simulator. The design overview was obtained and the flow was verified using testbench.

Here we have tried to simulate a NOC using a FPGA. The simulated architecture is decoupled from the architecture of the simulated NOC , by simulation time virtualization. DART eliminates the high cost of modifying and resynthesizing the hardware emulator and hence improves upon existing existing FPGA based simulators.

## CHAPTER 7: REFERENCES

- [1] **J K Singh**, A.K. Swain, T.N.K. Reddy and K. K. Mahapatra, “Performance evaluation of different routing algorithms in Network on Chip,” *IEEE PrimeAsia Proc.*, pp. 180–185, Dec. 2013.
- [2] V. Puente, J. Gregorio, and R. Beivide, “SICOSYS: an integrated framework for studying interconnection network performance in multiprocessor systems,” in *Parallel, Distributed and Network-based Processing, 2002. Proceedings. 10th Euromicro Workshop on.* IEEE, 2002, pp. 15–22.
- [3] F. Fazzino, M. Palesi, and D. Patti, “Noxim: Network-on-chip simulator,” 2008.
- [4] L. Jain, B. Al-Hashimi, M. Gaur, V. Laxmi, and A. Narayanan, “NIRGAM: a simulator for NoC interconnect routing and application modeling,” in *Workshop on Diagnostic Services in Network-on-Chips, Design, Automation and Test in Europe Conference (DATE’07)*, 2007,
- [5] Arash Farhadi Beldachi<sup>1</sup>, Simon Hollis<sup>2</sup>, Jose L. Nunez-Yanez<sup>1</sup>, eXtended Torus routing algorithm for networks-onchip: a routing algorithm for dynamically reconfigurable networks-on-chip
- [6] Duato, J., Yalamachili, S., Ni, L.: ‘Interconnection networks: an engineering approach’ (Morgan Kaufmann, 2003)
- [7] N. Agarwal, T. Krishna, Li-Shiuan Peh, and N.K. Jha. GARNET: A detailed on-chip network model inside a full-system simulator. In *Proc. Int’l Symp. on Performance Analysis of Systems and Software*, April 2009.
- [8] T.W. Ainsworth and T.M. Pinkston. Characterizing the Cell EIB On-Chip Network. *IEEE Micro*, pages 6–14, 2007.
- [9] M. Baron. The Single-Chip Cloud Computer. *Microprocessor Report*, April 26, 2010.
- [10] J.L. Hennessy and D.A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, 4th edition, 2007.
- [11] N.E. Jerger and L.S. Peh. On-Chip Networks. *Synthesis Lectures on Computer Architecture*, 2009.
- [12] Y.H. Kao, N. Alfaraj, M. Yang, and H.J. Chao. Design of High-Radix Clos Networkon-Chip. In *Fourth Proc. Int’l Symp. on Networks-on-Chip*, pages 181–188. IEEE,2010.

